AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Praca dyplomowa inżynierska

Optymalizacja trasy z wykorzystaniem algorytmu przybliżonego

Autor: Rafał Siniewicz
Kierunek studiów: Automatyka i automatyka
Opiekun: dr inż. Joanna Kwiecień

Kraków, 2019/2020r.

Spis treści

1. Wstęp

1.1. Ogólne omówienie pracy

Transport od zawsze pełnił bardzo ważną rolę w życiu społeczeństw. W przeciągu wieków zmieniały się środki transpotu, jego metody, a także szybkość przemieszaczania ludzi oraz towarów. Za każdym jednak razem wyzwanie stanowiło jak najoptymalniejsze przeprowadzenie procesu transportu. Szczególnie w dzisiejszych czasach zaplanowanie tras w taki sposób, aby zminimalizować czas ich pokonania- co wiąże się z mniejszymi kosztami oraz emisjami spalin- jest bardzo cenne. Również zwyklejsze sytuacje, jak np. zaplanowanie dojazdu do pracy (komunikacją miejską lub samochodem) wymagają od wielu osób skorzystania z odpowiednich narzędzi (świadczy o tym chociażby popularność google maps czy aplikacji jakdojade) przeszukujących wiele możliwych tras i wybierających tą najlepszą. Widać zatem jak dużą rolę odgrywa w życiu niemalże każdego dobranie odpowiedniej trasy dla danej podróży.

Problem optymalizacji trasy jest bardzo powszechny i wielokrotnie studiowany, zwłaszcza w badaniach operacyjnych, teorii grafów czy logistyce i dziedzinach pokrewnych. W teorii najczęściej dąży się do uzyskania jak najkrótszej trasy, choć problem można rozwinąć o wiele różnych czynników, które wpływają na przebieg trasy, np. jakość nawierzchni czy określona pojemność pojazdów.

Problemy optymalizacyjne rozwiązuje się przy pomocy algorytmów. Jest to kolejne ważne zagadnienie w tej pracy. Istnieje wiele różnych typów i rodzajów algorytmów, jednak z perspektywy rozwiązywania zadań optymalizacyjnych szczególnie istotna wydaje się podgrupa algorytmów zwana algorytmami przybliżonymi. Pozwalają one na bardziej praktyczne podejście do rzeczywistego problemu, tak by osiągnąć rozwiązanie zadowalające, ale niekoniecznie optymalne. Ze względu na bardzo dużą ilość obliczeń potrzebnych do wykonania większości algorytmów, a w szczególności tych mających za zadanie rozwiązanie problemów optymalizacyjnych, do ich wykonania używa się specjalnie utworzonych programów komputerowych. Podobnie i w tej pracy, w celu rozwiązania problemu i znalezienia zadowalającej trasy napisano program komputerowy, który automatyzuje cały proces oraz pozwala na jego wielokrotne użycie, a także możliwość wizualizacji.

Spora część wiedzy wykorzystanej w tej pracy pochodzi ze źródeł internetowych, książek, a także materiałów pochodzących z zajęć dotyczących matematycznych metod wspomagania decyzji oraz badań operacyjnych. Jest to wiedza ogólnie dostępna, a jej zakres został szczegółowo zbadany na przestrzeni lat w wielu pozycjach literatury naukowej.

Praca została podzielona na 7 rozdziałów- każdy zawierający kilka podrozdziałów i szerzej omówionych zagadnień. Na początku jest wstęp zawierający ogólne omówienie pracy, jej cel i zakres. Kolejny rozdział dotyczy poszukiwania optymalnej trasy w sensie ogólnym oraz przybliża najbardziej znane problemy z tej dziedziny badań naukowych. Następnie opisano metody wykorzystane przy rozwiązywaniu problemu, który porusza praca, tj. algorytmy ewolucyjne, a w szczególności ewolucyjne. W następnej kolejności omówiono wykonany projekt. Kolejny rozdział opisuje działanie aplikacji. W następnym zawarto opis przeprowadzonych testów. Na końcu pracy znajduje się rozdział podsumowujący, w którym zostały wyciągnięte wnioski dotyczące napisanej pracy oraz działania samej aplikacji.

1.2. Cel pracy

Praca ma na celu stworzenie aplikacji komputerowej oraz interfejsu graficznego napisanych przy użyciu języka programowania python. Aplikacja służy do obliczania optymalnej trasy na podstawie odpowiednich danych. Optymalna trasa jest obliczona przy pomocy odpowiednich algorytmów opisanych w poniższym wstępie teoretycznym. Program wyświetla również przebiegi tras w formie graficznej wraz z zaznaczeniem poszczególnych punktów na mapie. Całość stanowi ujednolicony i łatwy w użyciu system, w którym można ustalać parametry, takie jak, np. ilość pojazdów i ich pojemności, współrzędne geograficzne punktów, a także daje możliwość uzyskania informacji o długości trasy, jej przebiegu i innych statystykach dotyczących rozwiązywanego problemu.

1.3. Zakres pracy

W pracy zagadnienie optymalizacji trasy rozważono dla problemu marszrutyzacji, czyli VRP (Vehicle Routing Problem) oraz CVRP (Capacity Vehicle Routing Problem), a także problemu komiwojażera jako szczególny przypadek problemu marszrutyzacji z jednym pojazdem. Projekt skupia się głównie na dwóch czynnikach decydujących o optymalności: długości trasy oraz (opcjonalnie) zapełnieniu pojazdów ładunkiem.
Przy rozwiązywaniu problemu marszrutyzacji/ komiwojażera korzystano z algorytmów przybliżonych, a konkretniej algorytmu genetycznego, ze względu na fakt, że są to problemy N-P trudne. Przy rozwiązywaniu takich problemów dobrze sprawdzają się właśnie algorytmy przybliżone. Algorytm genetyczny wybrano ze względu sporą ilość opracowań na jego temat, ale również z uwagi na jego uniwersalność, możliwość uzyskania stosunkowo dobrych wyników w krótkim czasie oraz prostą koncepcję podejścia do rozwiązania.

2. Zagadnienie poszukiwania najlepszej trasy

2.1. Problem znalezienia trasy optymalnej

Optymalizacja trasy polega na znajdowaniu przy pomocy metod matematycznych najlepszego (biorąc pod uwage wybrane kryteria) rozwiązania z uwzględnieniem ściśle określonych ograniczeń. Problem poszukiwania optymalnej trasy należy do ważnych zagadnień z zakresu planowania transportu. Historycznie wywodzi się od problemu komiwojażera (Travelling Salesman Problem). Optymalizacja trasy może skupiać się na różnych czynnikach, takich jak: długość trasy, koszty, czas podróży, itp. Zazwyczaj jednak optymalizacja ma na celu znalezienie najkrótszej trasy.
Problem poszukiwania najkrótszej trasy najczęściej dotyczy wspomnianego wyżej problemu komiwojażera, ale także problemu marszrutyzacji czy innych, np. znajdowania najkrótszej ścieżki w grafie. Ze względu na to, że większość problemów z zakresu optymalizcji trasy jest NP-trudnych (w tym także problem komiwojażera oraz marszrutyzacji, które zostały opisane poniżej) rozwiązuje się je najczęściej przy pomocy algorytmów przybliżonych, tzn. takich, które nie dostarczają rozwiązania optymalnego, a jedynie zadowalające względem pewnych ustalonych kryteriów.

2.2. TSP

Na podstawie [1]:
Problem komiwojażera (Travelling Salesman Problem- TSP) – jeden z najstarszych problemów optymalizacyjnych mający na celu znalezienie minimalnego cyklu Hamiltona w n-wierzchołkowej sieci pełnej, tzn. takiej, w której każda para wierzchołków jest połączona łukiem. Jest to problem z kategorii NP- trudnych. Słowne przedstawienie problemu:
Komiwojażer ma za zadanie odwiedzić n lokalizacji. Znane są odległości (czasem także cena i czas podróży) między każdą parą lokalizacji. Celem komiwojażera jest odwiedzenie wszystkich punktów (tylko raz) w jak najkrótszym czasie/ najkrótszą drogą/ najmniejszym kosztem, zaczynając w ustalonym punkcie i do niego wracając.
Matematyczne ujęcie problemu zgodnie z [1]:
Rozpatrujemy graf nieskierowany G = (V,E), gdzie:
V - zbiór wierzchołków grafu,
E - zbiór krawędzi grafu
Oznaczmy punkty do odwiedzenia numerami od 1 do n. Dalej oznaczmy:

$$x_{ij} = \begin{cases} 1 & \text{jeśli trasa idzie z punktu i do j} \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

Przyjmijmy d_{ij} jako odległość (koszt) z punktu i do j. Wtedy problem komiwojażera może być zapisany jako problem programowania liniowego:

$$min \sum_{(i,j)\in E} d_{ij}x_{ij} \tag{1}$$

Przy ograniczeniach:

$$\sum_{j\in V} x_{ij} = 2, \quad \forall i \in V \tag{2}$$

$$\sum_{i,j\in S, i\neq j} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset \tag{3}$$

Problem komiwojażera można rozpatrywać jako:
1) symetryczny- gdzie dla dowolnych punktów X i Y odległość z X do Y jest taka sama jak z Y do X
2) asymetryczny- gdzie powyższe odległości mogą się różnić

Na podstawie notatek z wykładu nt. badań operacyjnych:
Metody rozwiązywania problemu komiwojażera:
1) metody dokładne:
- programowanie dynamiczne
- algorytm podziału i ograniczeń

2) metody przybliżone:
- algorytm najbliższego sąsiada
- algorytm wstawiania
- algorytm MTS (minimalne drzewo rozpinające)
- algorytm Lin-Kernigan
- algorytmy genetyczne

Trudność w rozwiązaniu problemu komiwojażera polega na dużej ilości danych- wszystkich możliwych kombinacji dla n punktów jest $(n-1)!/2$. Zatem już przy 15 punktach otrzymujemy ponad $43 \cdot 10^9$ (czterdzieści trzy miliardy) możliwych tras do rozpatrzenia. Rozwinięciem problemu komiwojażera jest problem marszrutyzacji.

2.3. VRP

Na podstawie [1]:
Problem marszrutyzacji (Vehicle Routing Problem- VRP) polega na wyznaczeniu optymalnych tras dla określonej liczby środków transportu. Jest on rozszerzeniem problemu komiwojażera czy problemu chińskiego listonosza. Jest to problem z kategorii NP- trudnych. Flota pojazdów ma za zadanie odwiedzenie zadanych punktów, z uwzględnieniem przyjętych ograniczeń. W ramach optymalizacji bierze się pod uwagę sumaryczny koszt pokonanych tras.
Matematyczne sformułowanie podstawowego problemu marszrutyzacji (VRP):
Rozpatrywany jest graf nieskierowany $G = (V, E)$, gdzie:
- V oznacza zbiór wierzchołków
- E oznacza zbiór krawędzi, do których przypisane są koszty przejazdu (również inne parametry, np. czas)

Zadanie polega na zminimalizowaniu funkcji celu:

$$min = \sum_{i\neq j} d_{ij} x_{ij} \tag{4}$$

Przy ograniczeniach:

$$\sum_{j} x_{ij} = 1, \quad \forall i \in V \tag{5}$$

$$\sum_i x_{ij} = 1, \quad \forall j \in V \tag{6}$$

$$\sum_i x_{ij} \geq |S| - v(S), \quad \{S : S \subseteq V \setminus \{1\}, |S| \geq 2\} \tag{7}$$

$$x_{ij} \in \{0,1\}, \quad \forall \{i,j\} \in E; \quad i \neq j \tag{8}$$

gdzie:
i, j - wierzchołki grafu między, którymi porusza się pojazd
d_{ij} - koszt przejazdu między wierzchołkadmi i oraz j
x_{ij} - zmienna przyjmująca wartości 0 lub 1 w zależności od tego czy pojazd wykonuje przejazd między wierzchołkami i oraz j (1 oznacza, że wykonuje)
$v(S)$ - odpowiednia dolna granica liczby pojazdów wymaganych do odwiedzenia wszystkich wierzchołków S dla rozwiązania optymalnego

2.4. CVRP

Na podstawie [7]:
CVRP (Capacity Vehicle Routing Problem), czyli problem marszrutyzacji z uwzględnieniem ograniczeń ładowności pojazdów jest rozwinięciem problemu komiwojażera poprzez dodanie ograniczenia w postaci określonej pojemności pojazdów. Jest najbardziej studiowanym wariantem z rodziny problemów VRP. Matematyczny model dla CVRP:

Funkcja celu:

$$min = c^T x \tag{9}$$

Przy ograniczeniach:

$$x(\delta(i)) = 2 \quad \forall i \in N \tag{10}$$

$$x(\delta(0)) = 2|K| \tag{11}$$

$$x(\delta(S)) \geq 2r(S) \tag{12}$$

$$x(\delta(S)) \geq 2r(S) \quad \forall S \subseteq N, S \neq \emptyset \tag{13}$$

$$x_e \in \{0,1,2\} \quad \forall e \in \delta(0) \tag{14}$$

$$x_e \in \{0,1\} \quad \forall e \in E \setminus \delta(0) \tag{15}$$

Dla zmiennych decyzyjnych:

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{16}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i,j \in V, \forall k \in K \tag{17}$$

gdzie:
c^T - macierz kosztów przejazdów
S - arbitralny pozbiór zbioru wszystkich wierzchołków V
N - zbiór n miejsc do odwiedzenia, N = 1,2,...,n
K - flota pojazdów, K = 1,2,...,|K|
r(S) - minimalna liczba tras pojazdu potrzebnych do obsługi S
$\delta(S) = \{\{i, j\} \in E : i \in S, j \notin S\}$ - zbiór krawędzi z dokładnie jednym (dwoma) punktem końcowym w S

Inne warianty problemu marszrutyzacji

Ponadto istnieją różne warianty problemu marszrutyzacji, na przykład:
- Vehicle Routing Problem with Time Windows- uwzględnienie okien czasowych odbioru/ wysłania towaru
- Split Delivery Vehicle Routing Problem- możliwość obsługi jednego klienta przez kilka pojazdów
- Vehicle Routing Problem with Multiple Trips (VRPMT)- możliwość pokonania więcej niż jednej trasy przez jeden pojazd
- Vehicle Routing Problem with Pickup and Delivery (VRPPD)- wiele towarów należy przetransportować z niektórych miejsc odbioru do innych miejsc dostawy
- Vehicle Routing Problem with Multiple Trips (VRPMT)- pojazdy mogą pokonywać więcej niż jedną trasę
- Open Vehicle Routing Problem (OVRP)- nie jest wymagane, aby pojazdy wracały do onego miejsca (magazynu)
- inne

Powyższy problem można również rozwinąć o wiele innych czynników, jak np. kolejność odwiedzenia miejsc czy opcjonalnego odwiedzenia niektórych punktów lub funkcja kosztu może rozpatrywać różne parametry, np. czas wykonania zleceń czy ilość przewiezionego ładunku. Widać zatem, że problem marszrutyzacji jest zagadnieniem bardzo rozległym, a przy tym elastycznym, tzn. można go dostosować do wielu różnych problemów.

Rozwiązywanie problemu VRP

Ze względu na trudność w znalezieniu optymalnych rozwiązań dla problemów związanych z trasowaniem pojazdów w dużej skali znaczny wysiłek badawczy został poświęcony metaheurystyce, takiej jak algorytmy genetyczne, wyszukiwanie Tabu czy symulowane wyżarzanie. Niektóre z najnowszych i skutecznych metaheurystyk dotyczących problemów z trasowaniem pojazdów osiągają rozwiązania w granicach 0,5% lub 1% wartości optymalnej dla przypadków problemowych liczących setki lub tysiące punktów dostawy. Metody te są również bardziej niezawodne pod takim względem, że można je łatwiej dostosować do różnych ograniczeń bocznych. Stosowanie technik metaheurystycznych jest często uprzywilejowane w przypadku aplikacji na dużą skalę ze skomplikowanymi ograniczeniami i zestawami decyzji.

Na podstawie [1]:
Metaheurystykę stanowią ogólne ramy algorytmiczne, często inspirowane naturą (np. algorytm mrówkowy), które zostały utworzone w celu rozwiązywania złożonych problemów związanych z optymalizacją. Od kilku dekad stanowią coraz większy obszar badań związanych z optymalizacją. Metaheurystyka jest skuteczną alternatywą do bardziej klasycznych podejść w rozwiązywaniu

problemów optymalizacyjnych, które zawierają w swoich wzorach matematycznych informacje nieokreślone konkretnie, stochastyczne i dynamiczne.

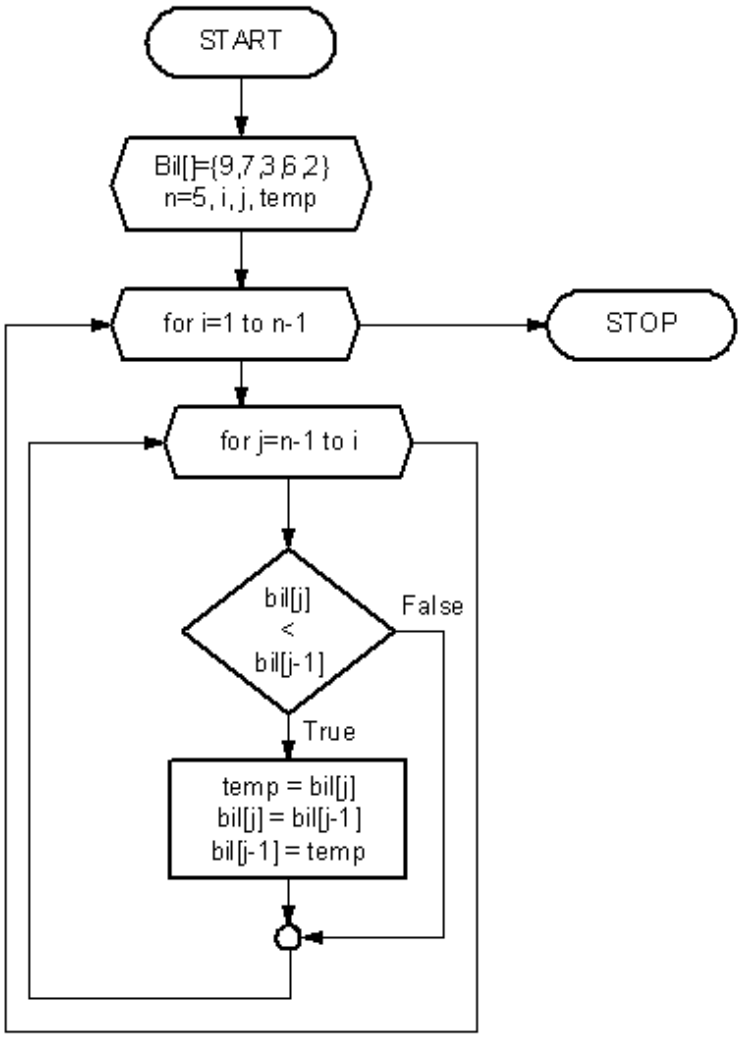
3. Algorytmy przybliżone w problemie optymalizacji trasy

Algorytm

Na podstawie [3]:

Algorytm to dowolna poprawnie zdefiniowana procedura obliczeniowa, która przyjmuje pewną wartość lub zestaw wartości jako dane wejściowe i generuje pewną wartość lub zestaw wartości jak wynik. Algorytm jest zatem sekwencją kroków obliczeniowych, które przekształcają dane wejściowe do wyjściowych.

Można także postrzegać algorytm jako narzędzie do rozwiązywania odpowiednio określonego problemu obliczeniowego. Opis problemu określa ogólnie pożądaną relację między danymi wejściowymi, a wyjściowymi. Algorytm opisuje określoną procedurę obliczeniową do osiągnięcia tej relacji wejścia / wyjścia. Klasycznym przykładem, do którego wykorzystuje się algorytmy jest procedura sortowania, np. liczb. Sposób przedstawienia algoytmu może być różny- algorytm może być zapisany jako: schemat blokowy, pseudokod, lista kroków, itp. Przykładowy diagram blokowy ilustrujący algorytm sortowania bąbelkowego:



Rysunek1. Diagram blokowy algorytmu sortującego[1]

Algorytm jest poprawny, jeśli dla każdej instancji wejściowej kończy się z poprawnym wyjściem. Mówimy, że poprawny algorytm rozwiązuje dany problem obliczeniowy. Niepoprawny algorytm może w ogóle nie zostać zatrzymany dla niektórych instancji wejściowych lub może się zakończyć z niepoprawną odpowiedzią. Czasami jednak niepoprawne algorytmy mogą okazać się przydatne, jeśli możemy kontrolować ich poziom błędów. W dalszej części zostanie opisany pewien rodzaj algorytmów nazywany algorytmami przybliżonymi.

3.1. Algorytmy przybliżone

Na podstawie [3]:
Wiele problemów o znaczeniu praktycznym jest NP-zupełnych, czyli takich, dla których nie ma rozwiązań w czasie wielomianowym. Często są one jednak zbyt ważne, aby je porzucić. Z tego powodu szuka się rozwiązań przybliżonych. Istnieją co najmniej trzy sposoby obejścia NP-zupełności:
- jeśli rzeczywiste ilości danych wejściowych są małe, algorytm z wykładniczym czasem działania może być całkowicie zadowalający
- można wyodrębnić ważne specjalne przypadki, które można rozwiązać w czasie wielomianowym
- można zaproponować metody znalezienia prawie optymalnych rozwiązań w czasie wielomianowym (w najgorszym lub oczekiwanym przypadku)

W praktyce przybliżona optymalizacja jest na ogół wystarczająca. Algorytm, który zwraca prawie optymalne rozwiązanie nazywamy **algorytmem przybliżonym (aproksymacyjnym)**.

Współczynniki wydajności dla algorytmów przybliżonych

Na podstawie [3]:

Mówimy, że algorytm dla pewnego problemu ma wskaźnik aproksymacji $\rho(n)$, jeśli dla dowolnego wejścia wielkości n koszt C rozwiązania wytworzonego przez algorytm zawiera się w ramach $\rho(n)$ współczynnika kosztu C* optymalnego rozwiązania:

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n) \tag{18}$$

Jeśli algorytm osiąga wskaźnik aproksymacji $\rho(n)$, nazywamy go algorytmem $\rho(n)$- aproksymacyjnym. Definicja wskaźnika aproksymacji algorytmu $\rho(n)$- aproksymacyjnego stosuje się zarówno dla problemów, w których jest minimalizcja jak i maxymalizacja funkcji celu.
Dla maxymalizacji jest: $0 \leq C \leq C^*$ oraz wskaźnik C^*/C daje czynnik mówiący o ile koszt rozwiązania optymalnego jest większy od kosztu rozwiązania przybliżonego.
Podobnie dla minimalizacji jest: $0 \leq C^* \leq C$ oraz wskaźnik C/C^* daje czynnik mówiący o ile koszt rozwiązania optymalnego jest większy od kosztu rozwiązania przybliżonego.
Ponieważ zakładamy, że wszystkie rozwiązania mają dodatni koszt, współczynniki te są dobrze zdefiniowane. Wskaźnik aproksymacji algorytmu przybliżonego jest nie mniejszy niż 1, skoro $C/C^* \leq 1$ implikuje $C/C^* \geq 1$.
Stąd algorytm przybliżony o wskaźniku aproksymacji wynoszącym 1 daje optymalne rozwiązanie, natomiast algorytm przybliżony o dużym wskaźniku aproksymacji może zwrócić rozwiązanie, które jest o wiele gorsze od optymalnego.

3.2. Algorytmy ewolucyjne

Na podstawie [3]:
Algorytm ewolucyjny jest to algorytm metaheurytycznej optymalizacji, należący do grupy algorytmów przybliżonych, który bazuje na populacji osobników, tworzących rozwiązania. Algorytm ewolucyjny jest inspirowany mechanizmami ewolucji biologicznej, jak np. mutacja, reprodukcja, selekcja. Potencjalne rozwiązania w tym algorytmie pełnią rolę jednostek w populacji. Odpowiednio dobrana funkcja celu określa jakość tych rozwiązań. Ewolucja populacji następuje po wielokrotnym zastosowaniu powyżej wspomnianych operatorów ewolucji. Algorytmy ewolucyjne często dobrze sprawdzają się do znajdowania przybliżonych rozwiązań dla problemów różnych typów, ponieważ nie czynią na początku żadnych założeń dotyczących skłaniania się ku potencjalnym "dobrym" rozwiązaniom. Są więc bardzo uniwersalne. Algorytmami podobnymi do ewolucyjnych są algorytmy rojowe, np. algorytm mrówkowy czy algorytm pszczeli.

Algorytm ewolucyjny w krokach:
1. Generacja losowa początkowej populacji osobników (pierwsza generacja).
2. Obliczenie funkcji celu dla każdego osobnika w populacji.
3. Powtarzanie następujących kroków aż do osiągniecią odpowiedniej wartości funkcji celu lub maksymalnej ilości iteracji:
    - wybierz najlepiej dopasowane osobniki do reprodukcji (rodzice)
    - rozmnażanie osobników poprzez krzyżowanie oraz mutacje, w celu wydania nowego potomstwa (rozwiązań)
    - oblicz funkcję celu dla nowo powstałych osobników
    - pozostawienie jedynie najlepszych osobników (rozwiązań)

3.3. Algorytm genetyczny

Na podstawie [3]:
Termin algorytm genetyczny po raz pierwszy wprowadził John Holland, bazując na koncepcji Darwina- teorii ewolucji. Jest to metaheurystyka (metoda poszukiwania rozwiązań, która nie gwarantuje znalezienia optymalnego rozwiązania) bazująca na zjawisku naturalnej selekcji (teorii ewolucji gatunków), a także dziedziczności. Należy do szerszej grupy- algorytmów ewolucyjnych. Algorytm ten polega na znajdowaniu rozwiązania naśladując zjawiska występujące w środowisku naturalnym, takie jak: mutacje, krzyżowania gatunków, a także selekcja. Z założenia w algorytmach genetycznych występuje element losowości, jednak fakt umiejętnego wykorzystania przeszłego doświadczenia w celu stworzenia nowych rozwiązań sprawia, że algorytm ten jest daleki od przypadkowego błądzenia. Taka metoda zyskuje na popularności ze względu na ograniczenia metod analitycznych.

Metodologia i sposób działania

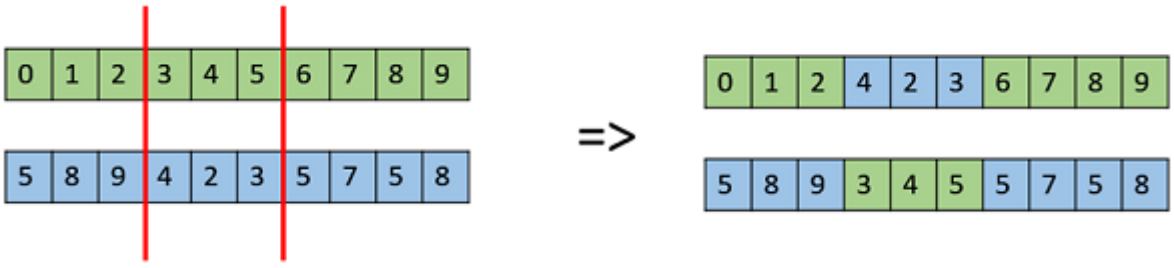
Ogólną koncepcją algorytmu genetycznego jest ewolucja populacji rozwiązań danego problemu optymalizacyjnego w kierunku coraz lepszych wartości funkcji celu. Każda jednostka w populacji posiada zbió właściwości (swoje chromosomy lub genotyp), który w ramach algorytmu zostaje zmieniony. Algorytm zazwyczaj rozpoczyna ewolucje bazując na losowo wygenerowanej populacji rozwiązań. Każdy cykl ewolucyjny tworzy nowe pokolenie. W każdym pokoleniu jest obliczana wartość funkcji celu dla każdego rozwiązania. Standardowo rozwiązanie prezentuje się jako tablica bitów. Poniżej opisano poszczególne kroki w AG:

Inicjalizacja populacji

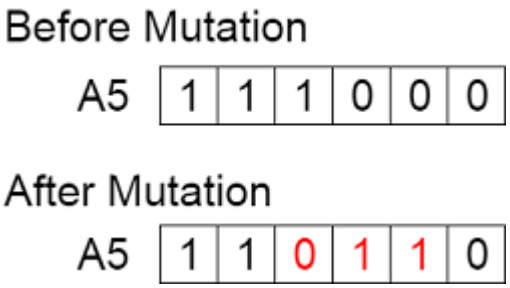
Rozmiar populacji początkowej może byc różny- zazwyczaj zawiera kilkaset lub kilka tysięcy osobników. Populacja początkowa jest generowana losowo, co powoduje, że rozwiązania nie są z góry "nakierowane" na pewien obszar.

Operatory genetyczne

Najczęściej operatorami genetycznymi są mutacja oraz krzyżowanie. Dla każdego nowego rozwiązania są wybierane dwa z poprzedniej populacji (rodzice). Dzięki operatorowi krzyżowania "produkują" oni nowe rozwiązanie (dziecko), które posiada zestaw cech odziedziczonych po rodzicach. Proces jest powtarzany w każdej iteracji. Dla jednej populacji ilość krzyżowań może być różna, zazwyczaj wynosi od kilkudziesięciu do kilkuset.



Rysunek1. Przykład krzyżowania [3]



Rysunek1. Przykład mutacji [3]

Selekcja

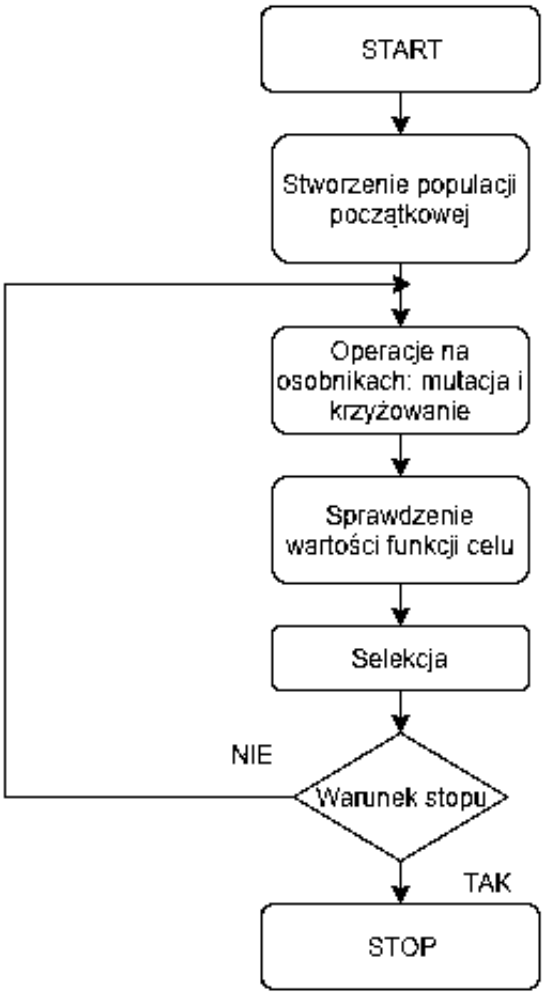
Proces selekcji jest dokonywany na podstawie funkcji celu: osobniki, które są najlepiej dostosowane, tzn. mają najwyższą/ najniższą wartość funkcji celu (w zależności od problemu) zostają dopuszczone do kolejnej iteracji- w ten sposób całość ewoluuje do coraz lepszych rozwiązań- dziecko, które otrzymało od rodziców najlepsze geny ma największą szanse "przetrwania".

Zakończenie

Przerwanie działania algorytmu może nastąpić w różnych sytuacjach, w zależności od sformułowania algorytmu:
- uzyskano rozwiązanie, które spełnia minimalne kryteria
- wykonała się ustalona ilość iteracji
- manualne przerwanie algorytmu
- inne, np. kombinacja powyższych

Przedstawienie algorytmu

Algorytm genetyczny w krokach:
1. Utworzenie początkowej populacji
2. Przeprowadzanie mutacji i krzyżowania
3. Sprawdzenie funkcji celu dla osobników
4. Wybór najlepszych osobników poprzez selekcję
5. Powtarzanie kroków 2-4 aż do spełnienia warunku stopu
6. Koniec algorytmu i wybór najlepszego osobnika



Rysunek1. Schemat algorytmu genetycznego

Różnice między algorytmem genetycznym (AG), a "tradycyjnymi" metodami

1. AG nie działają na dokładnych parametrach problemu, a na ich zakodowanej formie.
2. AG poszukują rozwiązań rozpoczynając nie od pojedynczego punktu, ale od pewnej grupy (populacji) rozwiązań.
3. W AG korzysta się jedynie z funkcji celu, a nie na jej pochodnych.
4. W AG wyboru dokonuje się na podstawie probabilistycznych reguł, a nie deterministycznych.

Zastosowania algorytmów genetycznych

Algorytmy genetyczne mają spore zastosowanie w wielu dziedzinach, m.in. matematyce, technice, medycynie. Poniżej zostały przedstawione niektóre z nich:

Problemy NP-trudne

Oczywistym zastosowaniem algorytmów genetycznych (bazujących na metaheurystyce) jest rozwiązywanie problemów, dla których nie da się znaleźć optymalnego wyniku w czasie wielomianowym. Przykładowym zagadnieniem z tej kategorii może być, np. problem komiwojażera. Zastosowanie AG pozwala otrzymać całkiem dobre rozwiązania w niedługim czasie- nie gwarantuje nam jednak znalezienia rozwiązania optymalnego. AG dobrze nadają się również do wyszukiwania przybliżeń ekstremów funkcji, których nie można znaleźć analitycznie.

Harmonogramy

Algorytmy genetyczne dobrze nadają się do ustalania różnego rodzaju harmonogramów, jak np. plany zajęć, godziny pracy, procesy produkcyjne. Pozwala to na optymalne rozplanowanie czasu z uwzględnieniem odpowiednich parametrów, np. odpoczynek i sen dla zawodowych kierowców.

Obwody elektryczne

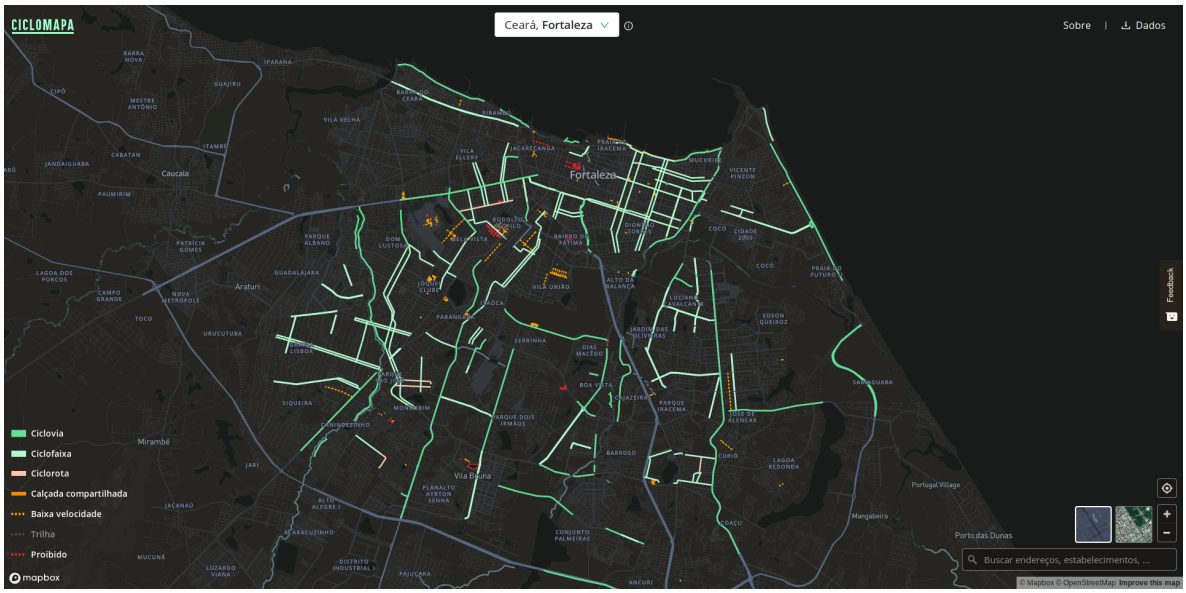
AG są wykorzystywane także do projektowania obwodów elektrycznych. Jest to problem bardzo praktyczny, a inżynierowie do zaprojektowania odpowiedniego obwodu nie koniecznie potrzebują najlepszego możliwego rozwiązania- w takich sytuacjach AG jest bardzo wydajn. Ze względu na fakt, że AG nie bazuje na wcześniejszych rozwiązaniach i schematach może okazać się źródłem nowych pomysłów w budowaniu obwodu elektrycznego.

4. Opis projektu

4.1. Użyte technologie

Na podstawie [4]:
Python- jest językiem programowania ogólnego przeznaczenia. Jest to język wysokiego poziomu. Może być skutecznie wykorzystywany przy budowie w zasadzie każdego programu. Nie potrzebuje bezpośredniego dostępu do sprzętu komputerowego. Python nie jest optymalny dla programów, które mają duże ograniczenia niezawodności lub są zbudowane i utrzymywane przez wiele osób lub przez długi czas. Ma on jednak kilka zalet w porównaniu z innymi językami programowania, m.in.:
- jest łatwy w nauce
- posiada bardzo dużą liczbę darmowych i powszechnie dostępnych bibliotek, zapewniających rozszerzoną funkcjonalność

Na podstawie [5]:
OSM (OpenStreetMap)- jest to darmowa, edytowalna mapa całego świata, tworzona przez wolontariuszy, wydawana na licencji open-content. Licencja OpenStreetMap pozwala na bezpłatny dostęp do obrazów i podstawowych danych map. Projekt ma na celu promowanie nowych zastosowań tych danych. W pracy OSM zostało wykorzystane do zwizualizowania przebiegów tras i pokazania ich na rzeczywistej mapie. Przykład obrazu uzyskanego dzięki OSM [6]:



Rysunek2. Przykładowy screen z osm

Poniżej przedstawiono krótki opis najważniejszych bibliotek użytych do stworzenia programu w języku python:

Na podstawie [5]:
PyQt- jest to zbiór bibliotek pythona stworzonych przez Riverbank Computing. Służą do tworzenia interfejsów graficznych aplikacji w ramach międzyplatformowego frameworka Qt,

który jest open-sourcowy. Możliwa w użyciu na wielu systemach opercyjnych i platformach. W programie została użyta w celu zrobienia GUI oraz wyświetlania w nim mapy.

Na podstawie [5]:
Folium- w znacznym stopniu opiera się na bibliotece leaflet.js służącej do mapowania, tworzenia interaktywnych map w javascript. Folium pozwala na przetwarzanie danych geograficznych oraz ich wizualizację w pythonie. W aplikacji za pomocą tej biblioteki tworzone są mapy w formacie html, które następnie są wyświetlane w gui stworzonym przy pomocy PyQt.

Na podstawie [5]:
Geopy- biblioteka pozwalająca na łatwą lokalizację współrzędnych geograficznych na podstawie adresów oraz obliczania odległości na globie między dwoma zbiorami współrzednych (długość i szerokość geograficzna). W pracy użyta w celu obliczenia odległości na globie dla zestawu danych.

Źródła

[1] Sysło M.M., Deo N., Kowalik J.S., Algorytmy optymalizacji dyskretnej, wyd. drugie, Warszawa: Wydawnictwo Naukowe PWN, 1995, ISBN 83-01-11818-0

[2] http://examples.gurobi.com/traveling-salesman-problem/

[3] https://www.researchgate.net/publication/313005083_Vehicle_routing_problem_Models_and_solutions

[4] https://www.researchgate.net/publication/284529903_Chapter_1_The_Family_of_Vehicle_Routing_Problems

[5] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Walter J. Gutjahr (2009). "A survey on metaheuristics for stochastic combinatorial optimization"

[6] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Clifford Stein (2009). Introduction To Algorithms (3rd ed.). MIT Press. ISBN 978-0-262-03384-8.

[7] https://javarevisited.blogspot.com/2014/08/bubble-sort-algorithm-in-java-with.html

[8] Vikhar, P. A. "Evolutionary algorithms: A critical review and its future prospects". Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). Jalgaon, 2016, pp. 261-265. ISBN 978-1-5090-0467-6.

[9] D. E. Goldberg: Algorytmy genetyczne i ich zastosowania. Warszawa: WNT, 1998. (pol.)

[10] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

[11] https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[12] Guttag, John V. (12.08.2016). Introduction to Computation and Programming Using Python: With Application to Understanding Data. MIT Press. ISBN 978-0-262-52962-4.

[13] https://wiki.openstreetmap.org/wiki/About_OpenStreetMap

[14] https://wiki.openstreetmap.org/wiki/File:Screenshot_2019-10-23_CicloMapa.png

[15] https://python101.readthedocs.io/pl/latest/pyqt/

[16] https://python-visualization.github.io/folium/