Projekt 2 – Błądzenie losowe i metoda Metropolisa

Szymon Słomkowski, indeks nr 260684

1. Wstęp

Błądzenie losowe to proces stochastyczny polegający na tym, że w każdej kolejnej chwili, nasz punkt przemieszcza się z jednego stanu w inny, losowo wybrany.
W przypadku błądzenia losowego po grafie przemieszczamy się z jednego wierzchołka grafu do innego, sąsiadującego z nim. Prawdopodobieństwo wybrania konkretnego sąsiedniego wierzchołka jest równe 1/d(wierzchołek startowy), gdzie d oznacza liczbę sąsiednich wierzchołków.
Metoda Metropolisa to metoda próbkowania typu MCMC, czyli próbkowania Monte Carlo łańcuchami Markowa. Polega on na odrzucaniu losowo wybranego kandydata przez porównaniu losowej liczby z rozkładu jednostajnego i wartości funkcji akceptującej.

2. Budowa grafu

Centralnym obiektem projektu jest klasa Graph zaimplementowana w module graphing. Przechowuje ona dwie informacje – listę wierzchołków zawierającą obiekty klasy Node oraz listę krawędzi. Wierzchołki będące obiektami również przechowują dwie informacje – wartość numeryczną danego wierzchołka oraz listę jego sąsiadów, czyli wierzchołków połączonych z nim krawędziami.
Klasa Graph posiada 8 metod:
add_node – pozwalająca dodać nam wierzchołek do grafu (pamiętając o tym, że wierzchołki muszą być klasy Node)
add_nodes_from – umożliwiająca nam dodać całą listę wierzchołków
add_edge – która pozawala nam na dodanie krawędzi, jednocześnie sprawdzając, czy wierzchołki zawarte w krawędzi istnieją – jeśli nie, to również dodaje je do grafu
add_edges_from – pozwalająca nam dodać całą listę krawędzi
print_graph – drukująca do konsoli kolejne wierzchołki wraz z listami ich sąsiadów
draw_graph – tworzy figurę z biblioteki matplotlib.pyplot reprezentującą nasz graf
check_node – zwraca informację, czy wierzchołek o podanej wartości numerycznej znajduje się w grafie

find_node – wyszukuje wierzchołek (obiekt klasy Node) o podanej wartości i zwraca go. W przypadku nieodnalezienia takiego obiektu, zwraca None.

Graf w błądzeniu losowym bardzo przystępnie może oddawać rozkład stacjonarny, czyli taki rozkład prawdopodobieństwa, który spełnia równanie $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$.

2.1. Zad. 1 Zaproponować graf o podanym rozkładzie stacjonarnym Π_rw

Π_rw = (2/20, 3/20, 5/20, 2/20, 1/20, 3/20, 4/20)

Ułamek 2/20 w tym rozkładzie informuje nas o tym, że do danego wierzchołka dołączone są dwie krawędzie z łącznej sumy 20 krawędzi w grafie. W ten sposób możemy utworzyć taki graf:

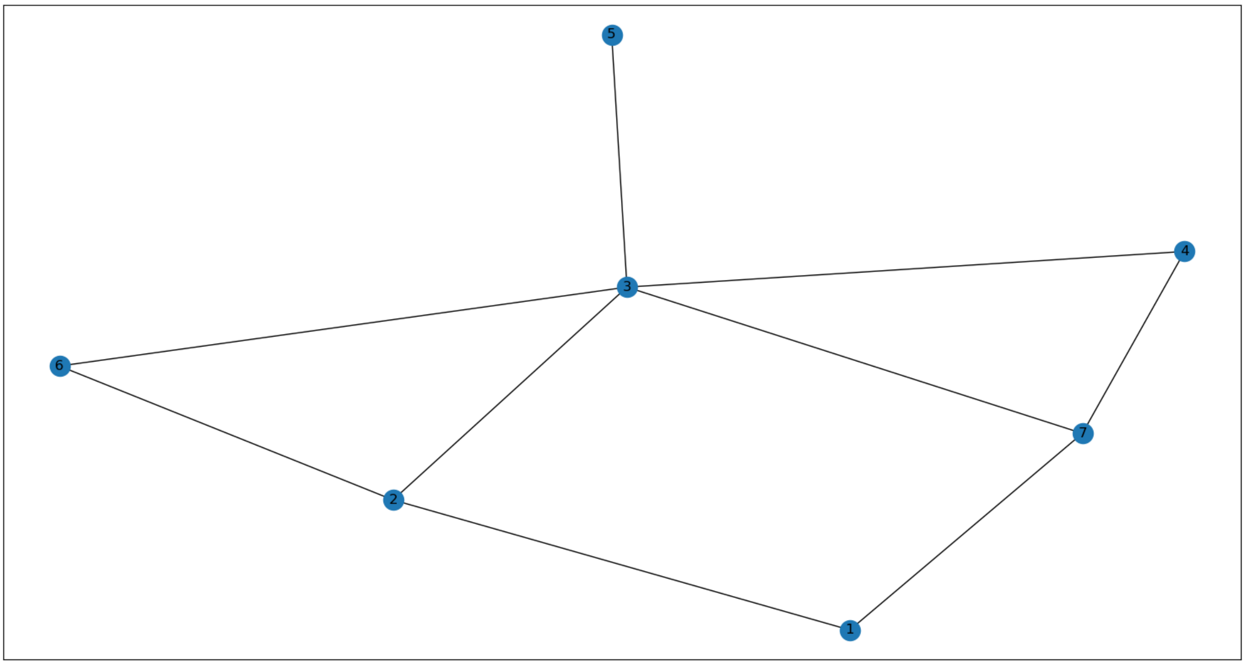


Fig 1: Graf o zadanym rozkładzie stacjonarnym Π_rw

1 -> [2, 7]

2 -> [1, 3, 6]

3 -> [2, 4, 5, 6, 7]

4 -> [3, 7]

5 -> [3]

6 -> [2, 3]

7 -> [1, 3, 4]

3. Funkcja błądzenia losowego i metoda Metropolisa

Funkcja *random_walk*() jest funkcją odpowiedzialną za wykonywanie błądzenia losowego w projekcie. Przyjmuje dwa parametry: graph, będący grafem, po którym chodzimy i current_node czyli wierzchołek, z którego będziemy wychodzić. Najpierw funkcja wylicza dystrybuantę prawdopodobieństwa przejścia do jednego z sąsiednich wierzchołków, dając każdemu z wierzchołków takie samo prawdopodobieństwo bycia wybranym. Następnie zostaje wylosowana wartość od 0, 1 przy użyciu funkcji *random*() z biblioteki random. Ostatecznie wylosowana wartość zostaje porównana z dystrybuantą i na podstawie tego funkcja zwraca losowo wybrany sąsiedni wierzchołek (obiekt klasy Node).

Funkcja *metropolis_walk*() odpowiada za implementację algorytmu Metropolisa. Poza parametrami graph i current_node przyjmuje jeszcze function_list, będące listą znanych wartości funkcji $f(i) = \frac{\Pi(\mathrm{i})}{\Pi\_rw(i)}$, gdzie Π to rozkład stacjonarny, do którego dążymy, a Π_rw to rozkład stacjonarny naszego grafu, oraz steps, czyli liczbę kroków jaką chcemy wykonać. Po zainicjowaniu pierwotnych elementów takich jak lista przechowująca nasze wartości funkcja wykonuje następującą pętlę:

1) Wylosuj przy pomocy funkcji random_walk() losowego sąsiada naszego obecnego wierzchołka.
2) Wyznacz stosunek akceptacji, ze wzoru $\alpha = \min\left(1, \frac{f(losowy\ sąsiad)}{f(obecny\ wierzchołek)}\right)$.
3) Wylosuj dowolną wartość u z rozkładu jednostajnego (0, 1).
4) Porównaj u i α, jeżeli u <= α, to zaakceptuj, dodaj wylosowanego sąsiada do listy wierzchołkami i ustaw go jako obecny wierzchołek. Jeżeli u > α, to odrzuć, czyli jako kolejny wierzchołek ustaw obecny wierzchołek.

Na koniec, po wykonaniu określonej ilości kroków zwróć listę kolejnych odwiedzonych wierzchołków.

3.1. Zad. 2 Zaimplementować metodę Metropolisa by uzyskać na tym samym grafie błądzenie o rozkładzie stacjonarnym:

Π = (1/20, 2/20, 8/20, 2/20, 2/20, 2/20, 3/20)

Wykorzystując twierdzenie ergodyczne zamiast wykonać dużą ilość powtórzeń, wykonałem dużą ilość kroków (10.000). Otrzymane wyniki umieściłem na dwóch histogramach zestawiając je z teoretycznymi rozkładami.

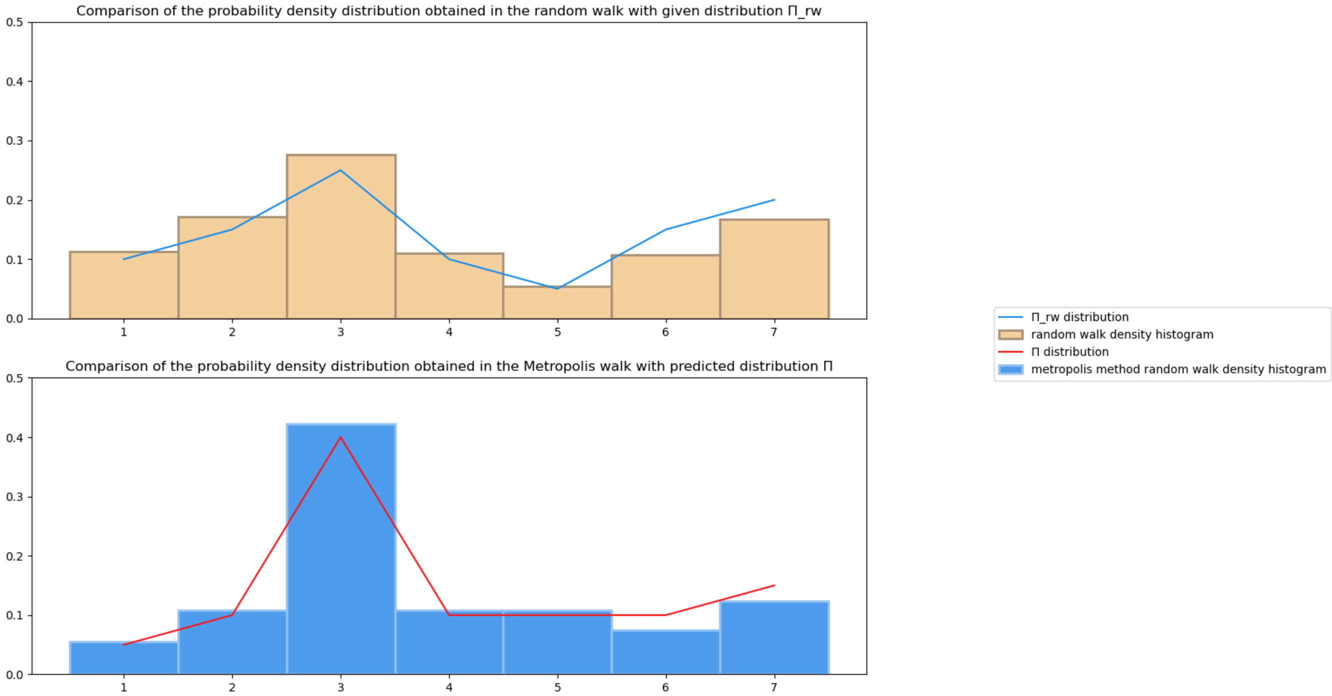
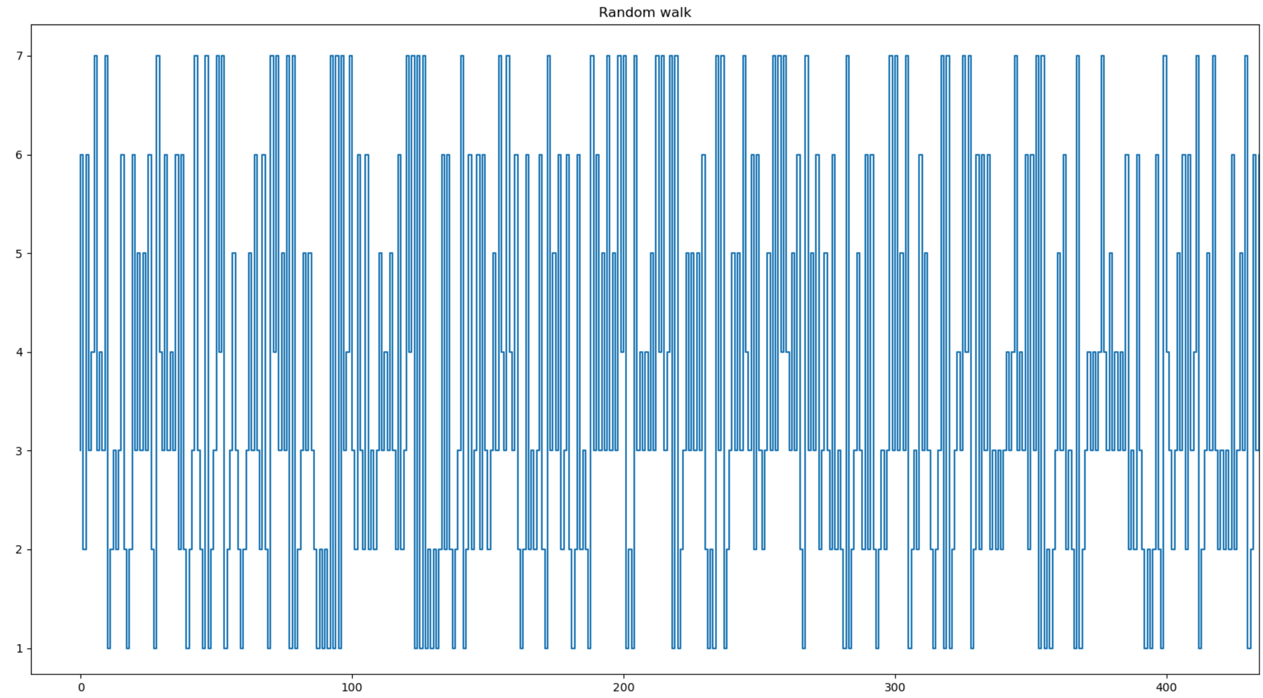


Fig 2 Zestawienie otrzymanych wyników z teoretycznymi rozkładami

Jak widać na drugim wykresie, metoda Metropolisa pozwoliła nam w dużym stopniu przybliżyć rozkład Π wykonując błądzenie losowe po grafie o rozkładzie stacjonarnym Π_rw. Prawdopodobnie zwiększenie ilości wierzchołków w grafie (obecnie jest ich tylko 7) pozwoliłoby z jeszcze większą dokładnością przybliżyć rozkład Π.



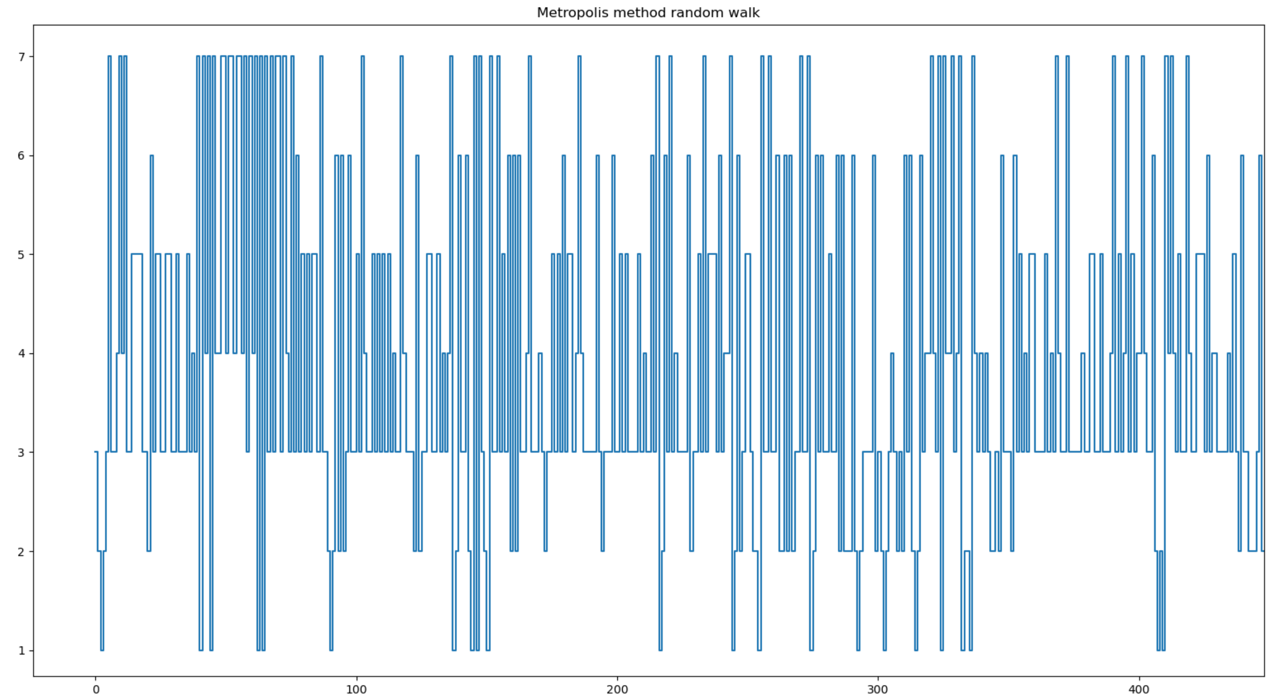


Fig. 3 i 4. Fragment wykresu krokowego reprezentującego kolejne kroki błądzenia losowego (Fig. 3.) i błądzenia losowego z metodą Metropolisa (Fig. 4.)