Google Summer of Code 2024 Final Report

(May 2024 - August 2024)

Project Title - Revisiting and expanding nx-parallel
Organisation - NumFOCUS(NetworkX : nx-parallel)
Mentors - Dan Schult, Mridul Seth
Contributor - Aditi Juneja

1. Abstract

This report details the work accomplished during Google Summer of Code 2024, focusing on the nx-parallel project, a parallel backend for NetworkX that leverages joblib for running graph algorithms in parallel. Expanding the nx-parallel project involved integrating the joblib and the networkx's configuration systems in nx-parallel and adding a pre-commit hook to automate the `get_info` updation. Additionally, the project involved switching to setuptools for the build tool, revisiting the previously added algorithms and enhancing their performance, enabling custom chunking, adding tests, updating the docs syntax, renaming the `Dispatcher` class and enhancing the functionalities of the `ParallelGraph` class. Beyond my GSoC project, I also created nx-parallel's conda-forge feedstock and added the conda installation guide to the nx-parallel README. Apart from these, there were several notable achievements and contributions made outside the scope of the GSoC project and towards the NetworkX dispatching and the bigger NetworkX project.

The contributions made during this period are crucial for future development, and this report, along with my blogs, will serve as a valuable resource for anyone continuing this work.

2. Background

The `nx-parallel` package is a backend for NetworkX designed to optimise the execution of graph algorithms through parallel computing using joblib. As an active contributor to the project, I had previously set up the ASV benchmarking infrastructure, analysed the impact of chunking on speedups, and introduced `get_chunks` to enable the users to provide their custom chunking and added 15 parallel graph algorithms to the nx-parallel project.

Building on this foundation, my work during GSoC 2024 aimed to revisit and refine existing implementations, centralise testing, integrating the configuration systems, and improve the overall robustness of `nx-parallel`. These efforts are aimed at making `nx-parallel` more efficient, easier to use, and better integrated with other NetworkX backends.

3. Work Done

Throughout my GSoC journey, I undertook several key tasks and enhancements within the \`nx-parallel\` project. You can refer to my GSoC blogs to understand everything in detail. Below is an overview of the main contributions:

1. Revisiting nx-parallel(PR#63)

1. **Chunking and \`get_chunks\` Implementation:**
I implemented the custom chunking mechanism across all parallel algorithms in \`nx-parallel\`. This implementation involved adding a \`get_chunks\` argument to every algorithm, which allows the user to provide a custom chunking, particularly for large graphs.

2. **Centralised Testing for \`get_chunks\`:**
Developed a centralised testing framework for \`get_chunks\`, ensuring uniformity across all functions. Currently, this framework omits functions requiring additional arguments beyond the graph \`G\`, which may be addressed in future work.

3. **Enhancements in \`tournament.py\`:**
Improved the performance and accuracy of the \`is_reachable\` and \`tournament_is_strongly_connected\` algorithms, but simplifying the `joblib.Parallel` usage.

4. **Docstring Alignment with Sphinx Guidelines:**
To enhance the readability and maintainability of the code, I revised the docstrings to adhere to the Sphinx documentation guidelines. This ensures that the documentation is consistent and well-structured across the project. Additionally, this would be beneficial if in the future we will create a website for nx-parallel because we would not have to update the docstrings' format.

5. **Renaming the \`Dispatcher\` Class:**
The \`Dispatcher\` class was renamed to \`BackendInterface\` to better reflect its role within the architecture.

6. **Enhancing the \`ParallelGraph\` class:**
The \`ParallelGraph\` class was updated to handle initialization without errors, so that running \`nxp.ParallelGraph()\` or \`nxp.ParallelGraph([(1, 2), (2,

3)])` doesn't give any error and added `__str__` so that `ParallelGraph with n nodes and m edges` is printed instead of `<nx_parallel.interface.ParallelGraph object at 0x10495e6d0>` when a `ParallelGraph` is passed in `print()` (This is in experimental section of the proposal)

7. **GitHub Workflow Improvements:**
Updated the `test.yml` in the GitHub workflow to ensure that all dependencies are installed when testing.

2. Switching to setuptools(PR#69)

The build tool for `nx-parallel` was switched from `hatchling` to `setuptools`. This decision was driven by the widespread adoption and robustness of `setuptools`, especially for larger projects. Unlike `hatchling`, `setuptools` requires explicit declaration of all dependencies, which ensures that all necessary packages are included during the build process. This task allowed me to deepen my understanding of build tools and package management, particularly in the context of Python projects.(origin PR - https://github.com/networkx/nx-parallel/pull/67)

3. Integrating Configurations in nx-parallel(PR#75)

In PR https://github.com/networkx/nx-parallel/pull/75, the focus was on making `nx-parallel` compatible with NetworkX's configuration system while ensuring that users could also configure nx-parallel via `joblib.parallel_config`. This integration was crucial for maintaining consistency across NetworkX's backends and providing flexibility in configuring `nx-parallel`.

Alongside the implementation, I thoroughly documented the configuration process in the `Config.md` file. Additionally, I addressed the challenges and trade-offs of synchronising NetworkX and `joblib` configurations in the Issue https://github.com/networkx/nx-parallel/issues/76. This documentation will serve as a guide for future contributors who may wish to further refine this integration.

Initially, in PR https://github.com/networkx/nx-parallel/pull/68, I attempted to create a unified configuration system in `nx-parallel` by wrapping the `joblib.Parallel()`(inside nx-parallel algorithms) within a `with joblib.parallel_config(configs)` context, here `configs` are extracted from `nx.config.backends.parallel`. This approach made NetworkX’s config closely mirror joblib’s, giving the appearance of synchronisation between the two systems. However, this approach wasn’t actually creating a robust synchronisation layer inside nx-parallel and was rather complicating things. So, the PR#75 was created.

In this PR#75, I tried to simplify things by clearly documenting both configuration methods for `nx-parallel` and also added implementation to make nx-parallel compatible with NetworkX's config. The `_configure_if_nx_active` decorator wraps the nx-parallel function call within a `joblib.parallel_config` context only when NetworkX configs are enabled. If not, then configs set via joblib take precedence. Additionally, to make nx-parallel compatible with `joblib.parallel_config` I just had to remove `n_jobs` from the internal `joblib.Parallel`.

4. Pre-commit hook for updation of `get_info`(PR#55)

I added a pre-commit hook that automates the update of the `get_info` function, which returns metadata about the backend and all its functions. This ensures that the information is always up-to-date, reducing the potential for manual errors.

Initially, I proposed different approaches for automating this process in this issue - https://github.com/networkx/nx-parallel/issues/62 , but the actual implementation was deferred as I focused on the config system and revisiting PRs. Eventually, I revisited it and worked on adding a pre-commit hook that would run a shell script to update the get_info function when the `pre-commit` is run.

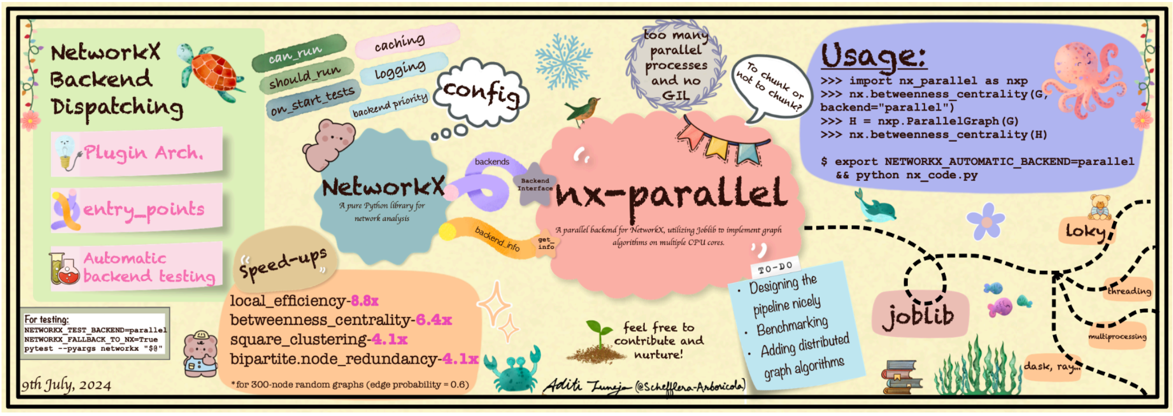
5. Creating nx-parallel's conda-forge feedstock

Although not originally part of the GSoC proposal, I contributed to the creation of a Conda feedstock for `nx-parallel` after the nx-parallel's pypi 0.2.0 release was created. (ref. https://github.com/conda-forge/nx-parallel-feedstock). Additionally, I updated the README with the conda installation instructions(ref https://github.com/networkx/nx-parallel/pull/71). This addition enhances the accessibility of `nx-parallel`, allowing users to install it via Conda, which is particularly useful for managing dependencies in scientific computing environments. And this was my first time creating a conda-forge feedstock and this helped me get a better understanding of python package management.

4. Noteworthy achievements

a. Accepted talks at EuroSciPy 2024 (with Erik) and PyData Amsterdam 2024(unable to present due to visa issues).
b. Also a co-speaker(with a lot of really smart people!) in the "Dispatching, Backend Selection, and Compatibility APIs" session at EuroSciPy 2024.
c. Became an active member of two significant open-source communities, i.e. NetworkX and conda-forge. I'm actively engaged in reviewing pull requests, resolving issues, and supporting first-time contributors within the NetworkX community.

d. Presented a poster virtually at SciPy 2024.


NetworkX Backend Dispatching
Plugin Arch.
entry_points
Automatic backend testing
can_run
should_run
on_start_tests
caching
logging
backend priority
config
NetworkX
A pure Python library for network analysis
backends
Backend Interface
backend_info
get_info
nx-parallel
A parallel backend for NetworkX, utilizing Joblib to implement graph algorithms on multiple CPU cores.
too many parallel processes and no GIL
To chunk or not to chunk?
Usage:
"Speed-ups"
TO-DO
joblib
loky
threading
multiprocessing
dask, ray...
feel free to contribute and nurture!
9th July, 2024
Aditi Juneja (@Schefflera-Arboricola)

For testing:

```
NETWORKX_TEST_BACKEND=parallel
NETWORKX_FALLBACK_TO_NX=True
pytest --pyargs networkx "$@"
```

local_efficiency-8.8x
betweenness_centrality-6.4x
square_clustering-4.1x
bipartite.node_redundancy-4.1x

*for 300-node random graphs (edge probability = 0.6)

```
>>> import nx_parallel as nxp
>>> nx.betweenness_centrality(G,
backend="parallel")
>>> H = nxp.ParallelGraph(G)
>>> nx.betweenness_centrality(H)

$ export NETWORKX_AUTOMATIC_BACKEND=parallel
  && python nx_code.py
```

- Designing the pipeline nicely
- Benchmarking
- Adding distributed graph algorithms

e. Gave a 10-minute lightning talk on "Coco and Parallel-Coco" at the PyRustLin meet-up in Delhi.

f. Delivered a 30-minute talk on "Building Backends Using `entry_points` and NetworkX's Parallel Backend" at the PyDelhi's July meet-up.

g. Gained advanced knowledge in API dispatching and parallelism within the Scientific Python ecosystem, particularly through the development of the `nx-parallel` backend (especially the config part during the GSoC period) and through documenting the dispatching developments in NetworkX.

5. Challenges and Learnings

Throughout my GSoC journey, I encountered several challenges that were instrumental in my growth. I honed my ability to communicate ideas effectively, whether through technical discussions, blog posts, or presentations. Balancing independent work with collaborative efforts allowed me to try a particular approach, seek feedback, iterate on it, refine it and then repeat.

I learned the critical importance of documenting and presenting my work clearly, a habit that has become ingrained in my workflow. This experience also taught me the value of persistence and consistency, especially when managing and prioritising tasks. I think I gained a deeper understanding of project timelines and I think I can estimate the duration of tasks more accurately.

Additionally, I developed skills in testing and maintaining a package, reviewing PRs, and differentiating between the approaches needed for first-time contributors versus experienced developers. For the latter, I found myself engaging more deeply, whereas for the former, I could often predict potential issues without needing to test every case. These experiences collectively contributed to my professional development and a deeper understanding of the open-source ecosystem.

Another challenging part for me was managing the demands of the project work, while also planning for future opportunities, all while serving as a core developer for the NetworkX project, which added another layer of responsibility. Balancing these diverse tasks and ensuring continued contributions to open source was a significant challenge, but it made the experience both demanding and rewarding.

6. Future work

All throughout my GSoC blog you will find loose threads and potential enhancements sprinkled in. And I have created or triaged issues for some of the major ones, and here's an outline of potential future directions:

- **Synchronisation of Configuration Systems**: A key area for improvement is achieving synchronisation between NetworkX and Joblib configurations in `nx-parallel`. Addressing this will enhance user experience by ensuring consistent behaviour across the two systems.
Related issue: https://github.com/networkx/nx-parallel/issues/76

- **Benchmarking and Performance**: Implementing more robust benchmarking tools, such as `conbench`, will allow for detailed performance analysis. Additionally, switching to something other than heatmaps to better represent the speedups will provide clearer insights into the efficiency of different parallel implementations.
Related issues: https://github.com/networkx/nx-parallel/issues/12 and https://github.com/networkx/nx-parallel/issues/51

- **Incorporation of Fast Algorithms**: Exploring the integration of very fast algorithms, such as `number_of_isolates` within `nx-parallel` by adding `should_run` to nx-parallel can significantly enhance the user experience. This will require careful structuring of algorithms and their respective pipelines.
Related issues: https://github.com/networkx/nx-parallel/issues/77 and https://github.com/networkx/nx-parallel/issues/79

- **nx-parallel and Infrastructure**: There is a need for better structuring of algorithms and parallelisation pipeline within `nx-parallel`.(Map-Reduce approach in https://github.com/networkx/nx-parallel/pull/7) This would not only streamline the codebase but also make it easier for future contributors to extend the library.
Related issue: https://github.com/networkx/nx-parallel/issues/30

- Several items from my GSoC proposal's experimental section are yet to be refined and developed into concrete solutions. These areas represent exciting opportunities for innovation and further development in the `nx-parallel` project.

I hope these enhancements will lead to a more robust and efficient parallel computing framework within the nx-parallel repository.

7. Conclusion

As I highlighted in my initial GSoC proposal, I believe that by providing the ability to update the configurations in `nx-parallel` to the end-users we have significantly enhanced its usability and performance. I have wrapped up my work by creating comprehensive issues and suggesting potential solutions for ongoing development. The GSoC 2024 experience has been incredibly rewarding, allowing me to contribute meaningfully to `nx-parallel` while growing as a developer and community member. The impact of this project will extend beyond GSoC, as I continue to work on `nx-parallel` and other initiatives in NetworkX and the broader Scientific Python ecosystem.

8. Acknowledgements

I would like to sincerely acknowledge my mentors, Dan Schult and Mridul Seth, for their invaluable guidance and support throughout the GSoC period. Dan's critical reviews were instrumental in enhancing my work, particularly on the configuration systems in nx-parallel. The weekly meetings and blogs provided a structured framework that facilitated continuous reflection and improvement. I also extend my heartfelt thanks to the NetworkX community for their support and encouragement. Special thanks to Erik Welch for his valuable insights and support through our discussions on backend dispatching. Lastly, I am deeply grateful to NumFOCUS and the GSoC organisers for providing this remarkable opportunity. It has been a privilege to work alongside such brilliant individuals. I hope to continue my work and keep contributing to Networkx, nx-parallel and the Scientific Python ecosystem. Thank you for your time and consideration.

9. Work links

GSoC Blogs : https://github.com/Schefflera-Arboricola/blogs/tree/main/networkx/GSoC24

1. GSoC-related

1. PRs Opened

- [Merged] https://github.com/networkx/nx-parallel/pull/75 : ENH: Adding and documenting configs in nx-parallel
- [Closed] https://github.com/networkx/nx-parallel/pull/68 : WIP: Adding config to nx-parallel
- [Merged] https://github.com/networkx/nx-parallel/pull/63 : Revisiting nxp algorithms
- [Merged] https://github.com/networkx/nx-parallel/pull/69 : MAINT: Switching to setuptools

- [Closed] https://github.com/networkx/nx-parallel/pull/67 : MAINT: adding lint.select and corresponding style fixes
- [Merged] https://github.com/networkx/nx-parallel/pull/55 - Adding a pre-commit hook to update get_info
- [Merged] https://github.com/networkx/nx-parallel/pull/78 - Ignoring functions in utils for the get_info dict
- [Merged] [PR#7226](https://github.com/networkx/networkx/pull/7226) - Remove parallelization related TODO comments
- [Closed] - https://github.com/networkx/nx-parallel/pull/74 - ENH: added colliders and v_structures
- [Closed] https://github.com/networkx/nx-parallel/pull/61 - WIP: Updating timing scripts and heatmaps

2. Issues Raised

- https://github.com/networkx/nx-parallel/issues/62 - Possible approaches for automating the updation of the get_info function
- https://github.com/networkx/nx-parallel/issues/51 - Inconsistent heatmaps and the current timing script
- https://github.com/networkx/nx-parallel/issues/76 - Synchronizing NetworkX and Joblib configurations in nx-parallel
- https://github.com/networkx/nx-parallel/issues/77 - Incorporating should_run in nx-parallel
- https://github.com/networkx/nx-parallel/issues/79 - Addressing Slower Parallel Implementations in nx-parallel Compared to NetworkX

2. Outside GSoC

1. PRs Reviewed

- Add parallel version of edge_betweenness_centrality - https://github.com/networkx/nx-parallel/pull/60
- Adds initial debug logging calls to _dispatchable - https://github.com/networkx/networkx/pull/7300
- Various improvements in information centrality - https://github.com/networkx/networkx/pull/7475
- Add "networkx" backend for dispatching - https://github.com/networkx/networkx/pull/7496
- Add nx.config.backend option - https://github.com/networkx/networkx/pull/7485
- Fix dispatch tests when using numpy 2 - https://github.com/networkx/networkx/pull/7506
- Strong product docs update - https://github.com/networkx/networkx/pull/7511
- Fixed the citation in dominance.py - https://github.com/networkx/networkx/pull/7524
- Prettify README.rst - https://github.com/networkx/networkx/pull/7514
- Update NetworkX reference links in doc index - https://github.com/networkx/networkx/pull/7500
- [ci skip] adding user @MridulS - https://github.com/conda-forge/nx-parallel-feedstock/pull/3

- Minor doc/test tweaks for dorogovtsev_goltsev_mendes - https://github.com/networkx/networkx/pull/7535
- More accurate NodeNotFound error message - https://github.com/networkx/networkx/pull/7545
- Add Introspection section to backends docs - https://github.com/networkx/networkx/pull/7556
- DOC: Rm redundant module from autosummary - https://github.com/networkx/networkx/pull/7599
- Log "can/should run" and caching in dispatch machinery - https://github.com/networkx/networkx/pull/7568
- Ensure we always raise for unknown backend in `backend=` - https://github.com/networkx/networkx/pull/7494
- DOC: Fix typo in the code snippet provided in the docstring of nx_pydot.pydot_layout() - https://github.com/networkx/networkx/pull/7572
- Addition of DomiRank centrality to NetworkX - https://github.com/networkx/networkx/pull/7443

2. Merged PRs of mine

- MAINT: updated README with Conda installation guide - PR#71: https://github.com/networkx/nx-parallel/pull/71
- DOC: Clarifying NetworkXPointlessConcept exception - PR#7434: https://github.com/networkx/networkx/pull/7434
- Minor doc_string changes to check write access - PR#70: https://github.com/networkx/nx-parallel/pull/70
- Renaming - PR#7492: https://github.com/networkx/networkx/pull/7492
- Made `plot_image_segmentation_spectral_graph_partition` example compatible with scipy 1.14.0 - PR#7518: https://github.com/networkx/networkx/pull/7518
- Added `default_config` in `get_info`'s description - PR#7567: https://github.com/networkx/networkx/pull/7567
- Added nx-parallel to conda-forge - PR#26768: https://github.com/conda-forge/staged-recipes/pull/26768

3. Issues Raised

- Updating prettier in .pre-commit-config.yaml - https://github.com/networkx/nx-parallel/issues/73
- Dispatch test failing due to graph `adj` comparison for some algorithms in `expanders.py` - Issue #7505: https://github.com/networkx/networkx/issues/7505
- Fix the docs of `strong_product` - Issue #7510: https://github.com/networkx/networkx/issues/7510
- Clarification of `networkx.karate_club_graph()` dataset - Issue #7519: https://github.com/networkx/networkx/issues/7519

4. Discussions

- Commented: Adding a new algorithm based on the paper: "Solving the Many to Many assignment" - https://github.com/networkx/networkx/discussions/7450
- Commented: GraphML does not support type <class 'type'> as data values - https://github.com/networkx/networkx/discussions/7439
- Commented: NetworkXError: '_key' is not a valid key - https://github.com/networkx/networkx/discussions/7440
- Answered: Description on strong product - Discussion #7509: https://github.com/networkx/networkx/discussions/7509

5. Issues Resolved

- Not compatible with Numpy 2.0 - Issue#7501: https://github.com/networkx/networkx/issues/7501
- "dispatch" decorator needs documentation for contributors and readers of code - Issue #7189: https://github.com/networkx/networkx/issues/7189
- Create a conda-forge feedstock for nx-parallel - Issue #65: https://github.com/networkx/nx-parallel/issues/65
- @conda-forge-admin, please add user @MridulS - Issue #2: https://github.com/conda-forge/nx-parallel-feedstock/issues/2

6. Open PRs of mine

- Updated SPEC 2 - PR#334: https://github.com/scientific-python/specs/pull/334
- Poster: Parallel Graph Algorithms and Building Backends with Entry Points - PR#981: https://github.com/scipy-conference/scipy_proceedings/pull/981
- nx-j4f : https://github.com/Schefflera-Arboricola/nx-j4f