УДК 681.58
DOI 10.63115/5060.2025.33.88.011

ПРИКЛАДНАЯ ИССЛЕДОВАТЕЛЬСКАЯ СТАТЬЯ

Оценка соседних альтернатив в моделях принятия решений средствами языка программирования Python

Доманов Алексей Олегович

научный сотрудник Отдела исследований европейской интеграции, Институт Европы РАН, Москва, Россия

domanov.aleksey@gmail.com, https://orcid.org/0000-0002-6253-2067

АННОТАЦИЯ

- Функционал языка программирования Python позволяет хранить и обрабатывать сведения о соседних географических объектах, сопоставляемых политическими акторами. Наиболее подходящими библиотеками последних лет с открытым кодом для анализа и прогнозирования перемещений по координатной плоскости или графу смежности (например, в задачах рационального выбора, диффузии инноваций и формирования институтов, переезда в уже сформированную институциональную среду) можно считать Helipad и Mesa.
- «Цифровые двойники» акторов обращаются к сведениям о соседстве (закодированным в явном виде или выведенным из координат территорий) при взаимодействии со средой: в ходе поэтапной симуляции перечень альтернатив для размещения в следующий момент времени ограничивается списком близлежащих мест.
- Выбирая оптимальное местоположение, компьютерные модели сопоставляют релевантные характеристики соседних объектов, ориентируясь на закодированные предпочтения. Для этого проводится автоматический расчёт полезности, которую актор получил бы через некоторое время благодаря свойствам выбранной зоны, и определяется наиболее приемлемая близкая альтернатива.
- Некоторые библиотеки содержат готовые процедуры, предназначенные для визуализации участков земли и образованного ими пространства (в виде тепловой карты — в зависимости от значения какого-либо параметра в различных местах)
- Переформулировав задачу исследования (перенеся акцент с характеристик территорий на свойства перемещений в различных направлениях), можно воспользоваться библиотеками на основе теории игр. Сравнение стратегий перехода на соседние участки земли, алгоритмизируется с помощью функций библиотек *Axelrod, QuantEcon, StratPy, NashPy, OpenSpiel*.

КЛЮЧЕВЫЕ СЛОВА

агентно-ориентированное моделирование, матрица смежности, теория игр, методы оптимизации, предпочтения

ГЕОТЕГИ

Россия, Австрия, ЕС

ДОПОЛНИТЕЛЬНЫЕ МАТЕРИАЛЫ

doi.org/10.7910/DVN/UHJQQR

UDC 681.58
DOI 10.63115/5060.2025.33.88.011

CASE ARTICLE

Evaluating Contiguous Alternatives in Decision-Making Models Using Python Programming Language

Aleksey Domanov
Research Fellow, Department of European Integration Research, RAS Institute of Europe, Moscow, Russia
domanov.aleksey@gmail.com, https://orcid.org/0000-0002-6253-2067

ABSTRACT

- Python programming language allows for storing and processing information about neighboring geographic objects compared by political actors. The most suitable open-source recent libraries for analyzing and predicting movements along a coordinate plane or adjacency graph (for example, in rational choice studies, diffusion of innovations and shaping institutions or relocation to an already reshaped institutional environment) are presumably Helipad and Mesa.
- Actors' «digital twins» access information about their neighborhoods (encoded explicitly or derived from a polygon's coordinates) when interacting with the environment: by limiting the number of alternatives for displacement or allocation at the next moment (during a step-by-step simulation) to a list of nearby places.
- When choosing the optimal location, computer models compare the relevant characteristics of neighboring objects, focusing on encoded preferences. To that end an actor automatically calculates the utility it would have received by the end of the period thanks to the envisaged zone's properties, and the most acceptable nearby alternative is determined.
- Some libraries contain ready-made procedures visualizing the territories and the space they form (as a heat map — with areas' colors depending on the value of any parameter).
- By reformulating the research task (by shifting the emphasis from the territories' characteristics to the properties of moving in various directions), one could use libraries based on game theory. Algorithms to compare strategies (relocating to neighboring areas) are provided in libraries *Axelrod, QuantEcon, StratPy, NashPy, OpenSpiel*.

KEYWORDS
agent-based modeling, contiguity matrix, game theory, optimization methods, preferences

GEOTAGS
Russia, Austria, the EU

SUPPLEMENTARY MATERIALS
doi.org/10.7910/DVN/UHJQQR

В ходе исследований политических факторов человеческой деятельности (особенно, ограниченной рациональности, формирования институтов силами отдельных акторов, диффузии инноваций за границу государства) нередко возникает потребность формализовать учёт информации о соседстве территорий. Такие данные позволяют избавиться от избыточной сложности модели, сократив перечень взаимосвязей (в частности, по критерию соседства: если автор полагает, что на решение действующих лиц наибольшее воздействие оказывают близкие

объекты, то он может сократить выборку влияющих переменных, отсекая дальние зоны). Например, при планировании тиража местного общественно-политического издания, которое лишено технической возможности доставлять печатную продукцию на дальние расстояния, можно учитывать спрос лишь потенциальных подписчиков — в регионах-соседях 1 порядка.

Язык программирования Python предоставляет возможность структурированно хранить значения необходимых переменных и оперировать ими, тем самым автоматизируя многие аналитические процедуры (прежде всего, валидацию построенных моделей). Рассмотрим несколько библиотек с открытым кодом на этом языке, позволяющие проверить гипотезы о закономерностях выбора между некоторыми близлежащими точками или зонами, в которые политические акторы хотели бы перейти или направить свои ресурсы.

Хранить информацию о соседствующих территориях, которые рассматриваются акторами в качестве альтернативных вариантов для перемещения, целесообразно, например, с помощью классов[28] *Patch* и *PatchesGeo* библиотеки Helipad[29]. Она позволяет проводить над этими структурами данных те операции, с помощью которых часто моделируют политических акторов, выбирающих местоположение для своего имущества или самих себя (объектов класса *Agent*)[30]. Архитектура программного пакета допускает, что решение о территории будущего размещения принимается по итогам оценки соседних территорий в настоящий момент (например, в Приложении закодирован выбор наиболее благоприятной институциональной среды руководителями португальских предприятий).

Эта библиотека предлагает инструменты, с помощью которых пользователь может ограничить количество вариантов будущих перемещений кратким перечнем соседних участков земли, причём установить различия между наборами альтернатив для разных положений актора. С этой целью на множестве территорий задаётся пространственная структура: функция *heli.spatial()* создаёт словарные объекты *Edges* для каждой территории[31], *model.agents.createNetwork()* — единый граф для нескольких точек[32]. Благодаря этим хранилищам информации о смежно-

28 Язык Python создавался для написания кода, прежде всего, в объектно-ориентированной парадигме программирования, поэтому упомянутые сведения представлены в компьютерной памяти в виде объектов (отдельных экземпляров общего класса) с атрибутами.

29 Harwick C. (2023), Helipad: A Framework for Agent-Based Modeling in Python // SSRN. 2023. DOI http://dx.doi.org/10.2139/ssrn.3870501 [Электронный ресурс]: URL: https://ssrn.com/abstract=3870501 (accessed: 09.01.2025).

30 The Agent Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/agent/ (accessed: 09.01.2025).

31 The Helipad.spatial Function // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/model/spatial/ (accessed: 09.01.2025); The Edges Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/edges/ (accessed: 09.01.2025).

32 The AgentsPlot Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/

сти существует возможность выявить участки рядом с местоположением акторов, обратившись к его атрибуту *neighbors*[33] (например, с помощью команды country1.neighbors для объекта *country1*).

С точки зрения проектирования ГИС примечательна возможность задавать не только матрицу смежности для сетевого анализа, но и подразумевать соседство в одном или двух измерениях: не только в цепи элементов, но и по абсциссе и ординате различных зон на плоскости, а также широте и долготе. Вывод о степени близости территорий делается автоматически, если координаты указываются в качестве первого аргумента метода *heli.patches.add()*[34]. Вызов этой процедуры добавляет к создаваемым объектам класса *Patch* географические характеристики. Они вводятся в оборот как простой список кортежей (из широты и долготы точек на границах полигонов) или объекты класса *Polygon* широкоиспользуемой библиотеки Shapely (которые, в свою очередь, могут быть прочитаны из сторонних файлов с расширением .geojson — например, при помощи популярной библиотеки Geopandas)[35]. Перед добавлением этих участков на карту требуется уточнить тип строящейся модели с помощью специального аргумента *heli.spatial(geometry='geo')*, причём пользователь может настроить другие значения этого параметра (например, *geometry='polar'* для полярных координат).

Сведения о смежности позволяют визуализировать все упомянутые элементы, в двумерном пространстве на дисплее. При наличии координат они размещаются на двух координатных осях, в отсутствие таковых граф соседства отображается вызовом метода *addPlot* у объектов класса *Charts*[36].

При создании пространственной модели у акторов появляются методы, которые тесно связаны с упомянутыми перечнями соседей — способы перемещаться по карте или сети *(.moveTo(patchID)*, который настраивается указанием смежного участка в скобках, *.moveUp* и прочие. В дальнейшем эти методы задействуются после ответа на запрос симулятора: при исполнении «декорированной функции» *modelStep*[37] отдаётся команда установить местоположение актора через определённый временной интервал, и сведения о соседстве привлекаются к вычислениям для выдвижения гипотез о том, куда выбирающий сможет добраться в конце

functions/agentsplot/#notes (accessed: 09.01.2025).

33 The Patch Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/patch/ (accessed: 09.01.2025).

34 Пример использования приведён в charwick/helipad // Github. 2024. [Электронный ресурс]: URL: https://github.com/charwick/helipad/blob/master/helipad/spatial.py#L252 (accessed: 09.01.2025).

35 The PatchesGeo Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/patchesgeo/#comment-547 (accessed: 09.01.2025).

36 The Charts Class // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/functions/charts/ (accessed: 09.01.2025); инструкция по применению: The AgentsPlot...

37 The modelStep Hook // Helipad Codex. 2024. [Электронный ресурс]: URL: https://helipad.dev/hooks/modelstep/ (accessed: 09.01.2025).

предстоящего периода. По умолчанию результатом такого запроса могут стать только близкие территории и пункт изначального размещения актора, поскольку при изначальных настройках за одну единицу времени он продвигается лишь на один шаг, а значит, компьютеру целесообразно оценивать только доступные для передвижения участки.

Учёт и визуализация факторов выбора между соседними территориями

Ответ на запрос о выборе актора, сгенерированный в ходе симуляции, неслучаен (а значит, поддаётся расчёту компьютером) благодаря записанным сведениям не только об окружении территорий, но и их свойствах. Так как модель допускает некоторую степень рациональности актора, можем предположить, что в процессе решения он ориентируется на информацию о каких-либо признаках близлежащих земель. Следовательно, результат динамического моделирования должен учитывать те входные данные о признаках, которые оценивает актор.

Значения, принимаемые в расчёт смоделированным актором при принятии решений, хранятся как атрибуты *stocks* для каждого созданного территориального образования. К конкретному признаку обращаются, заключая его название в квадратные скобки: например, величина независимой переменной для гипотезы о влиянии инвестиционного климата на патентование солнечных батарей доступна при вызове команды *heli.patches['Austria'].stocks['contracts_enforcement']*.

Характеристики участков отображаются (например, в виде тепловой карты) с помощью метода *.config()*[38], который применяется к единому изображению множества участков (объекта класса *AgentsPlot*). Благодаря этой особенности название столбца атрибутов, извлечённого из shape-файла, указывается не для каждой административно-территориальной единицы, а однократно в скобках приведённой команды: например, *plot_obj.config('attractiveness', 'inv_clim_val')*.

В библиотеке Helipad заложена возможность выразить предпочтения актора в отношении релевантных признаков территории. определить, каким образом учитывается релевантная информация. С помощью условного оператора внутри функции *agentStep*[39] пользователь определяет, каким образом учитываются сведения о соседях и по каким критериям она оценивается, а именно устанавливает соответствие между желательными проявлениями признака и возможными перемещениями. Если этот параметр измерен на интервальной или порядковой

38 The AgentsPlot ...

39 Пример использования для решения получать энергию на соседней территории: helipad/sample-models/grass.py // Github. 2024. [Электронный ресурс]: URL: https://github.com/charwick/helipad/blob/master/sample-models/grass.py#L44 (accessed: 09.01.2025).

шкале, то актору отдаётся команда стремиться к наибольшему или наименьшему значению (по формуле максимизации полезности); в других случаях исключается возможность перехода на участки с конкретными характеристиками (например, при законодательном требовании лицензировать производство определённого вида продукции).

Функционал, реализованный в Helipad (в частности, реакция на параметры окружения для выбора среди ограниченного круга альтернатив), доступен и пользователям программных пакетов более широкого профиля. Например, менее узкоспециализированная и зарекомендовавшая себя за 10 лет библиотека Mesa построена по принципу известной среды программирования NetLogo (написанной на языках Scala и Java), поэтому в ней тоже осуществим изложенный способ сократить количество вариантов в поле зрения актора (например, учитывать свойства участков лишь по соседству с ним). Граф смежности также визуализируется без прямого обращения к функциям отображения из сторонних модулей[40] (наподобие NetworkX, TopoNetX или Igraph). При этом запустить графический интерфейс немного сложнее, чем в Helipad (изображение появится на локальном сервере — в окне браузера); хотя остальная часть Mesa предусматривает более тонкую настройку и выглядит проще в использовании (алгоритм обработки окружающей информации прописывается напрямую в методе *step*, не прибегая к типу «декорированных функций» *hook*, который предназначен для разработки многопользовательских интернет-сервисов).

Инструментарий теории игр для моделирования выбора

Ограничить круг действий и принимать решения на основе характеристик территорий можно также в других библиотеках, содержащих схожие с упомянутыми структуры данных и функции. В частности, существует способ решать поставленные задачи с помощью теоретико-игровых пакетов[41], если переформулировать исследовательскую задачу и операционализацию.

40 Spaces // Mesa documentation. 2024. [Электронный ресурс]: URL: https://mesa.readthedocs.io/stable/apis/space.html#mesa.space.NetworkGrid.get_neighborhood (accessed: 09.01.2025).

41 Usage // StratPy 0.0.4 documentation. 2024. [Электронный ресурс]: URL: https://stratpy.ofstad.co/en/latest/Usage/index.html#decision-nodes (accessed: 09.01.2025); Create a Normal Form Game // NashPy 0.0.41 documentation. 2024. [Электронный ресурс]: URL: https://nashpy.readthedocs.io/en/stable/how-to/create-a-game.html (accessed: 09.01.2025); normal_form_game // QuantEcon 0.8.0 documentation. 2024. [Электронный ресурс]: URL: https://quanteconpy.readthedocs.io/en/latest/game_theory/normal_form_game.html#creating-a-player (accessed: 09.01.2025); Implement New Games // Axelrod 4.13.1 documentation. 2024. [Электронный ресурс]: URL: https://axelrod.readthedocs.io/en/stable/tutorials/implement_new_games/index.html (accessed: 09.01.2025); The code structure // OpenSpiel documentation. 2024. [Электронный ресурс]: URL: https://openspiel.readthedocs.io/en/latest/developer_guide.html#c-and-python-implementations (accessed: 09.01.2025).

Ситуация в центре внимания теории игр напоминает описанную — актор выбирает между несколькими вариантами действий, — но она моделируется с акцентом на его активности, а не на её результате. С этой точки зрения сопоставленными альтернативами можно считать не территории, а процесс перемещения между ними — шаги выбирающего. Этот подход открывает возможность применить теоретико-игровой инструментарий, благодаря которому некоторые библиотеки на языке Python сравнивают преимущества и недостатки различных действий (и, следовательно, их результаты).

С этой целью желательно отразить названия территорий в идентификаторах столбцов или строк матрицы игры: например, сопоставить стратегии «оплатить пошлину за заявку на патент в России» и «в Европейское патентное ведомство». Что касается оппонента, его стратегию в некоторых случаях корректно считать единственной и известной перед действием актора: смоделировать особый тип игр — «с природой» и последовательными ходами. Следовательно, изначальные свойства территорий в поле зрения актора представляются итогом «деятельности» соперника: допустим, «природа» заранее определила, какими значениями атрибутов наделяет каждую территорию (а значит, какой выигрыш получит переместившийся туда актор) и не стремится максимизировать свой выигрыш.

О сходстве теоретико-игровых инструментов с библиотеками Helipad и Mesa говорит успешное применение описанных функций для формализации различных состязаний, изученных классиками общественных наук. Благодаря рассмотренному функционалу реализованы «дилемма заключённого»[42] и её многократное повторение[43] в ходе чемпионата Р. Аксельрода [Axelrod, 1997], а также переезд жителей[44] в рамках модели сегрегации Т. Шеллинга [Zhang, 2011].

Если представить выбираемые действия ходами в последовательной «игре с природой», то спектр активности игрока можно сократить: ограничить количество доступных стратегий. Зная, какие территории находятся *по соседству* от участника, целесообразно запретить ему перемещаться в остальные места. Что касается соседних разрешённых зон, из стратегий перехода в них можно составить список доступных шагов — профиль действий. Перечень соседних территорий (следовательно, разрешённых шагов) извлекается из матрицы соседства, заданной пользователем. В результате для каждого местоположения актора остаются доступными действия, соответствующие лишь территориям с положитель-

42 kondylidou/manipulation_simulation // Github. 2024. [Электронный ресурс]: URL: https://github.com/kondylidou/manipulation_simulation/blob/main/pd_ext/agent.py#L87 (accessed: 09.01.2025).

43 charwick/helipad // Github. 2024. [Электронный ресурс]: URL: https://github.com/charwick/helipad/blob/master/sample-models/axelrod.py#L124 (accessed: 09.01.2025).

44 GeoSchelling Model (Points & Polygons) // Mesa-Geo 0.9.1 documentation. 2024. [Электронный ресурс]: URL: https://mesa-geo.readthedocs.io/stable/examples/geo_schelling_points.html#geoagent (accessed: 09.01.2025).

ными значениями взаимосвязей (только переходы по существующим рёбрам), а не нулевыми.

Выборка соседей вершины формируется на графах, созданных из географических данных в библиотеке PySAL: загрузкой shape-файла методом *from_shapefile* у объектов класса весов[45] и последующего поиска по загруженной матрице методом *to_adjlist* (у объектов того же типа или общего класса весов — *weights.W*); затем и функциями, наподобие Moran_Local того же пакета, можно выявить эффект соседства. В отсутствие координат поиск соседних узлов сети более трудоёмок, но осуществим с помощью бинарного оператора на графе, заданном в библиотеке «NetworkX» (соседи содержатся в атрибуте *.adj* каждого объекта класса *Graph*[46]) или независимо от дополнительных модулей: в форме списка парных кортежей или списка списков (специальной вложенной структурой данных языка Python).

Кроме указания соседства, теоретико-игровые пакеты напоминают упомянутые библиотеки возможностью автоматизировать выбор действия. Для этого актор получает рекомендацию перейти в какое-либо место, проведя поиск стратегии с наибольшим выигрышем.

* * *

На случай необходимости интегрировать матрицы смежности в политико-географические модели разработано множество инструментов компьютерной симуляции и анализа данных. Некоторые библиотеки на языке программирования Python позволяют структурированно хранить и разнообразно обрабатывать переменные, которые требуются для решения поставленных задач (например, рационального выбора и размещения ресурсов).

На примере библиотеки Helipad показан широкий функционал подобных пакетов (в частности, Mesa), способы смоделировать динамику взаимодействия политических акторов с окружением и учесть их стремление максимизировать полезность от перемещения на соседние территории. Процедуры автоматического выбора будущего местоположения можно ограничить с помощью информации о соседстве: запрашивать информацию лишь о прилежащих участках и допускать переход только на эти земли.

Поскольку факторы принятия решения могут моделироваться не только как

45 Например, у объектов weights.Queen: libpysal.weights.Queen // libpysal v.4.13.0 Manual. 2024. [Электронный ресурс]: URL: https://pysal.org/libpysal/generated/libpysal.weights.Queen.html#libpysal.weights.Queen.from_shapefile (accessed: 09.01.2025).

46 Graph.adj // NetworkX 3.4.2 documentation. 2024. [Электронный ресурс]: URL: https://networkx.org/documentation/stable/reference/classes/generated/networkx.Graph.adj.html#networkx.Graph.adj (accessed: 09.01.2025).

атрибуты объектов, но и как промежуточный результат «игры с природой», для решения поставленной задачи можно воспользоваться функциями автоматического выбора в библиотеках на основе теории игр — например, Axelrod, QuantEcon, StratPy, NashPy, OpenSpiel. Сведения о соседних территориях позволяют сократить количество допустимых стратегий для каждого местоположения и определить реакцию актора на их параметры.

СПИСОК ЛИТЕРАТУРЫ:

1. Axelrod R. (1997), The complexity of cooperation: Agent-based models of competition and collaboration, Princeton: Princeton University Press, 248 p.
2. Zhang J. (2011), Tipping and residential segregation: A unified schelling model, Journal of Regional Science, vol. 51, no. 1, pp. 167–193.

REFERENCES:

1. Axelrod R. (1997), The complexity of cooperation: Agent-based models of competition and collaboration, Princeton: Princeton University Press, 248 p.
2. Zhang J. (2011), Tipping and residential segregation: A unified schelling model, Journal of Regional Science, vol. 51, no. 1, pp. 167–193.

ПРИЛОЖЕНИЕ. Код программы на языке «Python» «Восприятие регуляторных режимов и налогового законодательства Испании и Португалии руководителями португальских предприятий»

APPENDIX. “Python” code “Spanish and Portuguese regulatory regimes and tax law, as perceived by Portuguese enterprises’ directors”

Комментарии автора начинаются с символа #

```
from helipad import Helipad

def buildModel():
   mod = Helipad() # в начале строки 2 или 4 пробела
   mod.name = ‘TSpaMo’ # ‘The Spatial Model’
   mod.stages = 1
   mod.agents.order = ‘random’
```

```
   # Решат переместить производство из Португалии в Испанию,
оценив свойство территории «удобство регулирования»:
   mod.goods.add('regulComf', 'red', 1) # значение признака
участка по умолчанию, улучшено для Испании в функции
patchStep

        @mod.hook # функции выбора местоположения и изменения
свойства страны (например, с наступлением нового года)
дописывается к одной из основных функций вызываемых (в конце)
командой запустить и визуализировать симуляцию
        def agentStep(agent, model, stage):
        for contig in agent.patch.neighbors:
             if contig.stocks['regulComf'] > agent.patch.
stocks['regulComf']:#.attr:
             agent.moveTo(contig)

   @mod.hook
        def patchStep(patch, model, stage):
             if patch.name == «Spain»: patch.
stocks['regulComf']=2

   mapPlot = mod.spatial(dim=(2,1), geometry='geo',
wrap=False, corners=False)
   mapPlot.config( {'patchProperty': 'good:regulComf'} )
        shp=[ [«Portugal», [ (0,0),(0,1),(1,0) ]] , [«Spain»,
[ (1,0),(0,1),(2,1),(2,0) ]] ]
             # или вызвать ГИС-пакет в явном виде, например
для поточечного построения: shapely.geometry.shape({type:
«Point», «coordinates»: (0,0)})
   for ctryAttrs in shp: mod.patches.add(shape=ctryAttrs[1],
name=ctryAttrs[0])

        return mod

sim = buildModel()
sim.launchCpanel() #либо sim.launchVisual()

# assert(sim.patches[«Portugal»].agentsOn==[]) # поместить
внутрь buildModel() при необходимости проверить, что в
список акторов на португальской территории пуст (что все
переместились в Испанию)
```