Wprowadzenie do symulacji i metod Monte Carlo 02.01.2026

Projekt: Optymalizacja relokacji samochodów
z wykorzystaniem łańcuchów Markowa

Paweł Lorek

DEADLINE: xx.01.2026, 23:59

1 Opis problemu

Rozważamy uproszczony model firmy wypożyczającej samochody na dużym obszarze geograficznym. Celem firmy jest podjęcie decyzji, jak relokować samochody w nocy tak, aby zmaksymalizować oczekiwany dzienny zysk.
W tej sekcji opisujemy problem w postaci ogólnej, modelowej. Konkretne wartości wszystkich parametrów używanych w projekcie zostaną podane dalej i są ustalone na stałe na potrzeby zadania.
Obszar geograficzny jest zdyskretyzowany do kwadratowej siatki o rozmiarze $L \times L$. Każda komórka siatki (x, y) reprezentuje strefę, gdzie

$$x, y \in \{0, 1, \ldots, L-1\}.$$

Stan systemu jest reprezentowany przez macierz

$$\mathbf{S} = \big(S(x,y)\big)_{x,y},$$

gdzie $S(x, y)$ oznacza liczbę samochodów znajdujących się w strefie (x, y). Łączna liczba samochodów w systemie jest stała i równa n. W konsekwencji stan spełnia

$$\sum_{x,y} S(x,y) = n.$$

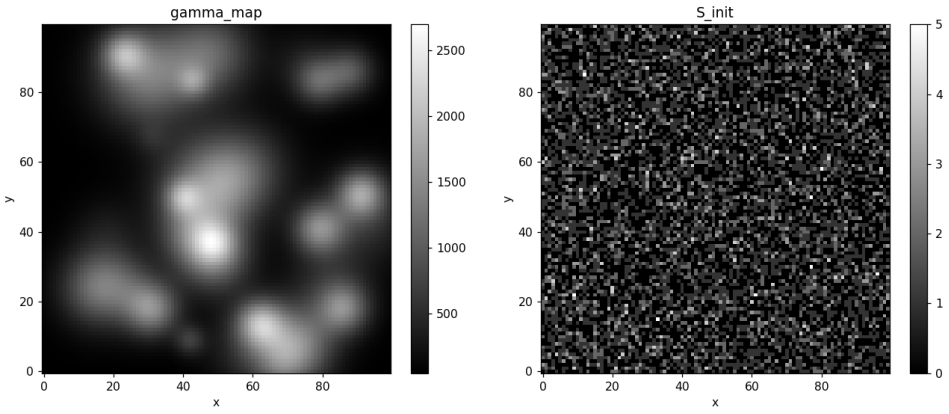
Pod koniec każdego dnia zakładamy, że samochody są losowo rozproszone po siatce. W nocy firma może relokować samochody pomiędzy strefami. Relokacja wiąże się z kosztem transportu, ale może zwiększyć całkowity dochód, jeśli samochody zostaną przeniesione do bardziej dochodowych obszarów.
Celem jest znalezienie konfiguracji $\mathbf{S}$, która równoważy te dwa efekty:

- wysoki oczekiwany dochód w następnym dniu,
- niski łączny koszt relokacji względem konfiguracji początkowej.

Model dochodu. Każda strefa (x, y) jest scharakteryzowana przez parametr dochodowości

$$\gamma_{x,y} > 0.$$

Intuicyjnie, $\gamma_{x,y}$ mierzy atrakcyjność danej strefy dla wypożyczeń samochodów. Strefy o dużych wartościach $\gamma_{x,y}$ odpowiadają obszarom o wysokim popycie, natomiast strefy o małych wartościach $\gamma_{x,y}$ – obszarom o niskim popycie.
Parametry $\gamma_{x,y}$ są ustalone dla danej instancji problemu i są dostarczone wraz z danymi. Przykładowa realizacja tzw. heatmapy $(\gamma_{x,y})$ jest pokazana na Rysunku 1 (a).



a) b)

Figure 1: a) Mapa (heatmapa) parametrów dochodowości stref $\gamma_{x,y}$. Jaśniejsze obszary odpowiadają strefom o większym oczekiwanym popycie. b) Konfiguracja początkowa $\mathbf{S}_{\text{init}}$. Samochody są rozmieszczone w przybliżeniu równomiernie po siatce. Rozmiar siatki $L \times L = 100 \times 100$ oraz łączna liczba samochodów $n = 8000$.

1.1 Funkcja lokalnego dochodu

Jeśli w strefie (x, y) na początku dnia znajduje się s samochodów, to oczekiwany dochód generowany przez tę strefę jest dany przez

$$f(x, y, s) = f(\gamma_{x,y}, s).$$

W projekcie funkcja lokalnego dochodu ma postać Michaelisa–Mentena

$$f(\gamma, s) = \gamma \, \frac{s}{s + k},$$

gdzie $k > 0$ jest ustalonym parametrem nasycenia.
Taki wybór odzwierciedla następującą intuicję ekonomiczną:

- dla małych s dochód rośnie w przybliżeniu liniowo wraz z liczbą samochodów,
- dla dużych s dochód ulega nasyceniu, co odzwierciedla ograniczony lokalny popyt,
- strefy z większym $\gamma_{x,y}$ generują większy dochód przy tej samej liczbie samochodów.

Rysunek 2 pokazuje funkcję $f(\gamma_{x,y}, s)$ dla kilku losowo wybranych stref o różnych poziomach dochodowości.

1.2 Całkowity dochód

Całkowity oczekiwany dochód odpowiadający konfiguracji $\mathbf{S}$ otrzymujemy sumując po wszystkich strefach:

$$D(\mathbf{S}) = \sum_{x,y} f\big(\gamma_{x,y}, S(x, y)\big).$$

Dla konfiguracji początkowej $\mathbf{S}_{\text{init}}$, w której samochody są losowo rozmieszczone po siatce, rozkład przestrzenny samochodów jest pokazany na Rysunku 1 (b). Ze względu na

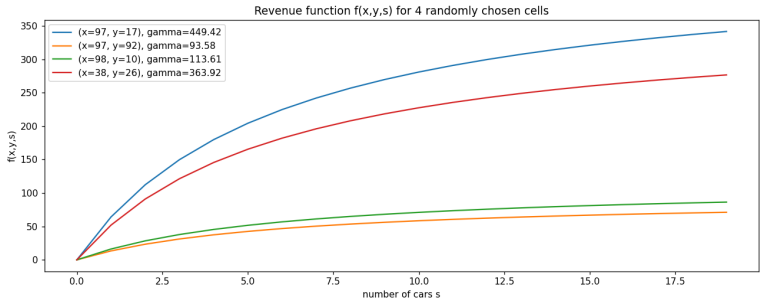


Figure 2: Funkcja lokalnego dochodu $f(\gamma_{x,y}, s)$ dla czterech losowo wybranych stref. Każda krzywa odpowiada innej wartości $\gamma_{x,y}$.

losowe rozmieszczenie samochody nie są skoncentrowane w strefach o wysokim popycie, a uzyskany dochód jest zatem suboptymalny.
W projekcie rozważamy jedną, ustaloną instancję problemu o następujących parametrach:

- rozmiar siatki: $L = 100$,
- łączna liczba samochodów: $n = 8000$,
- parametr nasycenia w funkcji dochodu: $k = 6$,
- parametr kosztu transportu $\gamma_{\text{transport}}$.

Dodatkowo instancja zawiera:

- mapę dochodowości $(\gamma_{x,y})$ na siatce,
- konfigurację początkową $\mathbf{S}_{\text{init}}$.

Wszystkie powyższe wielkości, w szczególności wartości liczbowe L, n i k, a także tablice $(\gamma_{x,y})$ i $\mathbf{S}_{\text{init}}$, są zapisane w jednym pliku `instance.npz`. Plik ten stanowi część specyfikacji zadania i nie wolno go modyfikować.
Technicznie, plik `instance.npz` przechowuje dane instancji jako tablice NumPy. Można go wczytać w Pythonie następująco:

```
import numpy as np
data = np.load("data/instance.npz")
```

a następnie odwoływać się do poszczególnych elementów instancji, np.:

```
S_init = data["S_init"]
gamma_map = data["gamma_map"]
k = data["k"]
L = data["L"]
n = data["n"]
```

Tutaj `S_init` jest dwuwymiarową tablicą NumPy o rozmiarze $L \times L$, reprezentującą początkową konfigurację samochodów, natomiast `gamma_map` jest dwuwymiarową tablicą o tym samym rozmiarze, zawierającą parametry dochodowości $\gamma_{x,y}$. Wartości `k`, `L` i `n` są zapisane jako skalarne tablice NumPy.
W całym projekcie parametry instancji są ustalone. Zadanie polega na zaprojektowaniu i przeanalizowaniu algorytmu, który aktualizuje konfigurację $\mathbf{S}$, startując od $\mathbf{S}_{\text{init}}$, tak aby poprawiać całkowity zysk.

1.3 Koszt relokacji

Relokacja samochodów pomiędzy strefami generuje koszt transportu. Intuicyjnie, przenoszenie samochodów na duże odległości jest kosztowne i powinno być wykonywane tylko wtedy, gdy prowadzi do wystarczająco dużego wzrostu oczekiwanego dochodu.
Niech $\mathbf{S}_{\text{init}}$ oznacza konfigurację początkową, a $\mathbf{S}$ konfigurację docelową uzyskaną po nocnych relokacjach. Koszt relokacji definiujemy jako łączny koszt potrzebny do przetransportowania samochodów z ich lokalizacji początkowych do lokalizacji końcowych.
W projekcie zakładamy, że koszt przetransportowania jednego samochodu ze strefy (x_1, y_1) do strefy (x_2, y_2) jest proporcjonalny do odległości Manhattan pomiędzy tymi strefami,

$$d\big((x_1, y_1), (x_2, y_2)\big) = |x_1 - x_2| + |y_1 - y_2|.$$

Odpowiadający temu koszt transportu wynosi

$$\text{cost} = \gamma_{\text{transport}}\, d\big((x_1, y_1), (x_2, y_2)\big),$$

gdzie $\gamma_{\text{transport}} > 0$ jest ustalonym parametrem określającym koszt jednostkowy (przetransportowania jednego samochodu na jednostkę odległości).
Całkowity koszt relokacji związany z przejściem z $\mathbf{S}_{\text{init}}$ do $\mathbf{S}$ otrzymujemy sumując koszty wszystkich pojedynczych przemieszczeń samochodów. Koncepcyjnie odpowiada to zagadnieniu optymalnego transportu pomiędzy dwiema dyskretnymi dystrybucjami na siatce.
Implementacja kosztu relokacji **nie jest celem** tego projektu. W pliku `helpers.py` dostarczona jest **gotowa funkcja licząca koszt relokacji** (zob. Rozdział 3.2), z której można bezpośrednio korzystać. Funkcję tę można traktować jako czarną skrzynkę; nie trzeba implementować żadnych algorytmów transportu ani dopasowań.

1.4 Całkowity zysk

Celem problemu relokacji samochodów jest maksymalizacja oczekiwanego zysku całkowitego. Zysk ten definiujemy jako różnicę pomiędzy dochodem uzyskiwanym w ciągu dnia a kosztem relokacji ponoszonym w nocy.
Niech $\mathbf{S}_{\text{init}}$ oznacza konfigurację początkową, a $\mathbf{S}$ konfigurację po relokacjach. Całkowity zysk odpowiadający konfiguracji $\mathbf{S}$ definiujemy jako

$$\text{profit}(\mathbf{S}) = D(\mathbf{S}) - C(\mathbf{S}_{\text{init}}, \mathbf{S}), \tag{1.1}$$

gdzie

- $D(\mathbf{S})$ oznacza całkowity oczekiwany dochód dla konfiguracji $\mathbf{S}$,
- $C(\mathbf{S}_{\text{init}}, \mathbf{S})$ oznacza całkowity koszt relokacji potrzebny do przejścia z $\mathbf{S}_{\text{init}}$ do $\mathbf{S}$ (liczony funkcją `relocation_cost` dostarczoną w pliku `helpers.py`).

Składnik $D(\mathbf{S})$ faworyzuje umieszczanie samochodów w strefach o wysokich wartościach $\gamma_{x,y}$, natomiast składnik $C(\mathbf{S}_{\text{init}}, \mathbf{S})$ penalizuje relokacje na duże odległości. Wynikowe zadanie optymalizacyjne polega więc na znalezieniu kompromisu pomiędzy tymi dwoma efektami.
W całym projekcie konfiguracja początkowa $\mathbf{S}_{\text{init}}$ jest ustalona. Zadanie polega na zaprojektowaniu algorytmu, który generuje kolejne konfiguracje o rosnących wartościach $\text{profit}(\mathbf{S})$.

2 Zadanie

Celem projektu jest znalezienie konfiguracji $\mathbf{S}^*$ o dużym całkowitym zysku, najlepiej bliskiej optymalnej

$$\mathbf{S}^{\text{opt}} = \arg\max_{\mathbf{S}} \text{profit}(\mathbf{S}),$$

gdzie

$$\text{profit}(\mathbf{S}) = D(\mathbf{S}) - C(\mathbf{S}_{\text{init}}, \mathbf{S}).$$

Ponieważ przestrzeń wszystkich możliwych konfiguracji jest ogromna, przeszukanie zupełne jest niemożliwe. Twoim zadaniem jest zatem implementacja algorytmu Metropolisa (lub Metropolisa–Hastingsa), który konstruuje łańcuch Markowa

$$\mathbf{S}_{\text{init}} \;\to\; \mathbf{S}^{(1)} \;\to\; \mathbf{S}^{(2)} \;\to\; \cdots,$$

i ma tendencję do przechodzenia w kierunku konfiguracji o większych wartościach $\text{profit}(\mathbf{S})$. W każdej iteracji należy zaproponować modyfikację aktualnej konfiguracji $\mathbf{S}$ (zgodnie z pewnym *mechanizmem propozycji / macierzą generowania kandydatów*) oraz zaakceptować bądź odrzucić propozycję zgodnie z regułą akceptacji Metropolisa, na podstawie zmiany wartości $\text{profit}(\mathbf{S})$.
Można korzystać z funkcji dostarczonych w pliku `helpers.py` do obliczania:

- całkowitego dochodu $D(\mathbf{S})$,
- kosztu relokacji $C(\mathbf{S}_{\text{init}}, \mathbf{S})$.

Implementacja samego kosztu relokacji nie jest częścią zadania.

Pliki do oddania. Należy oddać:

- plik Pythona `main_123456.py`, gdzie `123456` należy zastąpić numerem indeksu,
- raport w formacie PDF.

Raport. Należy przygotować krótki raport opisujący algorytm i wyniki eksperymentów. Raport musi zawierać:

- jasny opis mechanizmu propozycji oraz reguły akceptacji zastosowanej w algorytmie,
- wykres $\text{profit}(\mathbf{S}^{(t)})$, tj. całkowity zysk w kolejnych iteracjach,
- wizualizację relokacji wykonanych przez algorytm, zgodnie z opisem poniżej.

Wizualizacja relokacji. Oprócz wykresu zysku należy dołączyć rysunek pokazujący, *gdzie* algorytm relokował samochody. Wizualizacja powinna bazować na różnicy pomiędzy konfiguracją końcową a konfiguracją początkową,

$$\Delta\mathbf{S} = \mathbf{S}^{(\text{final})} - \mathbf{S}_{\text{init}}.$$

Rysunek musi być wykonany zgodnie z następującą konwencją:

- mapa $(\gamma_{x,y})$ jest pokazana w tle jako heatmapa w skali szarości,
- strefy, z których *zabrano* samochody (tj. $\Delta S(x,y) < 0$), są oznaczone kolorem **niebieskim**,
- strefy, do których *dodano* samochody (tj. $\Delta S(x,y) > 0$), są oznaczone kolorem **czerwonym**,
- wielkość $|\Delta S(x,y)|$ może być odzwierciedlona w rozmiarze znaczników, tak aby większe relokacje były wyraźniej widoczne.

Przykład wymaganej wizualizacji jest pokazany na Rysunku 3.

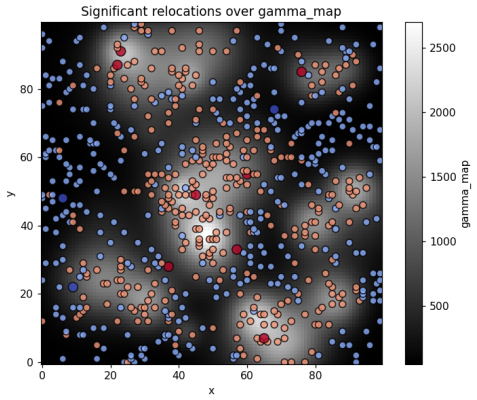


Figure 3: Wizualizacja istotnych relokacji samochodów. Tło przedstawia mapę dochodowości $(\gamma_{x,y})$ w skali szarości. Czerwone znaczniki oznaczają strefy, do których dodano samochody $(\Delta S(x,y) > 0)$, natomiast niebieskie znaczniki oznaczają strefy, z których zabrano samochody $(\Delta S(x,y) < 0)$. Rozmiar znacznika jest proporcjonalny do $|\Delta S(x,y)|$.

Funkcja pomocnicza `plot_relocations`, generująca taką wizualizację, jest dostępna w pliku `helpers.py`. Można użyć jej bezpośrednio lub w razie potrzeby zmodyfikować; jednak konwencja kolorów (czerwony = dodane samochody, niebieski = zabrane samochody) musi zostać zachowana.

3 Dostarczone pliki

3.1 Plik `sample_visualize.py`

Oprócz pliku `instance.npz` dostarczony jest skrypt Pythona `sample_visualize.py`. Celem tego pliku jest umożliwienie wstępnej eksploracji instancji oraz zilustrowanie struktury dostarczonych danych.
Skrypt wykonuje następujące kroki:

- wczytuje dane instancji z `instance.npz`,
- wypisuje wartości liczbowych głównych parametrów modelu,
- wizualizuje mapę dochodowości $(\gamma_{x,y})$ jako heatmapę,

- wizualizuje konfigurację początkową $\mathbf{S}_{\text{init}}$,
- rysuje funkcję lokalnego dochodu $f(\gamma_{x,y}, s)$ dla kilku losowo wybranych stref.

Rysunki generowane przez `sample_visualize.py` są dołączone do niniejszego opisu projektu i stanowią punkt odniesienia do interpretacji parametrów modelu. W szczególności ilustrują one niejednorodność przestrzenną popytu zakodowaną w mapie $(\gamma_{x,y})$ oraz nasycający charakter funkcji lokalnego dochodu.
Plik `sample_visualize.py` służy wyłącznie do podglądu i eksperymentów. Nie ma potrzeby jego modyfikowania. Jego rola polega na dostarczeniu intuicji dotyczącej instancji oraz na weryfikacji, że dane zostały poprawnie wczytane, zanim rozpoczniesz implementację algorytmu optymalizacji.

3.2 Szablon `main.py`

Plik `main.py` pełni rolę szablonu pokazującego, jak wczytać dane instancji oraz jak obliczać podstawowe wielkości związane z problemem.
W szczególności skrypt wykonuje następujące kroki:

- wczytuje ustaloną instancję problemu z `instance.npz`,
- oblicza całkowity dochód dla konfiguracji początkowej $\mathbf{S}_{\text{init}}$,
- wykonuje niewielką liczbę losowych relokacji samochodów,
- oblicza dochód oraz koszt relokacji dla zmodyfikowanej konfiguracji,
- porównuje wyniki przed i po relokacjach.

Celem `main.py` jest wyłącznie demonstracja podstawowych obliczeń. Skrypt pokazuje, jak dla danej konfiguracji wyznaczyć dochód oraz koszt relokacji i jak zmiany w konfiguracji wpływają na te wielkości.
Zachęcamy do traktowania `main.py` jako punktu startowego (szkieletu) dla własnych implementacji. W szczególności, skrypt można naturalnie rozszerzyć, zastępując losowe relokacje regułami aktualizacji wynikającymi z metod MCMC, itp.