Davis Math Lab Project Report, Spring 2026
Making passagemath more robust and easier to use

Samuel Chen
Undergraduate Scholar

Matthias Köppe
Faculty Mentor

June 17, 2026

Contents

1 Introduction

Passagemath is a modular, pip-installable repackaging of SageMath, the open-source math software. Rather than installing all of Sage, users can install just the parts of passagemath they need and run it in plain Python environments like Google Colab that can interact with other Python tools. Math researchers, educators, and students use it for symbolic computation, combinatorics, linear programming and more.

Professor Köppe leads the passagemath undergraduate group, which works on adding features or fixing bugs in the software to make it easier to use. The group also uses passagemath to explore math by writing notebooks and running experiments.

This quarter, I contributed to fixing bugs in the codebase, building tools and materials to help the group, and exploring how to make linear programming easier to use in passagemath.

2 Progress

2.1 Code contributions

I made 13 pull requests that were merged into the passagemath codebase. My approach to choosing what to work on was to prioritize based on scale, neglectedness, and fit. In the beginning, this looked like infrastructure problems that affected many areas and bugs from passagemath being modular. I used AI tools to find widespread codebase errors and read long CI logs that made it more feasible to work on the most important things. This was very helpful for getting to know the codebase. The code contributions fall into four themes.

2.1.1 Modular installation

Passagemath's modular design means code that assumed a full SageMath install could break when dependencies were missing. A common bug was a function trying to import an optional package, but the import doesn't happen, so the function errors when it is called but never assigned. I fixed this pattern for different files relating to combinatorics, number theory, and others. To help with this, I made `check_unbound_imports.py`, an abstract syntax tree tool that scans the codebase for this pattern, and categorized issues that needed attention or false positives.

2.1.2 Build and CI infrastructure

A newer version of setuptools deprecated `pkg_resources` and this caused Linux build crashes. I replaced it with `importlib.metadata`. I also discovered a Windows CI failure caused by an invalid path format that affects multiple packages across many releases. Professor Köppe made the fix based on that diagnosis.

2.1.3 Plotting in Python kernels

In Colab and JupyterLite, passagemath plots used to show up as text (`Graphics object consisting of 1 graphics primitive`) instead of images. The fix was to add the method (`_repr_png_` and `_repr_svg_`) to the `Graphics` class so python knew where to get the images. Now plots render as images in Colab and JupyterLite. I presented the plotting fix in the mid-quarter presentations.

2.1.4 LP ergonomics

I documented dual values in the passagemath linear programming guide and describe this more in the linear programming ergonomics section below.

2.2 Supporting the team

Later, my approach was to be a multiplier and help teammates get started fast.

2.2.1 Onboarding documents and exercises

When students joined the group at the beginning of Spring 2026, I made documents to help teammates set up their passagemath research environment, made a list of shovel-ready issues that described the problem and a potential approach, and also made exercises that introduced generalized bug patterns to familiarize teammates with the codebase.

I aimed for the exercise notebooks to be pedagogically great. I designed them so students could see the problem themselves, why it happens, and fix it. The goal was to give new contributors something concrete to do quickly by editing real code and fixing a real problem instead of reading documentation and files.

A common problem for teammates was getting the environment set up. This involved jupyter, git, and things tended to get messy so we spent a lot of time debugging. Several teammates used these docs to get their environment running.

I also helped a teammate, Runze (Tony) Yu, navigate the codebase and git workflow while he worked on a shovel-ready issue. His PR (#2358) was merged.

2.2.2 pm-explore

The first assignment Professor Köppe gave us was to make a notebook exploring some part of passagemath. Getting the environment to work with the right kernel got a lot of people stuck and a lot of time was spent on debugging environment problems. This inspired me to make `pm-explore` which automates this.

Given any passagemath file, `pm-explore` automatically generates a jupyter notebook with the environment set up so people can see what methods the file has and can immediately run and modify methods. It uses an abstract syntax tree to look at the text, so it works even when the environment is broken. Afterwards, I learned that this is similar to the runnable code in the passagemath documentation website. The `pm-explore` tool is good for having it work on your own computer like a student researcher who wants to

explore the codebase would use. Students can also make changes to the notebook to explore the methods. One teammate, whose work was mainly exploring passagemath features, used `pm-explore` for that.

2.3 Linear programming ergonomics

Eventually, I decided to work on LP ergonomics because I wanted to work on something in depth.

I was taking MAT 168 Optimization and tried to use passagemath instead of cvxpy. You had to write nested loops and keep track of indices often, which made it difficult to use. Some methods you would like easy access to like getting dual values had to be found in the backend (Issue #2347).

I thought it could be worth working on because this was a place where passagemath could really be improved. Passagemath can handle symbolics and is integrated with other math things. Passagemath can do exact arithmetic over rationals and number fields, while cvxpy only handles floats. This whole idea is very big though. There would be architectural and frontend changes that would take careful design considerations. I would have to familiarize myself with what was currently going on. My plan was to edit the documentation so it would be easier to find the dual values (PR #2353). Next, I tried tracing through the backend to understand what was going on, but it was very difficult since the files were scattered. The next step was to work through linear programming problems myself to identify specific pain points that I could raise and propose potential solutions for. I struggled to do this since I was already using cvxpy for class and doing it in passagemath was harder and extra work. I concluded that this was a problem worth solving, but making progress and designing it carefully would take understanding that I didn't have yet. This made me take a step back and consider other things to work on.

2.4 Connection between backpropagation and linear programming

One idea I had was exploring how backpropagation and linear programming relate to each other. In backpropagation, gradients are computed using the chain rule. While doing LP in MAT 168, I noticed the simplex method gives dual values similar to the gradient. The dual value is how much the optimum moves if you loosen a constraint. Doing the forward pass in backpropagation is similar to using simplex to get the optimum, and reading the dual values from the final simplex dictionary is similar to the backward pass.

3 Future Plans

I plan to present work at the Math Lab poster session in Fall 2026.

4 Selected Work

A few representative samples across the kinds of work I did this quarter.

Code contribution. Plotting fix (PR #2351): added `_repr_png_` and `_repr_svg_` so plots render as images in plain Python kernels like Colab and JupyterLite.

Codebase tool. `check_unbound_imports.py`: an AST scanner that finds the modular-install bug pattern across the codebase (added in #2283, documented in #2292).

Exploration tool. pm-explore: generates a runnable Jupyter notebook from any passagemath source file with the environment preconfigured, and works even when imports are broken.

Exercise. importerror-fix-pattern: a guided notebook where a new contributor fixes a real instance of the modular install bug. I endorsed this one because it teaches a pattern that generalizes across the codebase.

Supporting the team. Onboarding/setup docs and a list of shovel-ready issues with problem descriptions and potential approaches for new teammates.

Research writing. LP dual-values documentation (PR #2353): documented how to retrieve dual values, the first step of the LP ergonomics work.

5 References

Modular installation robustness

- #2253 — Fix NameError in `Partitions.cardinality()` when passagemath-flint is absent.
- #2282 — Fix unbound pari NameError in five modules; lazy-import sympy in `calculus_method`.
- #2283 — Fix unbound NameError in `padic_extension_leaves` and `multi_polynomial_ideal`; add AST checker.
- #2287 — Fix pickling regression introduced by `lazy_import` in `calculus_method.py`.
- #2293 — `polynomial_quotient_ring`: add `# needs sage.modules` doctest guards.
- #2292 — docs, tox: document `check_unbound_imports` and add tox env.
- #2354 — Add missing `sage.libs.linbox` doctest guards in modsym and geometry/cone.

Build / CI

- #2237 — build: replace `pkg_resources` with `importlib.metadata` in configure check.
- #2239 — `PIP_FIND_LINKS=file://$SAGE_SPKG_WHEELS` yields invalid uri on Windows.

Plot display

- #2351 — Fix plot display in plain Python Jupyter kernels.
- #2366 — plot: add `_repr_svg_()` for plain Python Jupyter kernels.
- #2368 — plot: fix circular import in plain Python by pre-importing `sage.structure.element`.

LP ergonomics

- #2353 — `glpk_backend`: document `get_row_dual` via `MixedIntegerLinearProgram`.
- #2347 — MIP workflow is harder than cvxpy for matrix-structured LP problems.
- lp-level2-prep notes.

Numerical correctness

- #2356 — `affine_homset`: fix complex-tolerance comparison in numerical point filtering.

External package

- numerical-interactive-mip #6 — backends: correct exception types in dictionary constructors and pivot update.

Teammate PR I mentored

- #2358 — graphs/generic_graph: fix `weighted_adjacency_matrix(vertices=True)` crash on sparse graphs (authored by Runze (Tony) Yu).

Davis Math Lab Project Report, Spring 2026

Polyhedral geometry and optimization in Passagemath

Viktor Becking

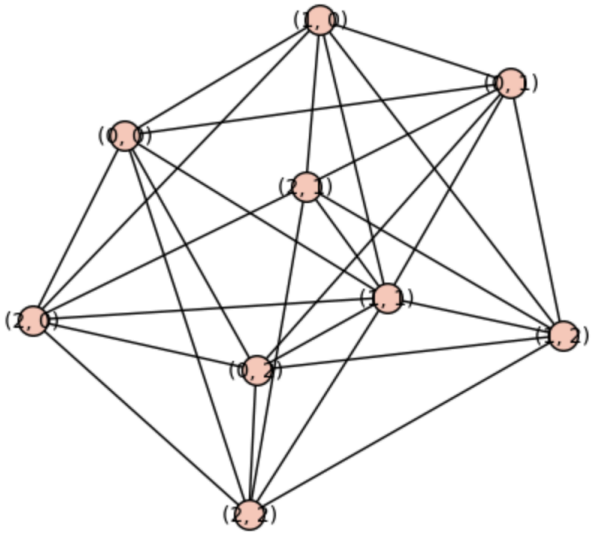
Matthidas Koeppe

Introduction:

My goal for this quarter was to familiarize myself with the passagemath repository. To do so, I explored several functions, including but not limited to chessboard graphics, world map plotting, platonic solids, general 2d plotting functionality, and general 3d plotting functionalities (for which I encountered an error).

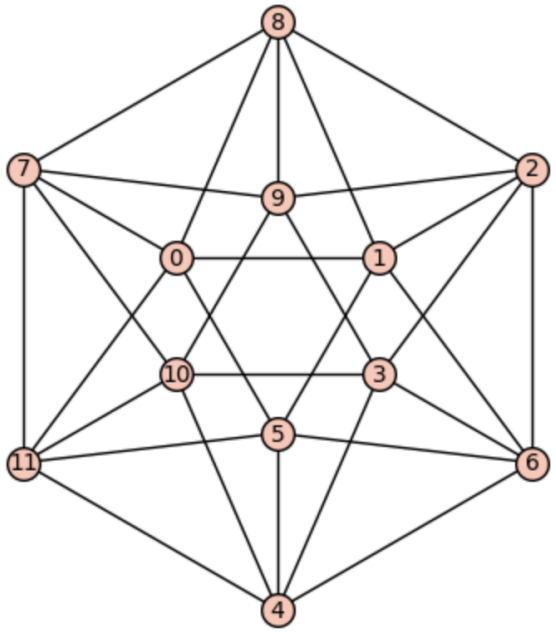
Progress:

I first set up my workspace, where I had some issues with my initial setup and finding the correct kernels to run code. After the setup, I was able to test the sage.graphs.generators.chessboard program with Samuel Chen, who helped me get the code running. This program shows how each piece moves on a d-dimensional chessboard. The figure below shows how a queen placed on 0,0 will move to other cells on a 3x3 chess board.



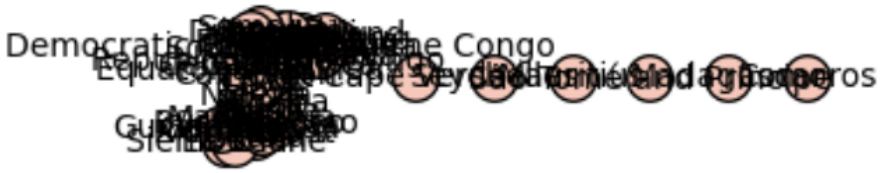
I was then tasked with exploring the passagemath repository. I found the plotting functionalities to be quite interesting, so created a Jupyter notebook that ran the platonic_solids() function in sage.graphs.generators. I coded various tests to ensure that the outputs were indeed platonic solids, which I presented on. The figure below shows the plot on an icosahedron and a test I ran (checks how many edges leave each vertex) to ensure it was a platonic solid.

```
Name of graph: Icosahedron
Number of vertices: 12
Number of edges: 30
```



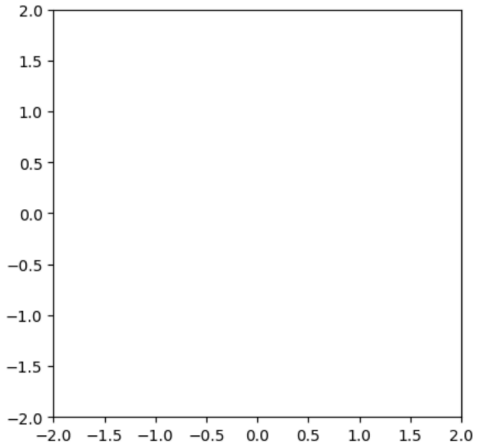

```
Name of graph: Icosahedron
Degrees of vertices: [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
All vertices have the same number of exiting edges: True
```

Later, I found another function, worldmap() in sage.graphs.graph_generators that creates a map of the world. Every country is a node, and every border is represented by a connection between the nodes. Interestingly, all the nodes and labels were overlapping, so it was completely impossible to understand. I found other ways of displaying the 2d plots so that all the nodes did not overlap in an illegible way. Below are the images of the Africa map before and after changing the display methods.



I also wanted to extend my study to 3d plotting functionalities. I looked at various plotting functions and decided to create a Jupyter notebook on the functions in sage.plot.plot3d.plot3d and sage.plot.arc. With the Arc() function in the sage.plot.arc folder, I was able to display a grid. I could never get the arc to actually plot as shown in the examples in the code itself despite the text data saying it should be within the boundary.

```
Arc with center (0.0,0.0) radii (1.0,1.0) angle 0.7853981633974483 inside the sector (1.5707963267948966,1.5707963267948966)
```



When using the plot3d sage.plot.plot3d.plot3d function, I ran into an issue that I later learned was already documented and fixed for the 2d plotting and recently documented for the 3d plotting. Passagemath has trouble displaying 3d plots and only outputs the words “Graphics3d Object,” without actually displaying anything. While this error was partially fixed on 2d plots, it still remains unresolved on 3d plotting functionalities.

```
Graphics3d Object
```

Future Plans:

Now that I am more familiar with the passagemath repository, I plan to contribute and find/fix other documented errors. Here is a list of things I plan to do:

- Fix/find a workaround for the 3d plotting object issue.
- Become more familiar with shell environments (e.g., zsh and bash).

- Figure out why “from sage.plot.graphics import Graphics” does not work. This issue caused me to use “import passagemath_graphs” graphics instead.
- Fix/figure out the issue with the Arc() function and why it is not showing up on the plot despite being well within the range.
- Read Understanding and Using Linear Programming by Jiří Matoušek and Bernd Gärtner. As well as, Linear Programming Foundations and Extensions by Robert J. Vanderbei, to become more familiar with the subject matter.

References:

Passagemath: https://github.com/passagemath

Davis Math Lab Project Report, Spring 2026

Polytope embedding algorithm in passagemath

Qihan Guan
Matthias Köppe

Introduction

My goal was to understand and implement a tensor embedding algorithm from a 2006 research paper (De Loera & Onn) as a module in passagemath, and to work toward upstreaming it.

My core goal for this project was not really the implementation itself, as the algorithm is already laid out clearly in the paper. Instead, I spent most of my time understanding the mathematics and the motivation behind the problem and the construction steps. At the start I spent considerable time practicing and reaffirming my understanding of the linear algebra prerequisites and related concepts such as affine isomorphism. I then implemented the algorithm in SymPy to debug the construction steps, before reimplementing it for passagemath. Much of my understanding, progress and testing is documented in my midterm presentation (pdf), docstring/comments and the accompanying Colab notebooks 1, 2

Progress

The theorem I implemented embeds any rational polytope $P = \{y \geq 0 : Ay = b\}$ as a face of a 3-way transportation polytope. This space consists of non-negative tensor entries $x_{i,j,k}$ constrained by fixed marginal sums: $\sum_{i,j} x_{i,j,k}$, $\sum_{i,k} x_{i,j,k}$, and $\sum_{j,k} x_{i,j,k}$. The original polytope P is the "face" formed by forcing a specific set of tensor entries $x_{i,j,k}$ to 0.

The full algorithm requires three stages, I implemented the first two. First, a coefficient-reduction stage ensures the construction remains strictly polynomial-time. Second, the transportation embedding constructs the actual tensor and calculates the exact slice-sums necessary to translate the original equations $Ay = b$ into marginal constraints. The final code is a single module on my passagemath fork.

Future Plans

Discuss with Professor Köppe about possibility of merged upstream and changes if needed.

References

De Loera, J. A. & Onn, S. All Linear and Integer Programs Are Slim 3-Way Transportation Programs. SIAM J. Optim. 17(3), 806–821, 2006.

Wu, Y. & De Loera, J. A. Turning Mathematics Problems into Games: Reinforcement Learning and Gröbner Bases Together Solve Integer Feasibility Problems. 2022.

passagemath: https://github.com/passagemath

Davis Math Lab Project Report, Spring 2026
Fixing Two Annoying Graph Module Bugs in passagemath

Runze Yu

Faculty Mentor: Professor Matthias Koeppe

Contents

1 Introduction

In this quarter, I mainly fixed two bugs in the graph modules. These two issues have been quite annoying for a while, making our flow calculations unstable and causing our matrix run to crash whenever things got a bit large and sparse.

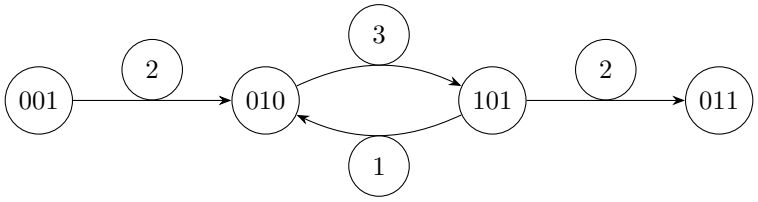
The two most important things I do is that the max-flow algorithm (PR #2358). The wasteful internal loops were messing up our edge ordering. And a mismatch crash (PR #2108) in the sparse matrix builder when handling custom vertex keys. Both patches are now fully merged and all CI tests are green.

2 Progress

Fixed these issues and cleaned up the messy edge-ordering behavior in the upstream repository. I gave a presentation in the math lab about the max-flow algorithm.

2.1 Cleaning Up Wasteful Flow Cycles (PR #2358)

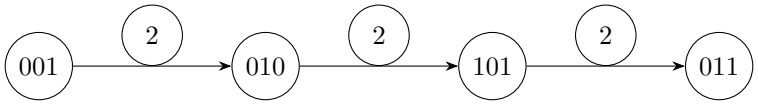
The max-flow algorithms can generate waste flow, for example, sending 3 units one way and bringing 1 unit back. Few people notice that it leaves behind these internal loops. It didn't change the net flow, but it made the edge uncontrollable and slowed down visualizations.



I added a quick post-processing cleanup pass to solve this problem. The code now finds the cycle, catches the minimum flow along the loop, and uses the maximum flow to subtract it to kill the redundant backward edge:

$$\Delta = \min(\text{flow}_{010\to101}, \text{flow}_{101\to010}) = \min(3, 1) = 1 \tag{1}$$

By subtracting 1 from both sides, the flow on $101 \to 010$ drops straight to 0. Since sinks like 011 don't have any leaving edges, they are immune to cycles. This simple cleanup makes the whole algorithm predictable.



2.2 The Sparse Matrix Builder Crash (PR #2108)

1	2	3
4	5	6
7	8	9

Dense Matrix
(stores all entries, size n^2)

1		
	5	
		9

Sparse Matrix
(stores only non-zero values, memory efficient)

The second one is a pretty hidden part. SageMath automatically switches from dense matrices to sparse structures when a graph grows larger than 256 vertices to save memory. The code ran fine normally, but when the user passed custom vertex orderings via `vertices=True`, the builder got confused and tried to run a dense iteration loop directly on sparse data.

The system would instantly crash with a confusing traceback: `TypeError: dense iteration is not supported for sparse input`. The fix was pretty straightforward: we can intercepted the block and forced `sparse = False` whenever custom keys are provided, keeping the matrix builder happy:

```
# Quick fix: turn off sparse flag when custom vertex keys are specified
if keys is not None:
    sparse = False
    kwds = copy(kwds)
    kwds["row_keys"] = kwds["column_keys"] = keys
```

2.3 Doctest and CI Fixes

While fixing these, I also ran into some minor test failures. The geometric layout solver `_layout_bounding_box` kept throwing small floating-point precision differences across different machines (like returning `3.999999999999999` instead of `4`). Fixed that by adding standard ellipses (`...`) to the docstring. Also threw a `sort=True` into `_build_flow_graph` to stop dictionary hashing from randomizing our test outputs.

3 Future Plans

Add a bunch of random graphs right at the $n = 256$ threshold to ensure the dynamic dense-to-sparse switching never breaks again, and plan to read some linear programming books to build a strong foundation about the basic information.