ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 8 за 2024 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

А.Д. Авраменко, В.А. Судаков

Создание синтетических графов для задачи коммивояжёра

Рекомендуемая форма библиографической ссылки: Авраменко А.Д., Судаков В.А. Создание синтетических графов для задачи коммивояжёра // Препринты ИПМ им. М.В.Келдыша. 2024. № 8. 16 с. https://doi.org/10.20948/prepr-2024-8
https://library.keldysh.ru/preprint.asp?id=2024-8

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Российской академии наук

А.Д. Авраменко, В.А. Судаков

Создание синтетических графов для задачи коммивояжёра

Москва — 2024

Авраменко А.Д., Судаков В.А.

Создание синтетических графов для задачи коммивояжёра

Исследование посвящено актуальной проблеме решения задачи коммивояжёра. Работа акцентирует внимание на разработке нейросетевых подходов к решению этой задачи, подчёркивая необходимость обучения моделей на достаточном объёме данных, получить который в современных условиях за разумное время не представляется возможным. В связи с этим авторы предлагают алгоритм для генерации случайного графа с заранее известным кратчайшим путём или путём, близким к оптимальному. В результате работы был создан алгоритм аугментации матриц смежности для задачи коммивояжёра с заранее известным маршрутом.

Ключевые слова: синтетические наборы данных, задача коммивояжёра, статистический критерий

Aleksandr Dmitrievich Avramenko, Vladimir Anatolyevich Sudakov

Creating Synthetic Graphs with Predefined Paths for the Traveling Salesman Problem

The study is dedicated to the pressing issue of solving the Traveling Salesman Problem (TSP). The paper focuses on the development of neural network approaches to solving this problem, emphasizing the need for training models on a sufficient volume of data, which is not feasible to obtain in a reasonable time under current conditions. Therefore, the authors propose an algorithm for generating a random graph with a known shortest path or a path close to the optimal one. As a result of the study, an algorithm for augmenting adjacency matrices for the TSP with a known route was created.

Key words: synthetic data sets, traveling salesman problem, statistical test

Оглавление

Введение

В настоящее время задача коммивояжёра в том или ином виде присутствует во многих логистических проблемах, из-за чего решение задачи не теряет актуальности. При больших размерностях задачи точное решение не может быть найдено за разумное время. В последнее время наблюдаются попытки создания нейросетевых подходов к решению задачи коммивояжёра. Однако перед использованием любой модели машинного обучения, в частности нейросетей, как понятно из названия, модель нужно обучить. Найти готовый размеченный набор данных достаточного объёма для задач данного вида — невыполнимая задача в текущих реалиях. Поэтому возникает задача генерации случайного графа для задачи коммивояжёра с заранее известным кратчайшим путём или с путём, близким к оптимальному.

В данной работе рассматривается асимметричная, замкнутая задача коммивояжёра с полносвязным графом и целочисленной матрицей смежности.

На текущий момент представлено немало работ, посвящённых применению машинного обучения к различным комбинаторным задачам, в частности к задаче коммивояжёра. Например, в работе [1] рассматривается задача коммивояжёра в рамках планирования инфраструктуры для спорта. Также в работах [2], [3], [4] рассматривается возможность применения машинного обучения к ассиметричной, замкнутой задаче коммивояжёра. Из этого можно сделать вывод, что актуальность проблемы не падает с 2020 года. Именно поэтому для повышения эффективности методов машинного обучения требуется алгоритм генерации размеченных наборов данных, что является целью данной работы.

Существуют различные подходы к созданию синтетических данных:

1. Генерация случайных данных. Это самый простой подход, при котором данные генерируются случайным образом. Однако в этом случае характеристики, взаимосвязи и статистические закономерности, содержащиеся в исходных данных, не сохраняются.
2. Синтетические данные, сгенерированные на основе правил. В этом случае данные генерируются на основе определённых правил, которые могут быть известны из бизнес-процессов, правил создания и законов распределения.
3. Синтетические данные, сгенерированные искусственным интеллектом (ИИ). В этом случае модель ИИ обучается на исходных данных и затем генерирует новые данные, которые имитируют исходные. Это позволяет создавать «двойников» синтетических данных, которых можно использовать так, как если бы это были исходные данные.

В данной работе первый и третий подходы не могут быть реализованы в связи с постановкой задачи. Первый подход предполагает отсутствие согласованности сгенерированных данных, а значит, нельзя обосновать оптимальность маршрута, который должен быть получен в процессе генерации.

Третий подход предполагает наличие набора исходных данных, достаточного для обучения модели ИИ. Получение такого набора и является целью данной работы. При этом непонятно, как именно получать оптимальный маршрут для сгенерированных таким образом данных.

Поэтому в данной работе будет использоваться второй подход к созданию синтетических данных. В работе [5] представлены методы получения синтетических данных для задачи коммивояжёра, приближенных к используемым на практике данным. Задача создания синтетических данных с заранее известным решением среди научных исследований в авторитетных изданиях не найдена, поэтому можно говорить о новизне данной постановки задачи.

Методы

Для задания графа в задаче коммивояжёра будем использовать исключительно матрицу смежности. Задачу, поставленную в данной работе, можно решать с помощью двух подходов: Первый подход можно разделить на такие этапы: генерация случайной матрицы малой размерности, нахождение её оптимального решения и наращивание размерности так, чтобы оптимальное решение менялось по известному закону. Такой подход оставляет неясными достаточно много моментов, главный из которых — как узнать закон изменения оптимального решения.

Второй подход заключается в генерации случайного пути, вокруг которого строится матрица смежности так, чтобы любой другой путь был заведомо больше предыдущего. Именно такой подход разработан в данной работе.

При рассмотрении методов решения задачи коммивояжёра особенно интересны два таких часто используемых метода, согласно [6], как генетический метод и метод ветвей и границ. Оба этих метода в конце решения задачи, помимо найденного решения, предоставляют набор менее оптимальных решений, близких к найденному. Поэтому и появилась идея взять алгоритм решения задачи коммивояжёра и пойти в обратную сторону, генерируя условие задачи из какого-то заранее известного оптимального решения.

При более детальном рассмотрении генетического метода становится понятно, что с каждой следующей итерацией количество ограничений на генерацию будет только возрастать, что ясно из математической модели генетического алгоритма, представленного в работе [7], так как каждая новая особь будет ограничиваться предыдущими шагами независимо от того, как именно особи будут появляться из-за случайностей в популяции. Например, один и тот же маршрут может быть получен из-за разных скрещиваний, что накладывает дополнительные ограничения на согласованность маршрутов особей. Поэтому данный случайный алгоритм решения существенно проигрывает методу ветвей и границ.

В большинстве учебных материалов, рассматривающих задачу коммивояжёра, например в работах [6] и [7], в качестве учебных примеров

используются матрицы смежности, похожие на матрицы, полученные из нормального распределения. Поэтому в данной работе будет создан алгоритм, результатом работы которого будут матрицы смежности, близкие к матрицам, полученным из нормального распределения. Также в матрицах с равномерно распределенными значениями сложнее выделить признаки, которые могли бы помочь моделям машинного обучения переобучиться. Метод ветвей и границ, используемый для поиска оптимальных путей в графах малой размерности, описан в работе [8].

Перед описанием непосредственно алгоритма создания случайных графов введём обозначения:

N — количество городов;

d_{ij} — дуга из города i в город j;

r_i — i-й город маршрута, вокруг которого строится матрица смежности, где $i \in 1..N$;

$M = \left(d_{r_1 r_2}, d_{r_2 r_3}, \ldots, d_{r_N r_1}\right)$ — маршрут;

c_{ij} — стоимость перемещения по дуге d_{ij};

B_n — целочисленная случайная величина $c_{r_n r_{n+1}}$ на n-м шаге алгоритма;

$V_{ij} \sim f(v|B_n, n)$ — случайная величина c_{ij}, распределенная по закону $f(v|B_n, n)$, где $i, j \neq r_n$, на n-м шаге алгоритма при условии, что $c_{r_n r_{n+1}} = B_n$. Для случая $i = jV_{ij} = \infty$ независимо от распределения;

$p_B(B_n)$ и $p_{V_{ij}}(v|B_n, n)$ — функция плотности распределения для случайных величин B_n и V_{ij} соответственно. Эти случайные величины используются с целью описать алгоритм в общем виде, варианты их уточнения будут рассмотрены ниже вместе с вычислительным экспериментом.

Общая концепция алгоритма генерации предполагает следующее: в начале алгоритма получим случайный маршрут M. Этот случайный маршрут разбивается на дуги $d_{r_i r_{i+1}}$, соответствующие клетке матрицы смежности в строке r_i и столбце r_{i+1} с ценой перемещения $c_{r_i r_{i+1}}$. Эти пары перемешиваются случайным образом. Далее пошагово наращивается матрица смежности. На первом шаге матрица размера $1x1$ вида таблица 1.

Таблица 1

Первый шаг алгоритма

	r_2
r_1	B_1

Далее в матрицу добавляются строка и столбец, тогда матрица приобретает вид, показанный в таблице 2. И так далее, пока матрица не примет размеры $N \times N$.

Таблица 2

Второй шаг алгоритма

	r_2	r_3
r_1	B_1	$V_{r_1 r_3} \sim f(v \mid B_n, n)$
r_2	$V_{r_2 r_2} = \infty$	B_2

Шаг алгоритма n состоит в следующем: берётся случайная величина B_n в соответствии с законом плотности распределения $p_B(B_n)$, в матрицу записывается $c_{r_n r_{n+1}} = B_n$, далее формируются два вектора длины $n-1$ на основе закона распределения $f(v|B_n, n)$, которые дополняют матрицу строкой и столбцом, на этом шаг заканчивается. На рисунке 1 показано, как производится третий шаг алгоритма.

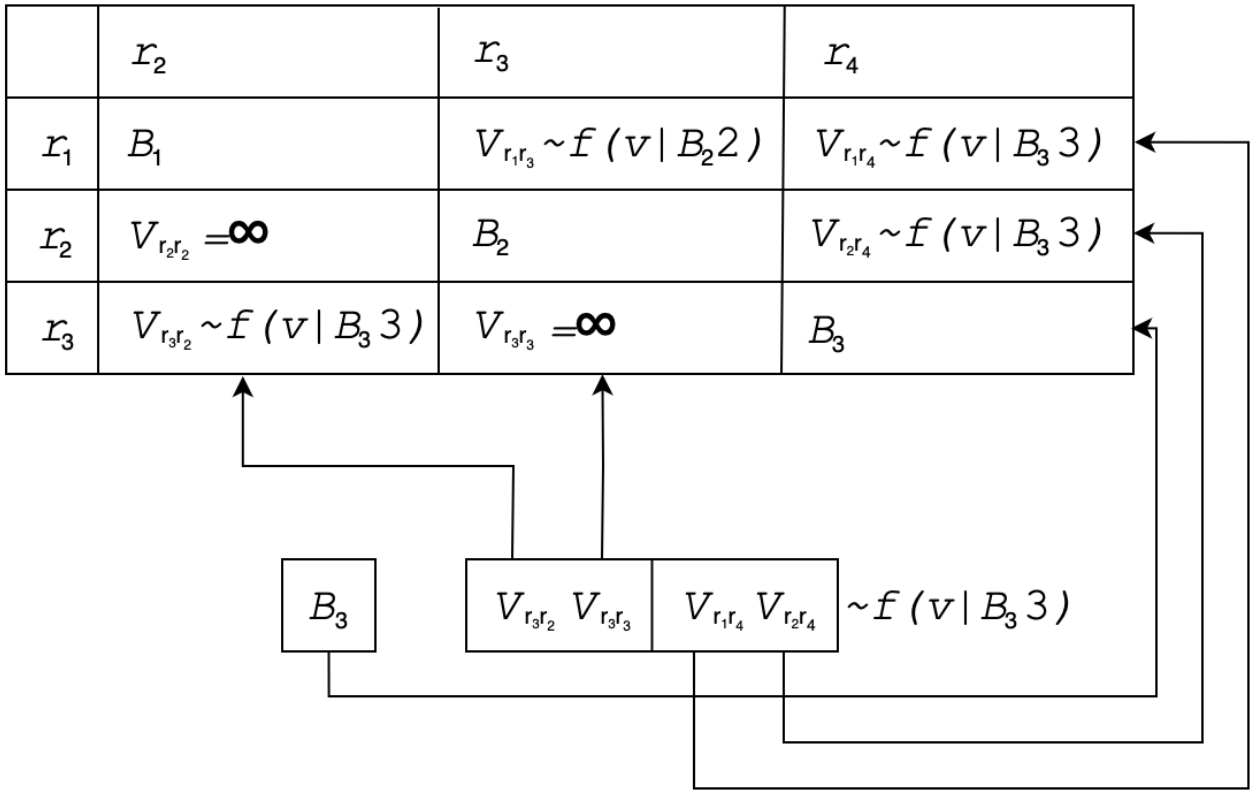

Рис. 1. Третий шаг алгоритма

Так, после на каждом шаге алгоритма размерность матрицы увеличивается на 1 и после N-го шага достигает требуемой размерности. Общий вид матрицы показан на таблице 3.

Таблица 3

Последний шаг алгоритма

	r_1	r_2	r_3	r_4	..	r_N
r_1	∞	B_1	$V_{r_1 r_3} \sim$ $\sim f(v \mid B_{2,} 2)$	$V_{r_1 r_4} \sim$ $\sim f(v \mid B_{3,} 3)$	..	$V_{r_1 r_N} \sim$ $\sim f(v \mid B_{N-1}, N-1)$
r_2	$V_{r_2 r_1} \sim$ $\sim f(v \mid B_N, N)$	∞	B_2	$V_{r_2 r_4} \sim$ $\sim f(v \mid B_{3,} 3)$	..	$V_{r_2 r_N} \sim$ $\sim f(v \mid B_{N-1}, N-1)$
r_3	$V_{r_3 r_1} \sim$ $\sim f(v \mid B_N, N)$	$V_{r_3 r_2} \sim$ $\sim f(v \mid B_{3,} 3)$	∞	B_3	..	$V_{r_3 r_N} \sim$ $\sim f(v \mid B_{N-1}, N-1)$
r_4	$V_{r_4 r_1} \sim$ $\sim f(v \mid B_N, N)$	$V_{r_4 r_2} \sim$ $\sim f(v \mid B_{4,} 4)$	$V_{r_4 r_3} \sim$ $\sim f(v \mid B_{4,} 4)$	∞	..	$V_{r_4 r_N} \sim$ $\sim f(v \mid B_{N-1}, N-1)$
..	..	..	..	..	..	..
r_N	B_N	$V_{r_N r_2} \sim$ $\sim f(v \mid B_N, N)$	$V_{r_N r_3} \sim$ $\sim f(v \mid B_N, N)$	$V_{r_N r_4} \sim$ $\sim f(v \mid B_N, N)$	..	∞

В таблице 4 показан пример работы алгоритма задачи коммивояжёра с четырьмя городами, где значения стоимостей перемещения не превышают 10. Так как распределения вероятностей для случайных величин $B(n)$ и $V(b_n, n, i, j)$ на данном этапе не уточняются, а будут уточняться далее, то конкретные значения взяты случайно с целью показать, как выстраиваются зависимости в процессе генерации матрицы смежности.

Таблица 4

Пример работы алгоритма для 4-х городов

	$r_2 = 3$	$r_3 = 4$	$r_4 = 2$	$r_1 = 1$
$r_1 = 1$	$B_1 = 7$	$2 \sim f_V(v \mid 1,2)$	$8 \sim f_V(v \mid 2,3)$	∞
$r_2 = 3$	∞	$B_2 = 1$	$3 \sim f_V(v \mid 2,3)$	$9 \sim f_V(v \mid 6,4)$
$r_3 = 4$	$6 \sim f_V(v \mid 2,3)$	∞	$B_3 = 2$	$5 \sim f_V(v \mid 6,4)$
$r_4 = 2$	$5 \sim f_V(v \mid 6,4)$	$2 \sim f_V(v \mid 6,4)$	∞	$B_4 = 6$

В начале работы алгоритма формируется маршрут, вокруг которого будет строиться матрица смежности. Маршрут для примера из таблицы 4 следующий: $1 \to 3 \to 4 \to 2$. На первом шаге реализуется случайная величина $B_1 = 7$, и заполняется матрица размером 1x1 для участка маршрута $1 \to 3$. На втором шаге алгоритма реализуется случайная величина $B_2 = 1$, т.к. уже имеется матрица 1x1, она дополняется до матрицы 2x2 строкой и столбцом, где в клетке $c_{r_2 r_3} = c_{34}$ соответствующей $3 \to 4$ ставится реализация $B_2 = 1$, а в $c_{r_2 r_2} = c_{33}$ и

$c_{r_1 r_3} = c_{14}$ ставятся реализации $V_{33,}V_{14} \sim f_V(v|1,2)$, где $V_{33} = \infty$. На третьем шаге матрица дополняется реализацией $B_3 = 2$ и реализациями $V_{43,}V_{44} = \infty, V_{12,}V_{32} \sim f_V(v|2,3)$. Четвёртый шаг аналогичен второму и третьему. После заполнения матрицы 4x4 следует переместить все элементы так, чтобы новая матрица соответствовала каноническому виду, у такой матрицы колонки и строки будут расположены в порядке: $r_1 = 1, r_4 = 2, r_2 = 3, r_3 = 4$.

Ниже приведён листинг 1 псевдокода алгоритма, где функции выделены жирным шрифтом, а переменные курсивом для удобства чтения. *Количество_вершин* соответствует значению N.

Листинг 1. Псевдокод алгоритма

```
Функция ГенерацияМатрицыСмежности(количество_вершин, B, V):
маршрут = ГенерацияСлучайногоМаршрута(количество_вершин)
матрица = ПустаяМатрица(количество_вершин, количество_вершин)
    для n от 0 до Длина(маршрут):
b = B(n)
массив1 = V(b, n, размер_массива = n-1)
массив2 = V(b, n, размер_массива = n-1)
УстановитьНазваниеСтроки(матрица, n, маршрут[n])
УстановитьНазваниеСтолбца(матрица, n, маршрут[n])
матрица[n][n] = b
        для j от 0 до n-1:
матрица[n][j] =  массив1[j]
матрица[j][n] =  массив2[j]
ПеремешатьМатрицу(матрица)
    вернуть матрица
```

В этом псевдокоде используются следующие вспомогательные функции:

- **ГенерацияСлучайногоМаршрута(*количество_вершин*)**: генерирует случайный маршрут, проходящий через все вершины графа. Возвращает массив перестановки вершин графа без повторений от 0 до *количество_вершин*-1.
- **ПустаяМатрица(*количество_строк*, *количество_столбцов*)**: возвращает матрицу, заполненную 0, размера [*количество_строк* x *количество_столбцов*].
- **УстановитьНазваниеСтроки(*матрица*, *номер_строки*, *название_строки*)**: устанавливает название *название_строки* для строки под номером *номер_строки.*
- **УстановитьНазваниеСтолбца(*матрица*, *номер_столбца*, *название_столбца*)**: устанавливает название *название_столбца* для столбца под номером *номер_столбца*.
- **ПеремешатьМатрицу(матрица)**: перемешивает строки и столбцы матрицы *матрица* так, чтобы индексы строк и столбцов шли в порядке возрастания.

Стоит отметить, что функции **B(*шаг_алгоритма*)** и **V(*стоимость_шага_маршрута*, *шаг_алгоритма*, *размер_массива*)** представляют собой некоторые случайные величины, зависящие от шага алгоритма и

реализации предыдущей величины. Точное поведение этих функций не указано в описании алгоритма, поэтому они представлены в псевдокоде как чёрные ящики.

Вычислительный эксперимент

Чтобы проверить работоспособность алгоритма, введём такие эмпирические метрики качества: процент сгенерированных матриц смежности, для которых верно, что образующий их маршрут действительно является кратчайшим, и проверки на принадлежность весов сгенерированной матрицы к равномерному распределению. Близость равномерному распределению позволяет говорить о том, что дальнейшая тренировка алгоритмов машинного обучения для поиска оптимального пути в графе не будет тривиальной.

Для проверки того, что веса сгенерированной матрицы принадлежат к равномерному распределению, возьмём матрицу, полученную из равномерного распределения, и сгенерированную матрицу. Для тестов используются U-критерий Манна-Уитни и критерий Колмогорова-Смирнова. В данной работе используются реализации вышеуказанных тестов из пакета scipy 1.4.1 для python3.7.3, а именно функции scipy.stats.mannwhitneyu и scipy.stats.ks_2samp соответственно.

U-критерий Манна-Уитни используется для сравнения двух выборок и проверки гипотезы о том, что обе выборки из одного и того же распределения.

$$U = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \quad (1)$$

Статистика U-критерия Манна-Уитни определяется как сумма рангов наблюдений одной из выборок, как показано в формуле 1, где R_1 — сумма рангов наблюдений в первой выборке, а n_1 и n_2 — размеры выборок.

Критерий Колмогорова-Смирнова также используется для сравнения двух эмпирических распределений или эмпирического распределения с теоретическим. Он основан на максимальном отклонении между двумя функциями распределения.

$$D_{nm} = \sup_x |F_n(x) - G_m(x)| \quad (2)$$

Статистика критерия Колмогорова-Смирнова определяется как максимальное абсолютное отклонение между двумя эмпирическими функциями распределения, согласно формуле 2, где $F_n(x)$ и $G_m(x)$ — эмпирические функции распределения двух выборок размером n и m соответственно.

Таким образом, использование этих критериев позволяет проверить, насколько близко веса сгенерированной матрицы к равномерному распределению.

Пусть случайные величины V_{ij} и B_n не зависят от шага. Тогда возьмём целочисленную равномерно распределенную величину $B_n \sim U[0, L)$ и $V_{ij} \sim U[B_n, L)$, где $L = 10$. Нижняя граница равномерного распределения равна 0

потому, что смещение всех значений матрицы смежности на какую-либо величину не изменит характера оптимального маршрута. Тогда полученная матрица смежности будет такая, что $0 \leq c_{ij} < 10 : \forall i, j \in 1..N$. Поэтому сравнивать её будем с матрицей, где $c_{ij} = X \sim U[0,10) : i, j \in 1..N$. В таблице 5 приведены эмпирические данные, полученные путём проверки, что маршрут, по которому генерируется матрица смежности, действительно оптимальный. В таблице представлены строками результаты проверки версий алгоритма с $L = 10$, $L = 50$ и $L = 100$. Для матриц размерности $N \geq 12$ проводилось 1000 экспериментов, тогда как для остальных 10000, что было сделано из-за меньшей вычислительной сложности для матриц меньшей размерности. По представленным данным можно заметить, что значительно разницы между $L = 50$ и $L = 100$ нет. Также с увеличением N уменьшается качество работы алгоритма.

Таблица 5

Доля оптимальных маршрутов алгоритма (в процентах)

L	$N = 5$	$N = 7$	$N = 10$	$N = 12$	$N = 15$	$N = 20$
10	98,93	92,85	80,09	67,1	46,8	17,5
50	99,26	98,21	89,10	78,1	65,3	32,7
100	99,34	98,58	90,45	75,6	69,1	31,0

Другая метрика представляет собой проверку на «похожесть» сгенерированной матрицы на матрицу, полученную из равномерного распределения. Результаты этой проверки приведены в таблицах 6 и 7. Таблица 6 содержит результаты тестирования для алгоритма, использующего $B_n \sim U[0,10)$, а таблица 7 — для алгоритма, использующего $B_n \sim U[0,50)$.

Таблица 6

Процент пройденных тестов алгоритмом на принадлежность матриц смежности равномерному распределению $\boldsymbol{B_n \sim U[0, 10)}$

	$N = 5$ случ.	$N = 5$ алг.	$N = 10$ случ.	$N = 10$ алг.	$N =$ 15случ.	$N = 15$ алг.	$N = 20$ случ.	$N = 20$ алг.
U-критерий	95	92	90	22	95	1	90	0
К-С	100	96	98	31	99	7	98	0

В колонках таблиц 6 и 7 приведены тесты с уровнем значимости 0.05 для алгоритма и такой же тест, только для равномерно случайной матрицы, для большей наглядности. Как видно из данных таблиц, увеличение максимального значения используемого равномерного распределения или увеличение N пагубно влияет на «равномерность» матрицы смежности, что имеет критическое значение в задачах машинного обучения. Модель машинного

обучения может переобучиться на артефактах сгенерированной матрицы, что не позволит эффективно решить задачу коммивояжёра.

Таблица 7

Процент пройденных тестов алгоритмом на принадлежность матриц смежности равномерному распределению $B_n \sim U[0, 50)$

	$N = 5$ случ.	$N = 5$ алг.	$N = 10$ случ.	$N = 10$ алг.	$N = 15$ случ.	$N = 15$ алг.	$N = 20$ случ.	$N = 20$ алг.
U-критерий	88	93	89	26	90	1	85	0
К-С	97	93	100	33	97	1	94	0

Все вышесказанное заставляет искать пути повышения эффективности алгоритма. В данной работе эффективность повышена за счёт статистических данных о ходе поиска пути в матрицах смежности, полученных с использованием равномерного распределения. $p_B(B_n)$ и $p_{V_{ij}}(v|B_n, n)$ теперь берутся как распределения значений на соответствующем шаге метода ветвей и границ. Для $p_{V_{ij}}(v|B_n, n)$ откажемся от параметров строки и столбца, поэтому далее будем обозначать как $p_V(v|B_n, n)$. Берётся ветвь, с помощью которой был получен оптимальный маршрут, и рассматриваются строки и столбцы, удаляемые пошагово. Значения в этих строках и столбцах служат основой для формирования плотностей распределений.

Пусть:

K_{all} — количество матриц, используемых для оценки $p_B(B_n)$ и $p_V(v|B_n, n)$;

$K(n)$ — количество значений в выборке, состоящей из отброшенных стоимостей матрицы смежности на n-м шаге метода ветвей и границ;

$a_i \in A$, где A — множество допустимых значений стоимостей матрицы смежности;

$k_b(n, a_i)$ — количество стоимостей, равных a_i, оптимального маршрута, найденных на n-м шаге метода ветвей и границ. Эти стоимости обозначим $b_i' = a_i$;

$k_v(n, a_i, a_j)$ — количество стоимостей, равных a_j, не принадлежащих маршруту, отброшенных на n-м шаге метода ветвей и границ, при условии, что $b_{ni}' = a_i$. Эти стоимости обозначим $v_{nij}' = a_j$. Тогда можно записать формулы 3 и 4.

$$p_B(a_i|n) = \frac{k_b(n, a_i)}{K_{all}} \quad (3)$$

$$p_V(a_i|B_n, n) = \frac{k_v(n, b_n, a_i)}{K(n)} \quad (4)$$

Покажем на примере, какие значения подсчитываются $k_b(n, a_i)$ и $k_v(n, a_i, a_j)$.

Таблица 8

Случайная матрица смежности на n-м шаге метода ветвей и границ

	1	3	4	5
1	∞	a_1	a_1	a_4
2	a_2	a_3	a_3	a_1
3	a_4	∞	a_2	a_4
4	a_4	a_1	∞	a_1

Возьмём случайную матрицу смежности на n-м шаге метода ветвей и границ, что показано в таблице 8. Пусть на n-м шаге метода ветвей и границ в маршрут добавляется значение $c_{34} = a_2$, тогда в таблице 9 показано, какие стоимости выбираются для формирования плотностей распределений $p_B(B_n)$ и $p_V(v|B_n, n)$.

Таблица 9

Выделение значений на n-м шаге метода ветвей и границ

	1	3	4	5
1	∞	a_1	v_{n21}	a_4
2	a_2	a_3	v_{n23}	a_1
3	v_{n24}	∞	${b_{n2}}'$	v_{n24}
4	a_4	a_1	∞	a_1

Ниже приведён листинг 2 псевдокода алгоритма, где функции выделены жирным шрифтом, а переменные — курсивом для удобства чтения. *Количество_вершин, K_all, min_val, max_val* соответствуют значениям N, K_{all}, нижней границе равномерного распределения, в данном случае 0, L. Функция из листинга возвращает пару **B(*шаг_алгоритма*)** и **V(*стоимость_шага_маршрута*, *шаг_алгоритма*, *размер_массива*)**, которая требуется в листинге 1.

Листинг 2. Псевдокод алгоритма генерации распределений

```
Функция ГенерацияРаспределений(количество_вершин, K_all, min_val, max_val):
B_таблица =ПустаяТаблица(
название_строк=МассивДиапазона(0,количество_вершин),
название_столбцов=МассивДиапазона(min_val, max_val))
V_массив_таблиц = ПустойМассив(размер=количество_вершин)
    для i от 0 до Длина(V_массив_таблиц):
V_массив_таблиц[i] =ПустаяТаблица(
название_строк=МассивДиапазона(min_val, max_val),
название_столбцов=МассивДиапазона(min_val, max_val))
    для k от 0 до K_all:
        матрица = ГенерацияМатрицыСтоимости(количество_вершин,
```

```
min_val, max_val)
маршрут = ПоискОптимальногоМаршрута(матрица)
        для n от 0 до Длина(маршрут):
вершина_откуда=  маршрут[n]
вершина_куда=  маршрут[(n+1)%Длина(маршрут)]
b = матрица[вершина_откуда][вершина_куда]
B_таблица[n][b] += 1
           для i от 0 до количество_вершин - n — 1:
если  матрица[вершина_откуда][i] !=∞:
V_массив_таблиц[n][b][матрица[вершина_откуда][i]] += 1
если  матрица[i][вершина_куда] !=∞:
V_массив_таблиц[n][b][матрица[i][вершина_куда]] += 1
B = Функция РаспределениеСЗахватомПеременныхB(шаг_алгоритма):
словарь_распределения = ЗначенияИВероятности(
значения=НазванияСтолбцов(B_таблица),
распределение=Нормализация(B_таблица[шаг_алгоритма]))
        вернуть РеализацияРаспределения(словарь_распределения)
    V = Функция РаспределениеСЗахватомПеременныхV(
стоимость_шага_маршрута, шаг_алгоритма, размер_массива):
V_таблица = V_массив_таблиц[шаг_алгоритма]
словарь_распределения = ЗначенияИВероятности(
значения=НазванияСтолбцов(V_таблица),
распределение=Нормализация(V_таблица[стоимость_шага_маршрута]))
массив = ПустойМассив(размер=размер_массива)
        для i от 0 до Длина(массив):
массив[i] = РеализацияРаспределения(словарь_распределения)
вернуть  массив
    вернуть B, V
```

В этом псевдокоде используются следующие вспомогательные функции:

- **ПустаяТаблица(*название_строк*, *название_столбцов*)**: возвращает пустую таблицу. Массивы *название_строк* и *название_столбцов* используются как названия строк и столбцов соответственно, формируют размер таблицы.
- **МассивДиапазона(*нижняя_граница*, *верхняя_граница*)**: формирует массив со значениями в диапазоне от *нижняя_граница* до *верхняя_граница* не включительно.
- **ГенерацияМатрицыСтоимости(количество_вершин, нижняя_граница, верхняя_граница)**: возвращает случайную матрицу стоимости для графа с количеством вершин, равным *количество_вершин*. Случайная матрица формируется из равномерного распределения U[нижняя_граница,верхняя_граница).
- **ПоискОптимальногоМаршрута(матрица)**: возвращает массив, соответствующий оптимальному маршруту матрицы стоимости *матрица.* В данной работе используется функция solve_tsp_dynamic_programming из модуля python_tsp.exact для языка python3.
- **ЗначенияИВероятности(значения, распределение)**: возвращает словарь, где ключами являются элементы массива *значения,* а плотность распределения берётся из массива *распределение.*

- **НазванияСтолбцов(таблица)**: возвращает массив названий столбцов таблицы *таблица.*
- **Нормализация(массив)**: возвращает нормализованный *массив.*
- **РеализацияРаспределения(словарь_распределения)**: возвращает реализацию случайной величины, распределенной по закону из *словарь_распределения.*
- **РаспределениеСЗахватомПеременныхB(шаг_алгоритма)**: возвращает функцию с аргументом *шаг_алгоритма,* в чей контекст попадают переменные из внешней области видимости по значению. В данном случае захватывается замыкание *B_таблица.*
- **РаспределениеСЗахватомПеременныхV(стоимость_шага_маршрута, шаг_алгоритма, размер_массива)**: возвращает функцию с аргументами *стоимость_шага_маршрута*, *шаг_алгоритма*, *размер_массива,* в чей контекст попадают переменные из внешней области видимости по значению. В данном случае захватывается замыкание *V_массив_таблиц.*

Однако в таком подходе возникает проблема решения большого количества задач коммивояжёра для того, чтобы собрать статистику. Стоит отметить, что при изменении размерности матрицы смежности или диапазона допустимых величин статистику придётся собирать заново. В таком случае алгоритм принимает другую роль, из алгоритма генерации данных алгоритм становится способом аугментации данных, что в специализированных задачах может быть даже плюсом.

Из-за потребности предварительного решения большого числа матриц и высокой вычислительной сложности решения задачи коммивояжёра количество тестов этой версии будет значительно меньше, чем у прошлой версии.

Таблица 10 — это аналог таблицы 4, только для новой версии алгоритма с $K_{all} = 100$. Сразу можно заметить высокую эффективность алгоритма как алгоритма аугментации, причём с повышением диапазона допустимых значений матрицы смежности повышается эффективность алгоритма. Явной зависимости эффективности алгоритма от размерности матрицы выделить нельзя, но можно отметить, что для любой размерности эффективность превышает 99%.

Таблица 10

Процент оптимальных маршрутов алгоритма аугментации

L	$N = 5$ ген.	$N = 5$ аугм.	$N = 10$ ген.	$N = 10$ аугм.	$N = 15$ ген.	$N = 15$ аугм.
10	98,93	99,91	80,09	99,94	46,8	99,8
50	99,26	100,00	89,10	99,99	65,3	100,0

Алгоритм аугментации показал высокие результаты не только в рамках проблемы генерации матрицы смежности по заранее известному маршруту, но

и в задаче «подражания» равномерному распределению. Новый алгоритм, согласно таблицам 11 и 12, не содержит недостатков, присущих алгоритму генерации.

Таблица 11

Процент пройденных тестов алгоритмом аугментации на принадлежность матриц смежности для $B_n \sim U[0, 10)$

	$N = 5$ случ.	$N = 5$ ген.	$N = 5$ аугм.	$N = 10$ случ.	$N = 10$ ген.	$N = 10$ аугм.	$N = 15$ случ.	$N = 15$ ген.	$N = 15$ аугм.
U-критерий	95	92	82	90	22	87	95	1	88
К-С	100	96	96	98	31	91	99	7	93

Таблица 12

Процент пройденных тестов алгоритмом аугментации на принадлежность матриц смежности для $B_n \sim U[0, 50)$

	$N = 5$ случ.	$N = 5$ ген.	$N = 5$ аугм.	$N = 10$ случ.	$N = 10$ ген.	$N = 10$ аугм.	$N = 15$ случ.	$N = 15$ ген.	$N = 15$ аугм.
U-критерий	88	93	82	89	26	93	90	1	96
К-С	97	93	96	100	33	89	97	1	88

Как видно из таблиц 11 и 12, алгоритм аугментации генерирует такие матрицы смежности, что достаточно часто их нельзя отличить от матриц, полученных из равномерного распределения с помощью U-критерия Манна-Уитни и критерия Колмогорова-Смирнова с уровнем значимости 0.05. Естественно, данное обстоятельство не может гарантировать отсутствия таких признаков в сгенерированной матрице, на которых могли бы переобучиться модели машинного обучения. Однако стоит заметить, что возможность переобучения в большой степени определяется формой подачи входных данных в модель. Поэтому проверки на наличие таких признаков надо проводить в соответствии с конкретной моделью машинного обучения.

Заключение

Целью данной работы было создать алгоритм, позволяющий генерировать матрицу смежности для задачи коммивояжёра с заранее известным маршрутом. Создан алгоритм аугментации таких матриц. Доля сгенерированных матриц,

для которых маршрут, полученный алгоритмом, совпадает с оптимальным, превышает 0.99, что позволяет успешно использовать его в моделях машинного обучения для решения задачи коммивояжера. В дальнейшем будет разрабатываться способ калибровки гиперпараметров алгоритма.

Библиографический список

1. Sudakov V.A., Belozerov I.A., Prudkova E.S. Reinforcement Machine Learning Model for Sports Infrastructure Development Planning // Math Models Comput Simul 15, 2023, pp. 608–614.
2. Löwens C., AshrafI., Gembus A., Cuizon G., Falkner J., Schmidt-Thieme L. Solving the traveling salesperson problem with precedence constraints by deep reinforcement learning. German Conference on Artificial Intelligence, 2022, pp. 160-172.
3. Bogyrbayeva A., Yoon T., Ko H., Lim S., Yun H., Kwon C. A deep reinforcement learning approach for solving the Traveling Salesman Problem with Drone // Transportation Research Part C: Emerging Technologies, Volume 148, 2023, 103981.
4. Da Costa P., Rhuggenaath J., Zhang Y. et al. Learning 2-Opt Heuristics for Routing Problems via Deep Reinforcement Learning. SN COMPUT. SCI. 2, 388, 2021.
5. Huang W., Yu J.X. Investigating TSP Heuristics for Location-Based Services. Data Sci. Eng. 2, 2017. – Pp. 71–93.
6. Некрасов В. П. Элементы дискретной математики: учебно-методическое пособие, Екатеринбург, Уральский филиал СибГУТИ, 2001. – 89 с.
7. Сперанский Д. В., Скобцов Ю. А. Эволюционные вычисления: учебное пособие, Москва, НОУ «ИНТУИТ», 2015. – 326 с.
8. Фомичев М. И. Модифицированная реализация алгоритма метода ветвей и границ для решения асимметричной задачи коммивояжёра: специальность 05.13.01 «Системный анализ, управление и обработка информации (по отраслям)»: диссертация на соискание ученой степени кандидата технических наук. – Нижний Новгород, 2022. – 120 с.