[PYSAL] - Produce Routing Engine for `spopt`

Google Summer of Code 2025 Proposal

Abstract

In the Python Spatial Analysis Library (`pysal`), the `spopt` package is concerned with the formulation and solving of various spatial optimization problems, particularly regionalization and location-based problems. This project proposes to enhance the `spopt` package with a `route` module, which will function as an API to set up and solve vehicle routing problems (VRP), including parameterizing vehicles, depots, and routes in various VRP setups. An example of the VRP can found in this Guinness delivery example. In order to accomplish this, the GSOC student will adapt code that has been produced by the project mentor(s) which currently relies on an Open Source Routing Machine (OSRM) instance running in the background. The primary goal of this project is to generalize the routing framework such that a user can provide precomputed duration and distances matrices with objectives and constraints natively in `spopt`.

A secondary (*stretch*) goal of this project is to also produce code and instructional tutorials for the VRPs in education research. For this, we can adapt the VRP framework such that the depots include origins (bus depots), supply depots (student bus stops), and final destinations (schools). Additional optimization problems will be considered, such as the student assignment-based Segregation Minimizing Problem.

GSOC Information

Associated organization

NUMFOCUS

GSOC page

Personnel

Project Mentors

- **Levi John Wolf**

Associate Professor, University of Bristol
email: levi.joh.wolf@bristol.ac.uk
github: https://github.com/ljwolf

- **Germano Barcelos**

Master Computer Science Student, Universidade, Federal de Viçosa
email: gegenbarcelos@gmail.com
github: https://github.com/gegen07

GSOC Student

- **Dylan Skrah**

Joint Doctoral Student, San Diego State University
email: dskrah5938@sdsu.edu
github: https://github.com/fiendskrah

Technical Details

`spopt's` Routing Engine

This project will enhance the `spopt` package in `pysal` with a `route` module. This module will be designed to create and solve problems from user provided data.

The project mentor(s) have already done some work to produce the code required to accomplish this for personal use, and seek to generalize the code as a new module in `spopt`. Because of this, the GSOC student is not starting from scratch. Some code shared to an issue thread contains a suggested example call:

```
problem = Route(
    depot_location=(9.2413, 52.2921),
    depot_close=pandas.to_datetime("2025-01-02 20:00:00"),
    depot_name='My Depot',
).add_clients(
    data.geometry,
    data.demand_volume,
    data.time_windows,
    data.source.index,
    data.service_times
).add_trucks_from_frame(
   truck_df
).solve()
```

In this example API call, `problem` is an object which contains all of the information and the solution to the VRP. The problem is primarily being constructed through `Route` and depot variables are being added/adjusted through `Route's` arguments. Functions are called as extensions to `Route`to add client and truck data, and then a `solve` function signals to `Route` that the parameterization is complete and the model is ready to be solved.

Something that is not exposed in the API is how the travel network is initialized for the problem. The depots and clients exist as static points on the network, and the trucks traverse the network to deliver product from the depots to the clients, but the network needs to be included in the call so that the optimization solver can weigh the total travel costs of different problem solutions.

Existing code

Upon inspection of the draft pull request, the routing engine code lives in `engine.py`. This file contains functionality to generate the distance/duration matrices that represent the network required to initialize a VRP in two key functions. `build_route_table()` and `build_specific_route()`. `build_route_table` requires the target nodes in the network, which can be provided by the arguments `demand_sites` and `candidate_depots`. The function takes these nodes and locates them using the `get_coordinates()` function. Similarly, the `build_specific_route()` function constructs a route between each waypoint provided in a geopandas GeoDataFrame or GeoSeries object by connecting to OSRM via the http interface or through `py-osrm` if the http fails.

Looking at these two core functions, the routes are generated from a travel network using the Open Source Routing Machine (OSRM). The existing code relies on an OSRM instance to produce the distances and durations of routes along the network. This connection is first attempted as an http call, but otherwise the functions connect to OSRM through the `py-osrm` package (link to repository).

This constitutes the OSRM instance as a core dependence of this module, along with the 'portal' used to initialize the instance, in this case `py-osrm`. It is anticipated that the decision of how to handle the generalized implementation of the travel network will comprise the bulk of work on this project. Two potential solutions are suggested below.

Retaining C++ power via `nanobind`

OSRM functions on C++, which is generally more efficient for handling heavy computational load relative to python. Because optimization can quickly become computationally taxing, particularly if heuristic solutions are undesirable, there is an argument in favor of simply retaining the relationship to OSRM, because it means that the user can 'drop in to' C++ from python syntax.

The nanobind package is designed to be an interface between the two languages, meaning that python syntax can be leveraged to call functions written in C++. The current code already does this via its core functions `build_route_table()` and `build_specific_route()` through `py-osrm`, but consistent functionality is currently limited to specific operating systems.

Because this would leverage code that has already been written, it is an attractive solution. However, the project proposes to add `pyvrp` and OSRM via `nanobind` and `py-osrm` as dependencies; the impacts of this dependence to the future maintenance burden of `spopt` is uncertain. The potential for updates to the ecosystems of the other packages suggest that maintaining this functionality could be impacted going forward.

Replacing the `OSRM` engine

An alternative solution would be to replace the OSRM element with something native to python. This solution would reduce, but not eliminate, the dependence on other projects. Because the remaining dependence would be with python packages, replacing the current engine avoids the need to bridge the gap between python and C++, and is less burdensome from a future-proofing perspective. One option suggested in the issue thread was to implement the routing engine via `routingpy` (link), a python client for interfacing with popular web services to retrieve distance/duration matrices.

This option requires addressing a major issue: Which of the online providers will be used for obtaining the travel network? This invites a subquestion: should `spopt.route` maintain support for multiple routing services? `routingpy` supports OpenStreetMap, Google Maps, Mapbox, and more, so building the engine around `routingpy` could mean that the `spopt.route` user could choose which of these mapping services to ping for the travel network. The relative accuracy of the networks provided between these services is unknown.

Additionally, routingpy has recently been archived, making future functionality uncertain. From the README of the main branch of the routingpy github repository:

> "In case anyone has interest to take over this repository, please write me on my public email. To be eligible you should have a proven track record of FOSS python projects and ideally some PyPI package management experience."

Of the extant services, OpenStreetMap (OSM) is perhaps the most ubiquitous in the open source world because of its ready availability. If it is decided that reliance on only one network source is acceptable, OSM is likely the path of least resistance, due to the abundance of code that has already been written around it. Other packages such as pandana offer effective OSM loaders that could be considered as routing engine drivers.

VRP and Optimization

Under discussed thus far are the quantitative methods underlying the proposed `spopt.route` module, which sit atop the `pyvrp` package. A quote from `pyvrp's` introduction to vrp:

> "various heuristics, metaheuristics, and exact methods have been developed to tackle the VRPs, including but not limited to, local search, genetic algorithms, and branch-and-cut algorithms."

The two general categories of algorithms are those that seek *heuristic* solutions, meaning estimated to be good, and those that seek *optimal* solutions, meaning precisely the best, relative to some quantified stated criteria. The decision of which algorithm to use depends on the specific application. In terms of the existing algorithms that can find optimality, the GSOC student is most knowledgeable about branch and bound. Learning about optimization through this project is an opportunity where guidance from the project mentors will be pedagogically impactful to the GSOC student.

Stretch Goal: Support for Other Route-Based Problems

The primary deliverable of this project is the implementation of the variants of the VRP, including Capacitated VRP, Split Delivery VRP, and Prize Collecting VRP. The application of the VRP represents pedagogical

value for the GSOC student, but the generalized implementation of the routing engine would enable the application of other route based models, i.e. optimization applications that require distances and durations from mapping services. Therefore, a stretch goal for the GSOC student would be to additionally write code to leverage a generalized routing engine for other problems, such as the project described in the *School District Optimization for Exploratory Spatial Data Analysis* section below. The decision to pursue this will be discussed and finalized during the *Community Bonding and Scoping Period*.

Schedule of Deliverables

Community Bonding and Scoping Period [May 8 - June 1]

The primary purpose of this period is to reach decisions about the issues raised in the *Technical Details* section and fill any gaps in the GSOC student's knowledge . The GSOC student will meet with the advisor(s) to itemize the project into tractable tasks which will populate the two phases of the project with various deadlines. Progress on the project will be tracked with regular postson my blog.

Deliverable targets

- Finalized decision about replacing the routing engine or further integrating with OSRM
- Finalized GSOC schedule of tasks and deliverable deadlines

Phase 1 [June 2 - July 14]

The primary deliverable of phase 1 is a working version of the generalized routing engine, which will be capable of initializing, parameterizing, and solving a VRP.

Deliverable target

- Generalized routing engine
- Tutorial jupyter notebook demonstrating implementation of VRP

Phase 2 [July 14 - Aug 25]

The phase 2 deliverable will iterate on the routing engine by implementing additional models and edge case functionality and reaching for the stretch goals outlined above. At the outset of phase 2, the GSOC student will meet with the mentor(s) to determine the extent of the stretch goals that will be pursued.

Deliverable target

- Tutorial jupyter notebooks demonstrating implementations of CVRP, Split Delivery VRP, and Prize-Collecting VRP
- Tutorial jupyter notebook demonstrating implementation of Segregation Minimizing problem

Final Week [Aug 25 - Sept 1]

- Final sign-off blog post
- Final commit and pull request

Development Experience

I have two projects on Github that are relevant to this proposal. They demonstrate my competency in collaborative coding projects (i.e. the git cycle) as well as in the relevant coding language (i.e. python modules with large data handling requirements) and application contexts (i.e. spatial optimization). Both of these projects are tied to dissertation research, which I aim to publish in academic journals. Because of this, I do not wish to make the repositories public until the associated work is published.

While I do not wish to make these repositories public at this time, I am happy to invite the project mentor(s) as well as any relevant NUMFOCUS or GSOC associates who are a part of the application decision process via their github accounts to inspect the repositories. Find a brief description of these projects below.

School District Optimization for Exploratory Spatial Data Analysis

Link to project repository

This project aims to apply a specific spatial optimization problem as an exploratory technique, rather than as a prescriptive solution, which is the usual application in education settings. Classic implementations of the Generic Districting Problem (6) have been recently adapted for the purposes of addressing segregation in school districts (1). This adaptation has revealed relationships between key constraints in the optimization framework, particularly an inverse relationship between student segregation in the district and the travel time for students to get from their neighborhoods to their schools. Because this relationship has only been demonstrated in a single school district in Riverside, California, this project aims to investigate this dynamic across every school district in California. Through the parameterization of optimization models, new variables are created. Another angle of this research is to leverage these new variables in econometric analysis with other socioeconomic variables that are of interest to education segregation scholars.

Comparative Neighborhood Fragmentation

Link to project repository

This project is a comparative analysis between two methodologies for neighborhood delineation based on census areal units. The first method involves the commonly applied K-means algorithm to regionalize units based on common attributes into typologies. We borrow an expanded version of this method from recent (2, 3) works and compare it to a spatially-explicit neighborhood regionalization method (4) in the context of neighborhood fragmentation. We use census data to compare these two methods in 50 metropolitan areas in the United States over a 30 year period.

Why this project?

This project provides tangible benefits to my growth as a scholar.

Firstly, it elevates my proficiency with python. Through my work in graduate school, I have evolved from someone who had a casual interest in computers to an intermediate data scientist. My work has gradually required more elaborate python code, from simple data processing scripts to multi-step analytics which leverages custom built python modules. However, because I have had little formal training in software development, being able to directly interface with the project mentor(s) on a new feature in an established package will teach me best practices of collaborative python development and code structure

This project will also deepen my understanding of optimization applications through developing code that aids in the parameterization and solving of these models. Whereas my previous work has leveraged Gurobi, a commercial solver made available to me on a student license, this project will allow me to see the code (5) behind the solver, allowing me to better understand the often opaque nature of optimization under the guidance of advanced developers.

Appendix

GSOC Project References

- **NumFOCUS GSOC Page**
  - https://summerofcode.withgoogle.com/programs/2024/organizations/numfocus
- **GSOC Student Blog**
  - https://fiendskrah.github.io/gsoc-2025-blog/
- **Guinness Python Example**
  - https://gist.github.com/ljwolf/e5927ab8c859ed477f496329c1ce19fc#file-guinness-py-L19

- **Issue #464 (PySAL/spopt)**
  - https://github.com/pysal/spopt/issues/464
- **Pull Request #465 (PySAL/spopt)**
  - https://github.com/pysal/spopt/pull/465
- **spopt.route Module**
  - https://github.com/ljwolf/spopt/tree/main/spopt/route

Other Packages

- **Nanobind** https://github.com/wjakob/nanobind
- **Open Source Routing Machine (OSRM)** https://project-osrm.org/
- **OpenStreetMap (OSM)** https://www.openstreetmap.org/
- **py-osrm** https://github.com/nilsnolde/py-osrm?tab=readme-ov-file
- **routingpy** https://github.com/nilsnolde/routingpy
- **Pandana** https://udst.github.io/pandana/

Optimization

- **Introduction to Vehicle Routing Problems (pyvrp)**
  - https://pyvrp.org/setup/introduction_to_vrp.html
- **Branch and Bound Algorithm**
  - https://en.wikipedia.org/wiki/Branch_and_bound

Student Project Repositories

- **School District Optimization for Exploratory Spatial Data Analysis**
  - https://github.com/fiendskrah/seg-opt-district-bounds-2024
- **Comparative Neighborhood Fragmentation**
  - https://github.com/fiendskrah/contig-frag

Academic References

- **[1] Wei, R. et al. (2022)** ***"Reducing Racial Segregation of Public School Districts"*** https://www.sciencedirect.com/science/article/pii/S0038012122002166
- **[2] Delmelle, E. C. (2019)** ***"The Increasing Sociospatial Fragmentation of Urban America"*** https://www.mdpi.com/2413-8851/3/1/9
- **[3] Delmelle, E. C. (2015)** ***"Mapping the DNA of Urban Neighborhoods: Clustering Longitudinal Seque*** https://www.tandfonline.com/doi/abs/10.1080/00045608.2015.10961
- **[4] Rey, S. J. et al. (2011)** ***"Measuring Spatial Dynamics in Metropolitan Areas"*** https://journals.sagepub.com/doi/10.1177/0891242410383414
- **[5] Rey, S. J. (2009)** ***"Show Me The Code"*** https://link.springer.com/article/10.1007/s10109-009-0086-8
- **[6] Shoepfle, O. B. and Church, R. (1991)** ***"A New Network Representation of a 'Classic' School Distr*** https://www.sciencedirect.com/science/article/pii/003801219190017L