

Как я исследовал парковочные функции

От маленькой парковки к симуляции, гипотезе и доказательству

Вася Петров, ученик 8 класса

Учебный пример школьной исследовательской статьи по мотивам доклада о парковочных функциях

Аннотация

Я исследовал задачу о парковке машин на односторонней улице. У каждой машины есть предпочитаемое место; если оно занято, машина едет вправо до первого свободного. Сначала я пробовал маленькие примеры вручную и с помощью симулятора. Потом написал программу полного перебора и получил числа

1, 3, 16, 125, 1296, 16807, ...

для $n = 1, 2, \dots, 6$. По этим данным я выдвинул гипотезу $A_n = (n+1)^{n-1}$, где A_n — число успешных наборов предпочтений. ИИ-помощник помог мне проверить код и сформулировать эксперимент, но я специально попросил его не сообщать готовую формулу. После этого я познакомился с круговым доказательством Поллака. В конце я исследовал вероятность успеха и сравнил точную формулу с методом Монте-Карло.

1. С чего начался вопрос

На лекции была предложена такая модель. Есть n парковочных мест, пронумерованных $1, 2, \dots, n$, и n машин. Каждая машина заранее выбирает предпочитаемое место $a_i \in \{1, \dots, n\}$. Предпочтения могут совпадать. Машины приезжают по очереди. Если место a_i свободно, машина занимает его; если оно занято, машина движется только вправо и занимает первое свободное место. Если свободных мест справа нет, машина уезжает.

Мой главный вопрос

Какие наборы предпочтений позволяют припарковаться всем машинам? И если предпочтения выбираются независимо и равновероятно, какова вероятность успешной парковки?



Рис. 1: Пример парковки. Число на капоте — предпочитаемое место. Две машины могут хотеть одно и то же место.

Формально набор $(a_1, \dots, a_n)$ называется *парковочной функцией*, если все n машин успешно припарковались. Именно такую модель использует доклад, на котором основано это мини-исследование. В докладе эксперимент и полный перебор используются как путь к числам 1, 3, 16, 125, 1296, а затем к гипотезе и доказательству. *Это важно: сначала данные, потом догадка, затем объяснение.* [1]

2. Немного истории

Оказалось, что это не просто школьная головоломка. Идеи, связанные с парковочными функциями, восходят к работе Рональда Пайка 1959 года; модель машин и термин *parking function* связаны с работой Алана Конхайма и Бенджамина Вайса 1966 года. Их задача возникла в связи с размещением данных в памяти компьютера и методом разрешения коллизий, похожим на *linear probing*. Позже Генри Поллак предложил очень короткое круговое доказательство формулы для числа парковочных функций. Современные обзоры связывают парковочные функции с деревьями, не пересекающимися разбиениями, гиперплоскостными конфигурациями и другими объектами комбинаторики. [2, 3]

3. Мои первые попытки

Сначала я пытался понять задачу без программы.

3.1. Попытка 1: “успех возможен только без повторов”

Мне казалось, что если две машины хотят одно и то же место, одна из них обязательно проиграет. Симулятор сразу показал, что это неверно. Например, при $n = 3$ набор $(1, 1, 2)$ успешен: первая машина занимает место 1, вторая хочет 1 и переезжает на 2, третья хочет 2 и переезжает на 3.

Вывод после первой ошибки

Повтор предпочтений — не проблема сам по себе. Важно, куда машины могут сдвинуться после столкновения предпочтений.

3.2. Попытка 2: угадать формулу слишком рано

После нескольких примеров я попробовал догадаться, что ответ похож на n^{n-1} . Но уже для $n = 3$ это дало бы 9, тогда как полный перебор дал 16. Значит, нужно было собрать больше данных, а не влюбляться в первую красивую формулу.

4. Симулятор: понять правило по шагам

Я использовал пошаговый симулятор из идеи доклада: очередная машина подъезжает, я нажимаю “Следующее”, и вижу, где она окажется. Для примера (3, 3, 1, 5, 2) все машины паркуются.

Пошаговая работа симулятора: число в ячейке — предпочтение машины



Рис. 2: Один эксперимент по шагам. Симулятор оказался полезен не только для ответа, но и для понимания самого условия.

Это оказалось важнее, чем я ожидал: визуализация помогла обнаружить мою первую ошибку о повторах, а ещё дала идеи, какие маленькие случаи стоит проверить вручную.

5. Полный перебор

Для маленьких n можно проверить все возможные наборы. Всего наборов предпочтений n^n , потому что каждая из n машин независимо выбирает одно из n мест.

Для $n = 2$ есть всего четыре набора:

$$(1, 1), \quad (1, 2), \quad (2, 1), \quad (2, 2).$$

Первые три успешны, а (2, 2) неуспешен. Поэтому $A_2 = 3$ и вероятность успеха равна $3/4$.

Полный перебор дал таблицу:

n	1	2	3	4	5	6
A_n	1	3	16	125	1296	16807
n^n	1	4	27	256	3125	46656

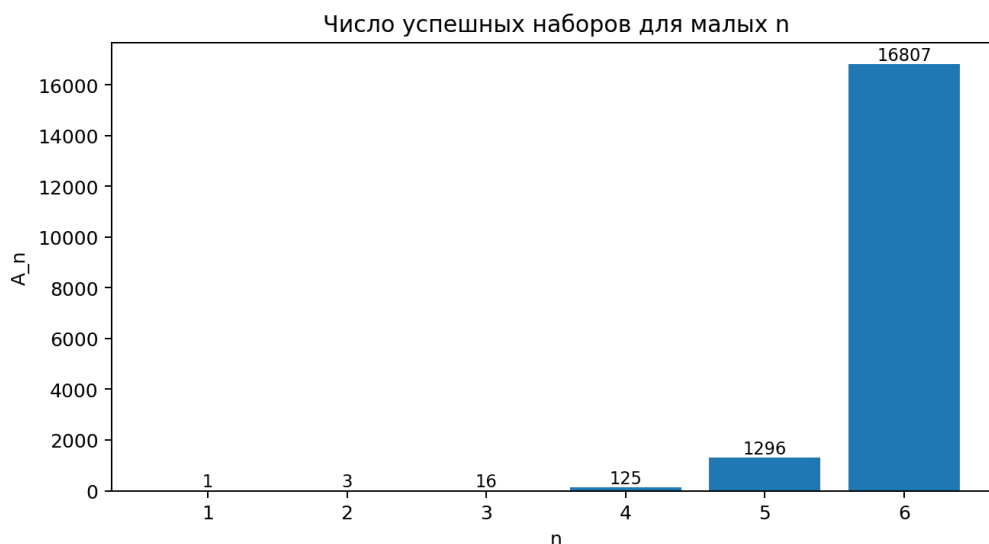


Рис. 3: Число успешных наборов быстро растёт. Но важнее было заметить структуру этих чисел.

6. Как мне помог ИИ — и как я поставил ему ограничения

Я хотел использовать ИИ как помощника, но не хотел, чтобы он сразу сообщил ответ. Поэтому написал запрос так, чтобы сохранить исследование за собой.

Мой промпт ИИ-помощнику

Я исследую парковочную задачу. Есть n мест и n машин. У каждой машины предпочтение от 1 до n ; если место занято, машина едет вправо до первого свободного. **Не сообщай мне общую формулу числа успешных наборов.** Помоги написать короткую функцию Python, которая проверяет один набор предпочтений, и код полного перебора для $n = 1, 2, 3, 4, 5, 6$. Объясни код так, чтобы его понял школьник. Если увидишь ошибку в моей логике, укажи на неё, но не решай задачу за меня.

Фрагмент ответа ИИ-помощника

“Сначала сделай функцию, которая хранит занятые места. Для каждой машины начни с её предпочтения и двигай индекс вправо, пока место занято. Если вышли за границу парковки — набор неуспешен. Затем используй `itertools.product`, чтобы перебрать все n^n наборов.”

Я переписал предложенную идею своим кодом:

Листинг 1: Проверка одного набора и полный перебор

```
from itertools import product

def parks(prefs):
    n = len(prefs)
    occupied = [False] * n
    for pref in prefs:
        place = pref - 1
        while place < n and occupied[place]:
            place += 1
        if place == n:
            return False
```

```

        occupied[place] = True
    return True

for n in range(1, 7):
    good = 0
    for prefs in product(range(1, n + 1), repeat=n):
        good += parks(prefs)
    print(n, good)

```

Программа напечатала:

```

1 1
2 3
3 16
4 125
5 1296
6 16807

```

Что сделал ИИ, а что сделал я

ИИ помог с техническим инструментом: структурой функции и идеей перебора. Выбор вопросов, эксперименты, проверка маленьких случаев, распознавание закономерности и решение о том, что считать доказательством, остались частью моего исследования.

7. Гипотеза

Теперь я посмотрел на числа иначе:

$$1 = 2^0, \quad 3 = 3^1, \quad 16 = 4^2, \quad 125 = 5^3, \quad 1296 = 6^4, \quad 16807 = 7^5.$$

Так возникла гипотеза

$$A_n = (n + 1)^{n-1}.$$

Почему это всё ещё не доказательство?

Даже если формула совпала для $n = 1, 2, \dots, 6$ и ещё для тысячи случайных тестов, остаётся вопрос: *почему она обязана работать для любого n ?* Проверки поддерживают гипотезу, но не заменяют объяснение.

8. Круговое доказательство Поллака

Здесь я уже воспользовался известной математической идеей. Добавим ещё одно место и расположим $n + 1$ мест по кругу. Теперь n машин всегда смогут припарковаться, и ровно одно место останется пустым.

Каждая машина имеет $n + 1$ вариантов предпочтения, поэтому всего круговых наборов

$$(n + 1)^n.$$

Если одновременно повернуть все номера предпочтений по кругу, положение пустого места тоже поворачивается. В каждой группе из $n + 1$ поворотов ровно один вариант оставляет пустым выбранное место, например место $n + 1$. Если разрезать круг именно там, получается успешная линейная парковка. Значит,

$$A_n = \frac{(n + 1)^n}{n + 1} = (n + 1)^{n-1}.$$

Это и объясняет числа, которые сначала появились только как результат эксперимента. [3]

9. Вероятность и Монте-Карло

Так как всего наборов n^n , точная вероятность успеха равна

$$P_n = \frac{(n+1)^{n-1}}{n^n}.$$

Например,

$$P_2 = 0.75, \quad P_3 \approx 0.593, \quad P_4 \approx 0.488, \quad P_5 \approx 0.415.$$

Для больших n полный перебор становится дорогим. Тогда можно использовать Монте-Карло: много раз случайно генерировать предпочтения и считать долю успешных парковок.

Листинг 2: Оценка вероятности методом Монте-Карло

```
import random

def monte_carlo(n, trials=20000):
    good = 0
    for _ in range(trials):
        prefs = [random.randint(1, n) for _ in range(n)]
        good += parks(prefs)
    return good / trials
```

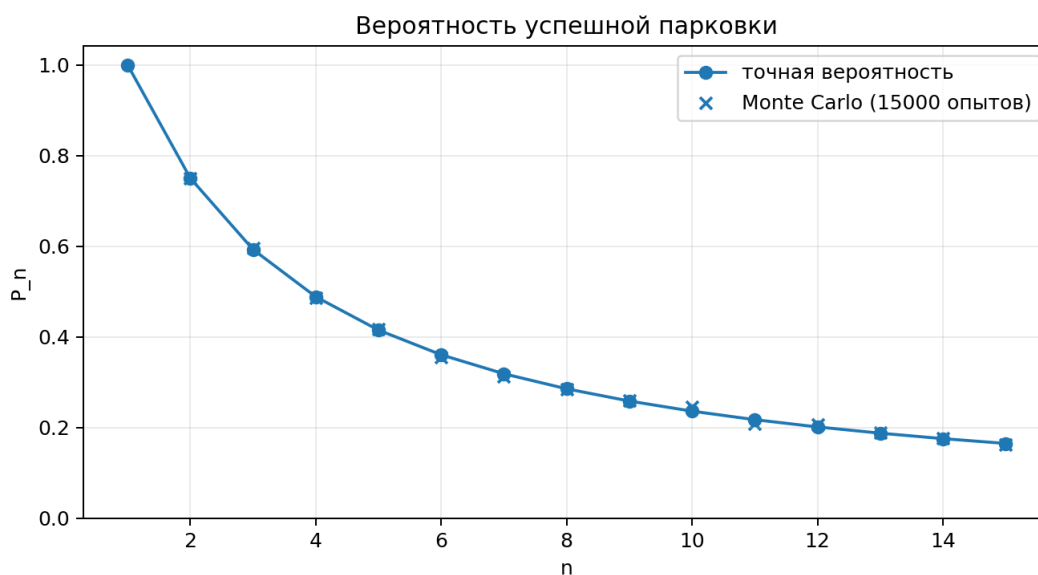


Рис. 4: Точная вероятность и оценки Монте-Карло. Симуляция хорошо приближает формулу, но сама по себе её не доказывает.

Из формулы также видно, что

$$P_n = \frac{(1 + 1/n)^{n-1}}{n} \sim \frac{e}{n},$$

поэтому вероятность успешной парковки стремится к нулю.

10. Что я узнал

Это исследование оказалось не столько про парковку, сколько про способ думать.

1. Маленькие примеры помогают понять условие и найти ошибки в интуиции.
2. Симулятор позволяет быстро увидеть динамику процесса.
3. Полный перебор даёт точные данные для малых n .
4. По данным можно сформулировать гипотезу, но нельзя объявлять её теоремой.
5. ИИ удобен как помощник по коду и проверке, особенно если явно попросить его *не раскрывать ответ*.
6. Настоящее понимание появляется, когда находится доказательство.
7. После ответа возникают новые вопросы: что будет, если мест больше, чем машин? если разрешить движение в обе стороны? если предпочтения неравновесны?

Мой главный вывод

Эксперимент помог мне увидеть закономерность; ИИ помог построить инструмент; а доказательство объяснило, почему закономерность верна. Ни один из этих этапов не был лишним.

Литература и источники

Список литературы

- [1] И. Е. Клейнер. Методический доклад о парковочных функциях, симуляции, ИИ и доказательстве. Семинар “Наука — Школе”, 2026. Материалы пользователя.
- [2] A. G. Konheim, B. Weiss. *An Occupancy Discipline and Applications*. SIAM Journal on Applied Mathematics, 14(6), 1266–1274, 1966. DOI: 10.1137/0114101.
- [3] R. P. Stanley. *A Survey of Parking Functions*. Обзор и материалы лекции: <https://math.mit.edu/~rstan/transparencies/pf.pdf>.