The Journal of Open Source Software

spopt: a python package for solving spatial optimization problems in PySAL

Xin Feng[1, 2], **Germano Barcelos**[3], **James D. Gaboardi**[4, 5], **Elijah Knaap**[1], **Ran Wei**[1], **Levi Wolf**[6], **Qunshan Zhao**[7], **and Sergio Rey**[1]

1 Center for Geospatial Sciences, University of California Riverside **2** Department of Geography and Environmental Sustainability, University of Oklahoma **3** Federal University of Viçosa **4** Oak Ridge National Laboratory **5** The Peter R. Gould Center for Geography Education and Outreach, Penn State **6** University of Bristol **7** Urban Big Data Centre, School of Social & Political Sciences, University of Glasgow

DOI: 10.21105/joss.03330

Software

- Review
- Repository
- Archive

Editor: Vissarion Fisikopoulos

Reviewers:

- @ArikaLZ
- @tmickleydoyle

Submitted: 21 May 2021
Published: 19 November 2021

Summary

Spatial optimization is a major spatial analytical tool in management and planning, the significance of which cannot be overstated. Spatial optimization models play an important role in designing and managing effective and efficient service systems such as transportation, education, public health, environmental protection, and commercial investment among others. To this end, spopt (**sp**atial **opt**imization) is under active development for the inclusion of newly proposed models and methods for regionalization, facility location, and transportation-oriented solutions. Spopt is a submodule in the open-source spatial analysis library PySAL (Python Spatial Analysis Library) founded by Dr. Serge Rey and Dr. Luc Anselin in 2005 (S. Rey & Anselin, 2007; Sergio J. Rey et al., 2015; Sergio J. Rey et al., 2021). The goal of developing spopt is to provide management and decision-making support to all relevant practitioners and to further promote the appropriate and meaningful application of spatial optimization models in practice.

Statement of need

Spatial optimization methods/algorithms can be accessed in many ways. ArcGIS (https://www.esri.com/en-us/home) and TransCAD (https://www.caliper.com/) are two well-known commercial GIS software packages that provide modules designed for structuring and solving spatial optimization problems. The optimization functions they offer focus on a set classical single facility location methods (e.g., Weber, Median, Centroid, 1-center), routing and shortest path methods (e.g., shortest path on the network, least cost path over the terrain), and multi-facility location-allocation methods (e.g., coverage models, p-median problem). They are user-friendly and visually appealing, but the cost is relatively high (Murray, 2021).

Open-source software is another option to access spatial optimization. Although it may require users to have a certain level of programming experience, open-source software provides relatively novel and comprehensive methods, and more importantly, it is free and can be easily replicated. This is particularly true for regionalization and facility-location methods. Regionalization methods are limited in commercial GIS software, and may only have grouping analysis for vector data and region identification for raster data. On the contrary, there are many application-oriented open-source packages that facilitate the implementation of regionalization methods in various fields, including climate (e.g., HiClimR (https://cran.r-project.org/web/packages/HiClimR/index.html), synoptReg (https://cran.r-project.org/web/packages/synoptReg/index.html)), biography (e.g.,

Phyloregion (https://cran.r-project.org/web/packages/phyloregion/index.html), regioneR (http://bioconductor.org/packages/release/bioc/html/regioneR.html)), hydrology (e.g., nsRFA(https://cran.r-project.org/web/packages/nsRFA/index.html)), agricultural (e.g., OpenLCA (https://www.openlca.org/)), and so on. The functions of graph regionalization with clustering and partitioning have been provided by several packages such as Rgeoda, maxcut: Max-Cut Problem, RBGL: R Boost Graph Library, and grPartition. They are probably the most closely related projects to the regionalization section of spopt, however, they are written in R and MATLAB. For facility-location methods, commercial software such as TransCAD and ArcGIS implements models using a heuristic approach. However, they don't provide details about the solution found, which limits the interpretability of the results (Chen et al., 2021). On the other hand, existing open-source packages mostly aim at solving coverage problems such as PySpatialOpt (https://github.com/apulverizer/pyspatialopt), Allagash (https://apulverizer.github.io/allagash/) and maxcovr (https://github.com/njtierney/maxcovr), but the available models, solvers, and overall accessibility vary significantly. Therefore, it is necessary to develop an open-source optimization package written in Python that includes various types of classic facility-location methods with a wide range of supported optimization solvers.

Current functionality

Originating from the region module in PySAL, spopt is under active development for the inclusion of newly proposed models and methods for regionalization and facility location. Regarding regionalization, six models are developed for aggregating a large set of geographic units (with small footprints) into a smaller number of regions (with large footprints). They are:

1. Max-p-regions: the clustering of a set of geographic areas into the maximum number of homogeneous and spatially contiguous regions such that the value of a spatially extensive regional attribute is above a predefined threshold (Duque et al., 2012; Wei et al., 2020).
2. Spatially-encouraged spectral clustering (spenc): an algorithm to balance spatial and feature coherence using kernel combination in spectral clustering (Wolf, n.d.).
3. Region-K-means: K-means clustering for regions with the constraint that each cluster forms a spatially connected component.
4. Automatic Zoning Procedure (AZP): the aggregation of data for a larger number of zones into a prespecified smaller number of regions based on a predefined type of objective function (Openshaw, 1977; Openshaw & Rao, 1995).
5. Skater: a constrained spatial regionalization algorithm based on spanning tree pruning. Specifically, the number of edges is prespecified to be cut in a continuous tree to group spatial units into contiguous regions (Assunção et al., 2006).
6. WardSpatial: an agglomerative clustering (each observation starts in its own cluster, and pairs of clusters are chosen to merge at each step) using ward linkage (the goal is to minimize the variance of the clusters) with a spatial connectivity constraint (sklearn.cluster.AgglomerativeClustering).

Take the functionality of Max-p-regions as an example. Other methods can be applied in a similar process, including importing the needed packages, imputing and reading data, defining the parameters, solving the model, and plotting the solution.

```
from spopt.region import MaxPHeuristic as MaxP
import geopandas, libpysal
# Read in the data on regional incomes for Mexican states.
```

```
mexico = geopandas.read_file(libpysal.examples.get_path("mexicojoin.shp"))
# Specify parameters for the Max-p-regions model.
# Details can be found at https://pysal.org/spopt/notebooks/maxp.html.
attrs_name = [f"PCGDP{2000}"]
w = libpysal.weights.Queen.from_dataframe(mexico)
threshold_name, threshold, top_n, mexico["count"] = "count", 6, 2, 1
# Solve the Max-p-regions model.
model = MaxP(mexico, w, attrs_name, threshold_name, threshold, top_n)
model.solve()
# Plot the model solution.
mexico["maxp_new"] = model.labels_
mexico.plot(column="maxp_new", categorical=True, edgecolor="w");
```

The corresponding solution of Max-p-regions running the above code is shown in Figure 1. It results in five regions, three of which have six states, and two with seven states each. Each region is a spatially connected component, as required by the Max-p-regions problem.

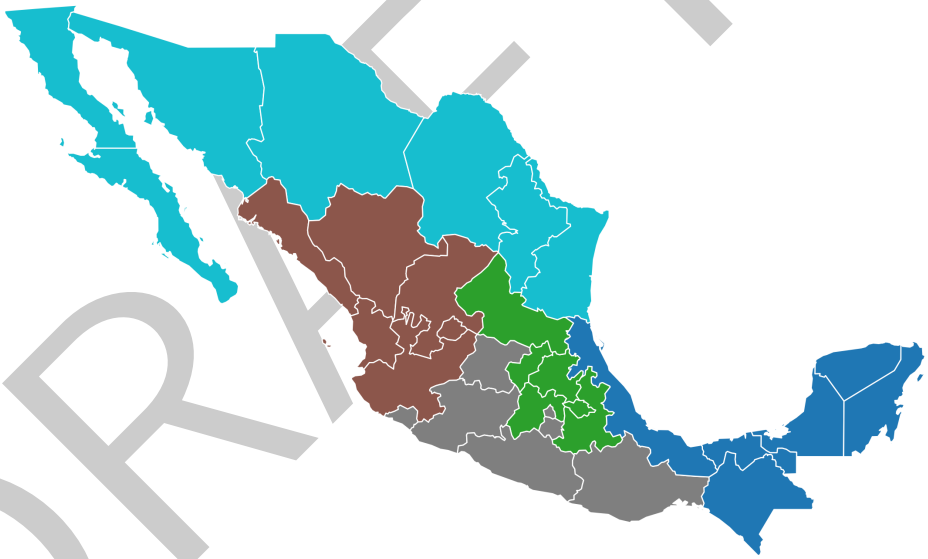

Figure 1: The solution of Max-p-regions when 32 Mexican states are clustered into the maximum number of regions such that each region has at least 6 states and homogeneity in per capita gross domestic product in 2000 is maximized.

For facility-location, four models, including two coverage models and two location-allocation models based on median and center problems, are developed using an exact approach.

1. Location Set Covering Problem (LSCP): Finding the minimum number of facilities and their locations such that all demands are covered within the maximal distance or time standard (Toregas et al., 1971).
2. Maximal Covering Location Problem (MCLP): Locating a prespecified number of facilities such that demand coverage within a maximal service distance or time is maximized (Church & ReVelle, 1974).
3. P-Median Problem: Locating p facilities and allocating the demand served by these facilities so that the total weighted assignment distance or time is minimized (ReVelle & Swain, 1970).

4. P-Center Problem: Locating p facilities and allocating the demand served by these facilities to minimize the maximum assignment distance or time between demands and their allocated facilities (Hakimi, 1964).

For example, Maximal Covering Location Model functionality is used to select 4 out of 16 store sites in the San Francisco area to maximize demand coverage, as shown in Figure 2. Other facility-location methods can be applied in a similar way.

```
from spopt.locate.coverage import MCLP
import geopandas, numpy, pandas, pulp
# Read in the datasets
ntw_dist = pandas.read_csv("SF_network_distance_candidateStore_16_censusTract_205_
demand_points = pandas.read_csv("SF_demand_205_centroid_uniform_weight.csv", index
facility_points = pandas.read_csv("SF_store_site_16_longlat.csv", index_col=0)
study_area = geopandas.read_file("ServiceAreas_4.shp").dissolve()
# Create a store site to tract centroid distance matrix
ntw_piv = ntw_dist.pivot_table(values="distance", index="DestinationName", columns
cost_matrix, ai, p = ntw_piv.to_numpy(), demand_points["POP2000"].to_numpy(), 4
mclp = MCLP.from_cost_matrix(cost_matrix, ai, max_coverage=5000, p_facilities=p)
mclp = mclp.solve(pulp.GLPK(msg=False))
# Build a facility-demand array for demand covered by each facility
mclp.facility_client_array()
fgeom = geopandas.points_from_xy(facility_points.long, facility_points.lat)
facility_points_gdf = geopandas.GeoDataFrame(
    facility_points, geometry=fgeom,
).sort_values(by=["NAME"]).reset_index()
dgeom = geopandas.points_from_xy(demand_points.long, demand_points.lat)
demand_points_gdf = geopandas.GeoDataFrame(
    demand_points, geometry=dgeom,
).sort_values(by=["NAME"]).reset_index()
# plot results
n_facilities, title = facility_points_gdf.shape[0], f"MCLP ($p$={p})"
#plot_results(mclp, facility_points_gdf, demand_points_gdf, n_facilities, title)
```

The Journal of Open Source Software

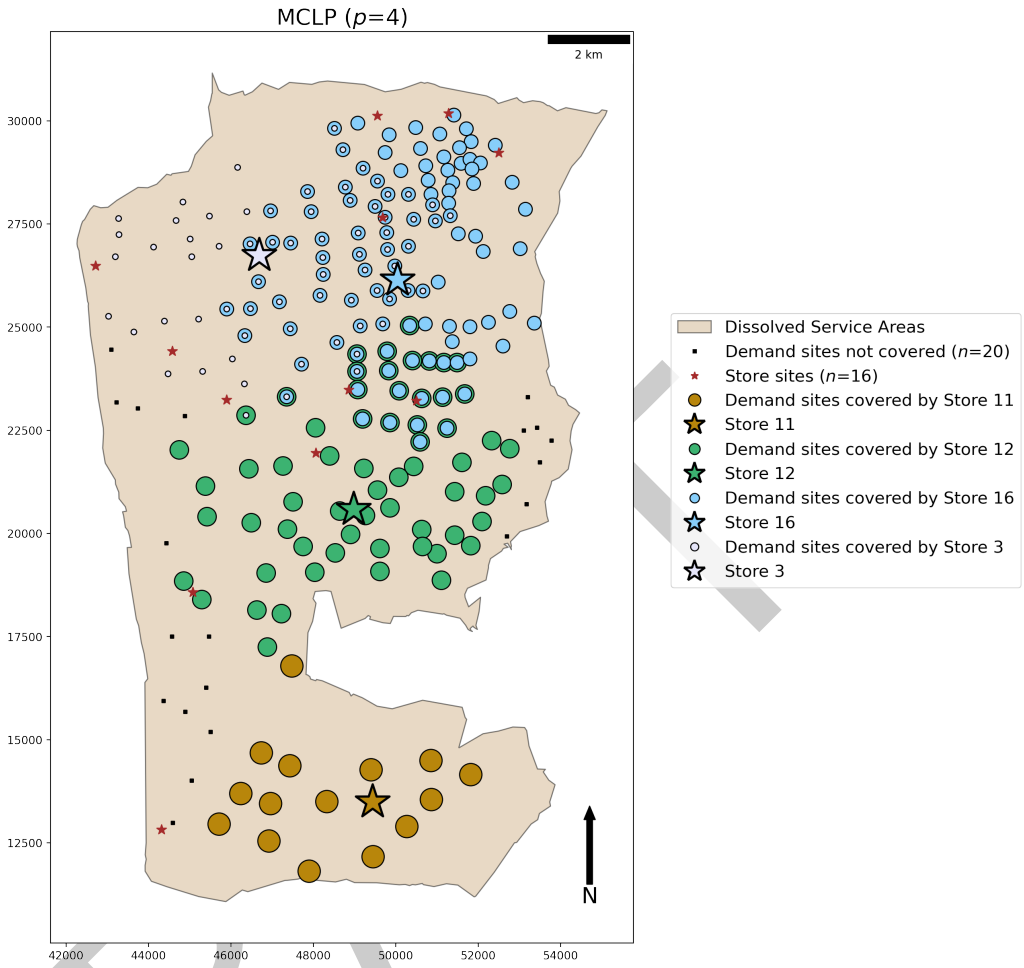

Figure 2: The solution of MCLP while siting 4 facilities using 5 kilometers as the maximum service distance between facilities and demand locations. See the "Real World Facility Location" tutorial (https://pysal.org/spopt/notebooks/facloc-real-world.html) for more details.

Planned Enhancements

Spopt is under active development and the spopt developers look forward to your extensive attention and participation. In the near future, there are three major enhancements we plan to pursue for spopt:

1. The first stream will be on the enhancement of regionalization algorithms by including several novel extensions of the classical regionalization models, such as the integration of spatial data uncertainty and the shape of identified regions in the max-p-regions problem.
2. The second direction involves adding capacity constraints and includes a polygon partial coverage on facility location models. No commercial and open-source software has provided these features before.
3. We anticipate adding functionality for solving traditional routing and transportation-oriented optimization problems. Initially, this will come in the form of integer programming formulations of the Travelling Salesperson Problem (Miller et al., 1960) and the Transportation Problem (Koopmans, 1949).

Acknowledgements

We would like to thank all the contributors to this package. Besides, we would like to extend our gratitude to all the users for inspiring and questioning this package to make it better. Spopt development was partially supported by National Science Foundation Award #1831615 RIDIR: Scalable Geospatial Analytics for Social Science Research.

The following acknowledgement applies to James D. Gaboardi (affiliation 4) *only*:

> This manuscript has been authored by UT-Battelle LLC under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

Assunção, R. M., Neves, M. C., Câmara, G., & Costa Freitas, C. da. (2006). Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees. *International Journal of Geographical Information Science*, *20*(7), 797–811. https://doi.org/10.1080/13658810600665111

Church, R. L., & ReVelle, C. S. (1974). The Maximal Covering Location Problem. *Papers in Regional Science Association*, *32*, 101–118. https://doi.org/10.1111/j.1435-5597.1974.tb00902.x

Duque, J. C., Anselin, L., & Rey, S. J. (2012). THE MAX-p-REGIONS PROBLEM*. *Journal of Regional Science*, *52*(3), 397–419. https://doi.org/10.1111/j.1467-9787.2011.00743.x

Hakimi, S. L. (1964). Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph. *Operations Research*, *12*(3), 450–459. https://doi.org/10.1287/opre.12.3.450

Koopmans, T. C. (1949). Optimum utilization of the transportation system. *Econometrica: Journal of the Econometric Society*, 136–146. https://doi.org/10.2307/1907301

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, *7*(4), 326–329. https://doi.org/10.1145/321043.321046

Murray, A. T. (2021). Contemporary optimization application through geographic information systems. *Omega*, *99*, 102176. https://doi.org/10.1016/j.omega.2019.102176

Openshaw, S. (1977). A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Transactions of the Institute of British Geographers*, 459–472. https://doi.org/10.2307/622300

Openshaw, S., & Rao, L. (1995). Algorithms for reengineering 1991 census geography. *Environment and Planning A*, *27*(3), 425–446. https://doi.org/10.1068/a270425

ReVelle, C. S., & Swain, R. W. (1970). Central Facilities Location. *Geographical Analysis*, *2*(1), 30–42.

Rey, S., & Anselin, L. (2007). PySAL: A Python Library of Spatial Analytical Methods. *The Review of Regional Studies*, *37*(1), 5–27. https://rrs.scholasticahq.com/article/8285.pdf

Rey, Sergio J., Anselin, L., Amaral, P., Arribas-Bel, D., Cortes, R. X., Gaboardi, J. D., Kang, W., Knaap, E., Li, Z., Lumnitz, S., Oshan, T. M., Shao, H., & Wolf, L. J. (2021). The PySAL Ecosystem: Philosophy and Implementation. *Geographical Analysis*. https://doi.org/10.1111/gean.12276

Rey, Sergio J., Anselin, L., Li, X., Pahle, R., Laura, J., Li, W., & Koschinsky, J. (2015). Open geospatial analytics with PySAL. *ISPRS International Journal of Geo-Information*, *4*(2), 815–836. https://doi.org/10.3390/ijgi4020815

Toregas, C., Swain, R., ReVelle, C. S., & Bergman, L. (1971). The Location of Emergency Service Facilities. *Operations Research*, *19*(6), 1363–1373. https://doi.org/10.1287/opre.19.6.1363

Wei, R., Rey, S., & Knaap, E. (2020). Efficient regionalization for spatially explicit neighborhood delineation. *International Journal of Geographical Information Science*, 1–17. https://doi.org/10.1080/13658816.2020.1759806

Wolf, L. (n.d.). *Spatially-encouraged spectral clustering: A technique for blending map typologies and regionalization*. https://doi.org/10.31219/osf.io/yzt2p