AGH
Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca inżynierska

Klaudia Fil

kierunek studiów: **informatyka stosowana**

Problem chińskiego listonosza w sieci ulic w Krakowie

Opiekun: **dr hab. inż. Przemysław Gawroński**

Kraków, styczeń 2021

Oświadczenie studenta

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. — Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

.....

(czytelny podpis)

Ocena merytoryczna opiekuna

Ocena merytoryczna recenzenta

Spis treści

1 Wstęp

1.1 Wprowadzenie

Szeroko pojęty problem związany z wyznaczaniem trasy (ang. General Routing Problem) zdefiniowano jako szukanie drogi o minimalnym koszcie, która dodatkowo musi spełniać uwzględnione w planowaniu wymagania[1]. Skupiając się na praktycznym aspekcie GRP należy wziąć pod uwagę jeden z bardziej znanych przypadków rozważań - problem chińskiego listonosza (ang.Chinese Postman Problem)[2].

W życiu codziennym wiele zawodów związanych jest wyznaczaniem trasy na tle procesów logistycznych, wśród nich różnego rodzaju dostawca, kurier czy właśnie listonosz. Ich szlakiem docelowym jest droga zawierająca w sobie każdą ulicę danego obszaru przynajmniej raz. Optymalnym rozwiązaniem CPP byłoby uniknięcie ponownych przejść jedną ścieżką, jednak w sytuacjach rzeczywistych jest to często niemożliwe.

1.2 Cel pracy

Celem pracy dyplomowej jest zaimplementowanie dwóch algorytmów, rozwiązujących kwestię znajdowania pełnej ścieżki z cyklem zamkniętym na grafie spójnym, przy optymalizacji kosztów przemieszczania się, co sprowadza się do CPP. Rozpatrzono w niej przypadki rzeczywiste, gdzie w powstałych z mapy Krakowa grafach mieszanych (ang. Mixed Graph) oraz nieskierowanych (ang. Undirected Graph), znajdowana jest najlepsza droga dla listonosza[3].

Jeden z algorytmów przedstawiony w pracy dedykowany jest MG. Swoje zastosowanie ma, kiedy przebieg trasy listonosza uwarunkowany jest czynnikami zewnętrznymi tj. drogami jedno- lub dwukierunkowymi. W przypadku pieszego ruchu, w którym sposób przejścia jest dowolny, problem dotyczy UG. Zaimplementowano dla niego drugi z schematów umożliwiający wyzna-

czenie najbardziej efektywnej trasy. Graf rzeczywisty tworzony jest z małych fragmentów miasta, dlatego założono, że poruszać się będzie na nim tylko jeden listonosz.

Dla sprawdzenia poprawności oraz wyznaczenia złożoności algorytmów wygenerowano losowe sieci, które umożliwiły szersze ich przetestowanie. Stworzono również program do wizualizacji grafów, oraz kolejnych etapów ścieżek, umożliwiający graficzne prześledzenie działania schematów.

1.3 Problemu chińskiego listonosza

Problem chińskiego listonosza jest jedynym z podstawowych zagadnień związanych z wyznaczaniem trasy. Swoją nazwę zawdzięcza chińskiemu matematykowi, który jako pierwszy go sformułował. Analizował drogę dostawcy w jednym z chińskich miast, który miał dostarczyć wszystkie przesyłki, odwiedzając każdą ulice, nie nadkładając niepotrzebnie drogi i powrócić do bazy[4].

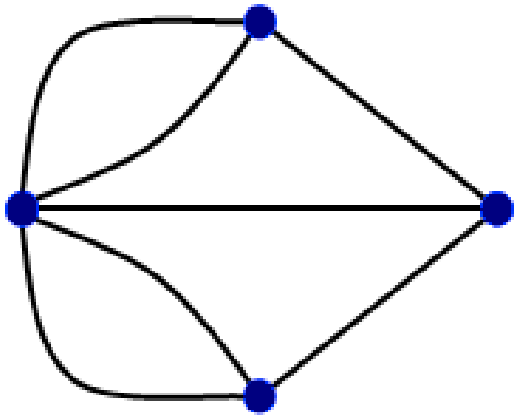
Patrząc przez pryzmat przypadku rzeczywistego, gdzie rozważany jest listonosz, czy dostawca ulotek, CPP sprowadza się do procesu wyboru najlepszej ścieżki w sieci dróg, gdzie każda ulica musi zostać odwiedzona przynajmniej raz, czyli odnalezienia na grafie cyklu Eulera[5].

1.4 Cykl Eulera

Cykl Eulera (ang. Eulerian Cycle) jest to ścieżka poprowadzona na dowolnym grafie, która przechodzi przez każdą z krawędzi dokładnie raz. Nazywamy ją cyklem, ponieważ zaczyna się i kończy w tym samym wierzchołku.

Nazwa takiego przejścia przez graf nadana została na cześć matematyka Leonharda Eulera, który w swojej pracy *Solutio problematis ad geometriam situs pertinentis*[6] poruszył zagadnienie mostów królewieckich. Odpowiadał

na pytanie, czy istnieje ścieżka prowadząca przez wszystkie mosty, biorąc pod uwagę, że przez każdy z nich można iść tylko raz.



Rysunek 1: Graf przestawiający problem mostów w Królewcu

Mapa, którą rozważał, sprowadzała się do grafu z rysunku 1, gdzie krawędzie odpowiadają siedmiu mostom. Wykazał, że znalezienie EC jest możliwe tylko dla grafów spełniających określone warunki, nazwanych na jego cześć grafami Eulera. Była to jedna z pierwszych prac związanych z teorią grafów.

1.4.1 Graf Eulera

Twierdzenie 1.1 (Euler, 1736) *Graf spójny jest eulerowski wtedy i tylko wtedy, gdy stopień każdego wierzchołka jest liczbą parzystą.*

Twierdzenie 1.2 *Graf skierowany spójny zawiera cykl Eulera wtedy i tylko wtedy, gdy dla każdego wierzchołka v zachodzi $d^+(v) = d^-(v)$, gdzie*

$d^+(v)$ - ilość krawędzi wchodzących do wierzchołka v,

$d^-(v)$ - ilość krawędzi wychodzących z wierzchołka v.

Aby możliwe było znalezienie poprawnej ścieżki dzięki opisanym w tej pracy algorytmom, grafy w nich wykorzystywane muszą spełniać twierdzenia 1.1 i 1.2.

W przypadku wygenerowania UG niezgodnego z 1.1 wywoływana jest funkcja odpowiedzialna za dodanie duplikatów krawędzi, które zapewnią poprawny stopnień każdego z wierzchołków. Wierzchołki o nieparzystych stopniach dobierane są w możliwie najlepsze pary (wykorzystując algorytm najkrótszych ścieżek) i łączone ze sobą.

Analogiczna sytuacja jest dla DG w odniesieniu do twierdzenia 1.2.

1.5 Sieci złożone

1.5.1 Sieć Barabási–Albert

Pod koniec XX w. na Uniwersytecie Notre Dame w Indianie Reka Albert i Albert-Laszlo Barabasi podczas badania struktury sieci WWW przypadkiem odkryli ciekawą zależność, a mianowicie potęgowe rozkłady prawdopodobieństwa opisujące połączenia pomiędzy stronami internetowymi.

Rozkład ten wyglądał następująco:

$$P(k) \sim k^{-\alpha} \tag{1}$$

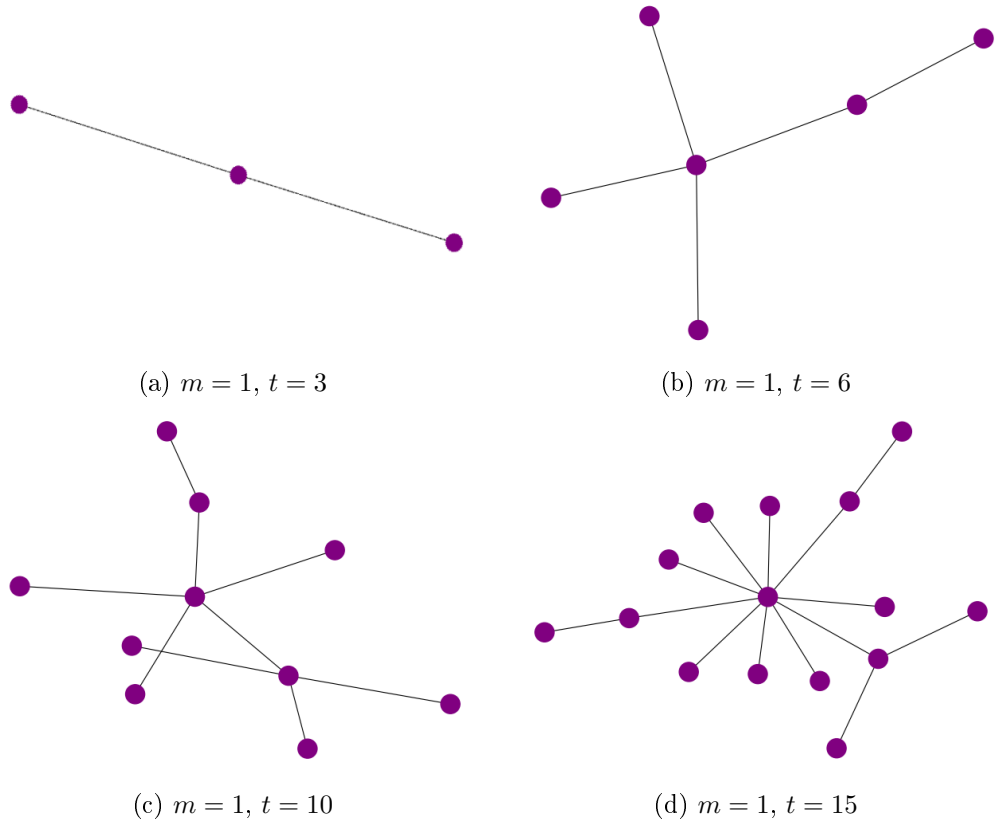
co oznaczało, że sieć WWW ma własności fraktalne.

Zauważono, że większość układów rzeczywistych również charakteryzuje się rozkładem podobnym do równania 1. Szerszą analizę i reguły tworzenia sieci tego typu Panowie przedstawili w swojej pracy zatytułowanej *Emergence of scaling in random networks*[7]. Ustalili, że rozkład 1 wynika z wzrostu sieci i reguły związanej z preferencyjnym dołączaniem węzłów.

Formułowanie ewoluującej sieci polega na stopniowym jej rozroście w każdym momencie czasowym. Proces rozpoczęty w chwili zerowej składa się z grafu zawierającego m_0 połączonych wierzchołków. Podczas kolejnych kroków czasowych nowo dodawany węzeł łączony jest z m innymi, istniejącymi już. Szansa, że świeżo utworzony wierzchołek połączony zostanie krawędzią do starego węzła jest proporcjonalne do stopnia węzła k_i:

$$\Pi(k_i) = \frac{k_i}{\sum_{i=1}^{t} k_i} \tag{2}$$

co przedstawione jest na rysunku 2.



(a) $m = 1$, $t = 3$ (b) $m = 1$, $t = 6$

(c) $m = 1$, $t = 10$ (d) $m = 1$, $t = 15$

Rysunek 2: Graf losowy stworzony zgodnie z modelem Barabási–Albert

Algorytm posiłkowy, z którego skorzystano w pracy, tworzący grafy w wyżej opisanym modelu to *barabasi_albert_graph*, zaczerpnięto go z biblioteki *Networkx* szerzej opisanej w rozdziale 2.

1.5.2 Sieć Watts–Strogatz

Zjawiskiem małego świata tylko dla pewnego ograniczonego zakresu, zależnie od wielkości sieci.[8]

2 Narzędzia

Aplikacja w całości została zaimplementowana w Pythonie, ze względu na jego prostotę i czytelność, które ułatwiały znacznie pracę. Innym atrybutem przemawiającym za tym językiem jest duża ilość bibliotek związanych z teorią grafów. Jedna z nich została wykorzystana przy tworzeniu projektu, m. in. do wyliczania najkrótszych ścieżek, ale również przy tworzeniu losowych grafów.

Biblioteki użyte w pracy:

- *Networkx*[11] - biblioteka w języku Python używana przy badaniu i tworzeniu dużych grafów i sieci. Wykorzystana została w algorytmie konwertującym graf do takiego, który posiada ścieżkę Eulera opisaną w 1.4 i podczas tworzenia przypadkowego digrafu oraz sieci złożonych z podpunktu 1.5.

- *Matplotlib*[12] - jedna z najbogatszych bibliotek do tworzenia wykresów w języku Python. W pracy gównie wykorzystano zawarte w niej API *pylab* wykorzystujące prosty interface analogiczny do środowiska MATLAB[13].

Inną istotną rzeczą użytą w projekcie to mapa świata OpenStreetMap (OSM)[14], która zbudowana jest z danych gromadzonych przez szeroką społeczność. Dowolna osoba może przyczynić się do jej rozwoju poprzez dostarczenie informacji geograficznych zbieranych przez urządzenia z odbiornikami GPS lub zdjęcia satelitarne. Dane w niej są udostępnione za darmo do ogólnego użytku.

3 Algorytmy i implementacja

3.1 Modyfikacja do grafu Eulera

3.1.1 Modyfikacja grafu nieskierowanego

dddd

Algorytm 1: *makeEulerianGraph* przekształcający graf nieskierowany do grafu Eulera

```
1  def makeEulerianGraph(graph):
2          listOfOddDegreeNodes = []
3
4          for node in graph.nodes(data=True):
5                  if idOddNumber(graph.degree[node[0]]):
6                          listOfOddDegreeNodes.append(node[0])
7
8          if idOddNumber(len(listOfOddDegreeNodes)):
9                  return None
10
11         listOfOddDegreeNodes1 = listOfOddDegreeNodes[:len(listOfOddDegreeNodes) // 2]
12         listOfOddDegreeNodes2 = listOfOddDegreeNodes[len(listOfOddDegreeNodes) // 2:]
13         listOfRandomParams = list()
14
15         for x in range(0, 100):
16                 random.shuffle(listOfOddDegreeNodes2)
17                 listOfRandomParams.append(list(zip(listOfOddDegreeNodes1,
18                                                    listOfOddDegreeNodes2)))
19
20         minPath = getOptimalAdditionalPaths(graph, listOfRandomParams)
21         appendFakeEdges(graph, minPath)
```

sddd

3.1.2 Modyfikacja grafu skierowanego

dddd

Algorytm 2: *makeEulerianDiGraph* przekształcający graf skierowany do grafu Eulera

```
1  def makeEulerianDiGraph(graph):
2          diff = [None] * len(
3                          graph.nodes)  # list with nodes difference :
4                                        # predecessors nodes - successors nodes
5          for node in graph.nodes(data=True):
6                  diff[node[0]] = len(graph.pred[node[0]]) - len(graph.succ[node[0]])
7
8          negNodes = []
9          posNodes = []
```

```
10          for node in range(0, len(diff)):
11                  if diff[node] < 0:
12                          for rep in range(-1 * diff[node]):
13                                  negNodes.append(node)
14                  if diff[node] > 0:
15                          for rep in range(diff[node]):
16                                  posNodes.append(node)
17
18          if idOddNumber(len(negNodes) - len(posNodes)):
19                  return None
20
21          listOfRandomParams = list()
22
23          for x in range(0, 100):
24                  random.shuffle(negNodes)
25                  listOfRandomParams.append(list(zip(posNodes, negNodes)))
26
27          minPath = getOptimalAdditionalPaths(graph, listOfRandomParams)
28          appendFakeEdges(graph, minPath)
```

sddd

3.2 Graf rzeczywisty

Algorytmy docelowo szukają optymalnej trasy na planie miasta Kraków. Poprawne ich zastosowanie wymaga konwersji mapy na postać grafową.

Przetwarzając mapę z rysunku 3 formujemy graf, którego krawędzie odpowiadają ulicom, a wierzchołki reprezentują zarówno budynki, jak i istotne elementy dróg: skrzyżowania i strategiczne punkty trasy, które to potrzebne są do idealnego przeprowadzenia ulicy, czyli uwzględnienia zakrętów lub nieliniowych fragmentów drogi.

Program odpowiedzialny za generowanie grafu, przy pobieraniu danych z bazy OpenStreetMap, filtruje odpowiednie informacje o obiektach i znajduje placówkę pocztową (o ile taka istnieje na danych fragmencie miasta) dzięki konkretnym tagom[15]. W przypadku wybrania fragmentu mapy, gdzie nie ma poczty, to miano nadawane jest losowo wybranej nieruchomości.

Wierzchołki w grafie podpisano rzeczywistymi adresami oraz oznaczono

[1] Zrzut ekranu z strony https://www.openstreetmap.org/

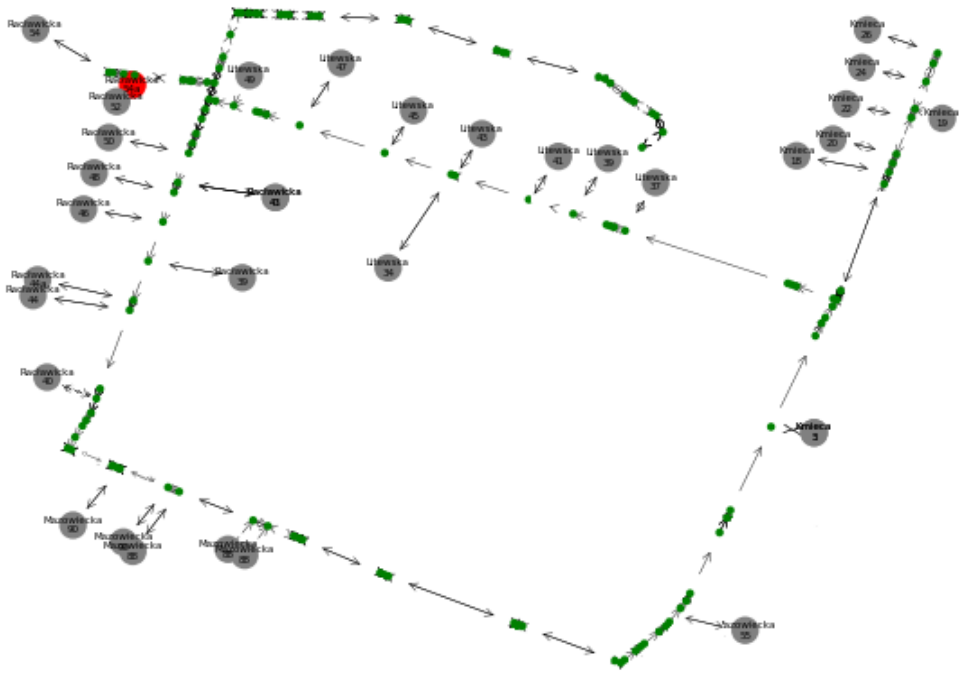
Rysunek 3: Fragment mapy Krakowa[1]

różnymi kolorami, w celu zwiększenia czytelności. Placówkę pocztową przedstawiono na czerwono, zwykłe budynki mieszkalne na szaro. Natomiast wierzchołki będące fragmentami drogi pokolorowano na zielono oraz zmniejszono ich rozmiar względem pozostałych.

Analizując kierunki dróg na rysunku 3. można zauważyć, że wpływają one na budowę grafu rzeczywistego z rysunku 4, który w przypadku kiedy listonosz porusza się np. samochodem jest MG. Dla pieszej podróży określono go jako UG.

3.3 Implementacja grafu rzeczywistego

.....



Rysunek 4: Graf rzeczywisty MG stworzony z mapy z rysunku 3

3.4 Algorytm Fleury'ego

.....

Algorytm 1: *FleuryAlgorithm* wyszukujący ścieżkę w grafie nieskierowanym

```
1 def FleuryAlgorithm(graph, startNode):
2         adjList = GraphHelper.getAdjList(graph)
3         EulerCycle = list()
4 
5         findNextNode(adjList, startNode, EulerCycle)
6         return EulerCycle
```

Algorytm 2: *findNextNode* rekurencyjna funkcja pomocnicza dla *FleuryAlgorithm*

```
1 def findNextNode(adjList, n, EulerCycle):
2         EulerCycle.append(n)
3         for v in adjList[n]:
4                 if isBridge(adjList, n, v):
5                         removeEdge(adjList, n, v)
6                         findNextNode(adjList, v, EulerCycle)
```

Algorytm 3: *findNextNode* rekurencyjna funkcja pomocnicza dla *FleuryAlgorithm*

```
def isBridge(adjList, u, v):
        if len(adjList[u]) == 1:
                return True
        else:
                listOfVisitedNeighbours = [False] * len(adjList)
                cntBeforeRestoreEdge = DFS(adjList, u, listOfVisitedNeighbours)

                removeEdge(adjList, u, v)
                listOfVisitedNeighbours = [False] * len(adjList)
                cntAfterRestoreEdge = DFS(adjList, u, listOfVisitedNeighbours)

                # restore edge
                addEdge(adjList, u, v)

                return cntBeforeRestoreEdge < cntAfterRestoreEdge
```

3.5 Algorytm Hierholzera

.....

Algorytm 1: *HierholzerAlgorithm* funkcja wyznaczająca ścieżkę dla grafów skierowanych i nieskierowanych

```
def HierholzerAlgorithm(graph, isDirected, startNode):
        adjList = GraphHelper.getAdjList(graph)
        EulerCycle = list()
        stackOfNodes = deque()

        node = startNode
        EulerCycle.append(node)

        while True:
                if checkOutDegree(adjList, node):
                        v = getNextOutEdge(adjList, node)
                        stackOfNodes.append(node)
                        removeEdge(adjList, node, v, isDirected)
                        node = v
                else:
                        node = stackOfNodes.pop()
                        EulerCycle.append(node)

                if not stackOfNodes:
                        break

        EulerCycle.reverse()
        return EulerCycle
```

4 Wyniki

.....

5 Podsumowanie

.....

Literatura

[1] M.K. Gordenko, S.M. Avdoshin *The Variants of Chinese Postman Problems and Way of Solving through Transformation into Vehicle Routing Problems.* Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, s. 221-232, 2018

[2] H. A. Eiselt, M. Gendreau, G. Laporte, *Arc Routing Problems, Part I: The Chinese Postman Problem* Institute for Operations Research and the Management Sciences (INFORMS), 1995

[3] M. Beck, D. Blado, J. Crawford, T. Jean-Louis, M. Young *On weak chromatic polynomials of mixed graphs*, Graphs and Combinatorics, 2013

[4] R. K. Ahuja, T. L. Magnanti, J. B. Orlin *Network Flows: Theory, algorithmsand applications*, Prentice Hall, New Jersey, s. 740-745, 1993

[5] J. Edmonds, E.L. Johnson *Matching Euler tours and the Chinese postman problem.* Mathematical Programming, s. 88–124, 1973

[6] Leonhard Euler *Solutio problematis ad geometriam situs pertinentis* (ang.), 1741

[7] A.-L. Barabási, R. Albert *Emergence of Scaling in Random Networks*, Science 286, 509-512, 1999

[8] K. Klemm, V. M. Eguiluz *Growing scale-free networks with small-world behavior*, Physical Review , vol. 65, 057102, 2002

[9] A. Fronczak, P. Fronczak *Świat sieci złożonych. Od fizyki do Internetu*, Wydawnictwo Naukowe PWN, 2009

[10] https://pl.python.org/ [dostęp: 23.12.2020]

[11] https://networkx.org/ [dostęp: 23.12.2020]

[12] https://matplotlib.org/ [dostęp: 23.12.2020]

[13] https://www.mathworks.com/help/matlab/ [dostęp: 23.12.2020]

[14] https://www.openstreetmap.org/ [dostęp: 23.12.2020]

[15] https://wiki.openstreetmap.org/wiki/Category:Properties
[dostęp: 20.12.2020]