Jacek Jaszczuk 218320

Algorytm mrówkowy w rozwiązaniu problemu komiwojażera

Metody sztucznej inteligencji

1. Cel projektu

Celem projektu było rozwiązanie problemu komiwojażera przy użyciu algorytmu mrówkowego.

2. Problem komiwojażera

Problem komiwojażera (ang. *travelling salesman problem*, TSP), jest typowym problemem optymalizacyjnym który polega na znalezieniu minimalnego drogi w grafie, która będzie przechodziła cykl stworzony ze wszystkich wierzchołków grafu. Każdy wierzchołek może być odwiedzony tylko raz.

Problem komiwojażera często jest przedstawiany za pomocą historyjki o obwoźnym sprzedawcy. Mamy zbiór miast. Komiwojażer zaczyna swą podróż w mieście rodzinnym. Aby zaoszczędzić na paliwie musi wyznaczyć najkrótszą drogę przez wszystkie miasta a na końcu wrócić jeszcze do domu.

Problem komiwojażera można rozwiązać na kilka sposobów – jednym z nich jest zastosowanie algorytm mrówkowego. Wszystkich możliwości ułożenia miast jest n!, gdzie n to liczba miast, tak więc złożoność algorytmu przeglądu zupełnego to O(n!). Jest to dosyć duża złożoność obliczeniowa dlatego problem próbowano rozwiązać szybciej za pomocą innych algorytmów np.: algorytmu mrówkowego.

Problem komiwojażera jest zaliczany do problemów NP-zupełnych, czyli takich dla których znalezienie rozwiązania w czasie wielomianowym nie jest możliwe.

Poniżej znajduje się tabela przedstawiająca liczbę możliwych rozwiązań w zależności od liczby wierzchołków grafu oraz teoretycznego czasu na znalezienie najlepszego rozwiązania. Dla liczby miast równej 10 ustalono czas rozwiązywania problemu przez algorytm przeglądu całkowitego na 1 sekundę. Jest to wartość przypadkowa i służy tylko do pokazania jak ten czas się zmienia w zależności od liczby miast:

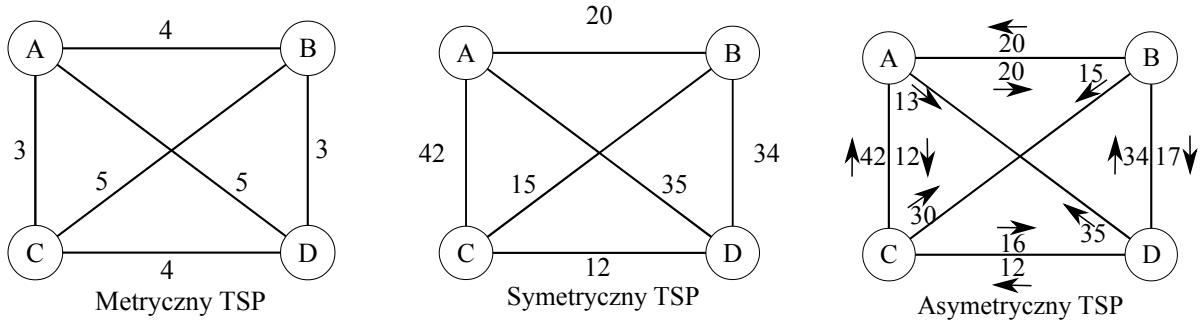
Liczba rozwiązań	Miasta	Czas w sekundach	Czas w dniach	Czas w latach
6	3	0,0000	0,0000	0,0000
24	4	0,0000	0,0000	0,0000
120	5	0,0000	0,0000	0,0000
720	6	0,0002	0,0000	0,0000
5 040	7	0,0014	0,0000	0,0000
40 320	8	0,0111	0,0000	0,0000
362 880	9	0,1000	0,0000	0,0000
3 628 800	10	1,0000	0,0000	0,0000
39 916 800	11	11,0000	0,0001	0,0000
479 001 600	12	132,0000	0,0015	0,0000
6 227 020 800	13	1 716,0000	0,0199	0,0001
87 178 291 200	14	24 024,0000	0,2781	0,0008
1 307 674 368 000	15	360 360,0000	4,1708	0,0114
20 922 789 888 000	16	5 765 760,0000	66,7333	0,1828
355 687 428 096 000	17	98 017 920,0000	1 134,4667	3,1081
6 402 373 705 728 000	18	1 764 322 560,0000	20 420,4000	55,9463

Problem komiwojażera można podzielić na 3 rodzaje:

a) Metryczny problem komiwojażera, w którym krawędzie grafu spełniają nierówność trójkąta,

b) Symetryczny problem komiwojażera, w którym droga z wierzchołka pierwszego do drugiego jest taka sama jak z drugiego do pierwszego,

c) Asymetryczny problem komiwojażera, w którym droga z wierzchołka pierwszego do drugiego nie musi być taka sama jak z drugiego do pierwszego.



3. Algorytm mrówkowy

Algorytm mrówkowy (ang. Ant Colony Optimization, ACO) został zaproponowany i przedstawiony w 1992 roku przez Marco Dorigo. Jest on przykładem zastosowania techniki probabilistycznej do szukania dobrych ścieżek w grafie.

Algorytm mrówkowy jest przykładem inteligencji roju.

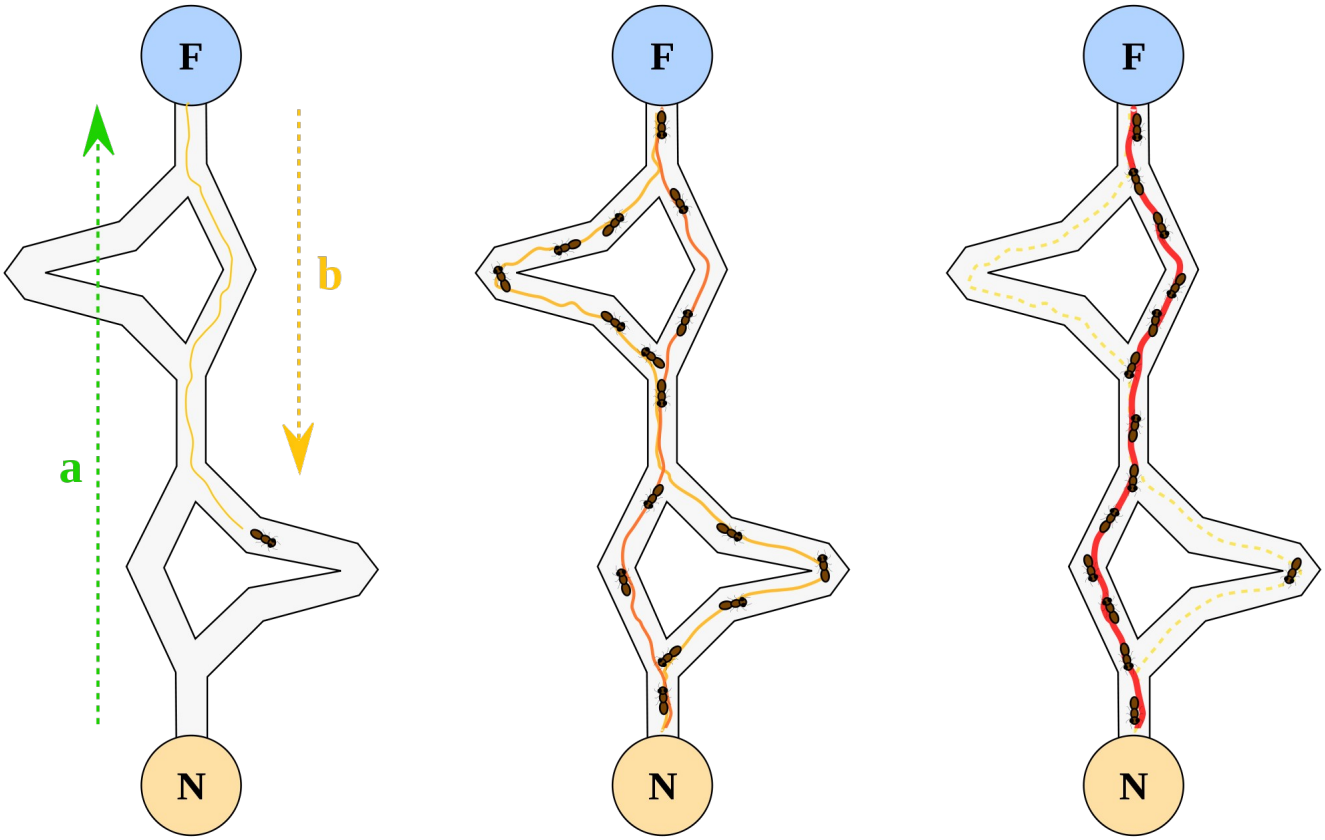
Swoje działanie opiera o zachowanie przyrody, w tym przypadku mrówek które szukają pożywienia. Każda mrówka wyruszająca z mrowiska poszukuje pokarmu. Kiedy ten pokarm znajdzie, to wraca do mrowiska zostawiając po drodze ślad feromonowy. Jeżeli inna mrówka natknie się na ten ślad, to przestaje poruszać się w sposób losowy i podąża za śladem feromonowym w stronę pokarmu. Feromon na długiej lub ślepej ścieżce do mrowiska ulega stopniowemu wyparowaniu, i ścieżka zostaje zapomniana. W końcowym rozrachunku, mrówki korzystają tylko ze ścieżek o najsilniejszym śladzie feromonowym, które przy okazji są najkrótszymi ścieżkami do mrowiska.

Na poniższej ilustracji przedstawiono działanie zachowania mrówek.

W punkcie 1 mrówka znalazła pożywienie F i rozpoczęła powrót do mrowiska N zostawiając po drodze ślad feromonowy.

W punkcie 2 inne mrówki, podążają po pokarm. Jeżeli natkną się na feromon, to będą bardziej preferować tę ścieżkę, ale to nie oznacza, że nie będą szukać alternatywnych dróg. Dzięki przeszukiwaniu alternatywnych ścieżek, utrwala się ślad feromonowy na najkrótszej z nich.

W punkcie 3 ślad feromonowy został tylko na najkrótszej ścieżce, wyparowując z ścieżek dłuższych. Większość mrówek porusza się najkrótszą ścieżką, tylko nie liczne szukają, alternatywnej drogi.



4. Metodologia rozwiązania

Algorytm mrówkowy został napisany w języku programowania Rust. Jest to stosunkowo młody język programowania, którego głównym zadaniem jest budowanie niezawodnego i wydajnego oprogramowania. Więc informacji o języku programowania Rust można uzyskać na stronie: https://www.rust-lang.org/.

Funkcja wykonująca algorytm mrówkowy ma następującą definicję:

```
fn algorytm_mrowkowy(
graf: & Graf<usize>,                // Referencja na graf do przejścia.
liczba_iteracji: usize,             // Liczba iteracji algorytmu.
waga_losowosci: f32,                // Waga losowości z przedziału 0.0 do 1.0
                                    // waga feromonów liczona jest jako
                                    // 1.0 - waga losowości.
zostawiany_feromon:f32,             // Ilość zostawianego feromonu.
ulatnianie_feromonu: f32            // Współczynnik zostawianego feromonu.
)
```

Algorytm mrówkowy dla problemu komiwojażera został zaimplementowany w następujący sposób:

a) Do każdej ścieżki przypisana jest początkowa liczba feromonów wynosząca 0.0.

b) Z każdego miasta wyrusza jedna mrówka,

c) Mrówka wybiera najlepszą według niej ścieżkę. Ścieżka ta musi prowadzić do jeszcze nieodwiedzonego miasta. Mrówka wybiera ścieżkę na podstawie losowej liczby z przedziału 0.0 do 1.0 oraz tablicy feromonów. Wagi z jaką będą brane pod uwagę liczby losowe i feromony ustala się w parametrze funkcji `waga_losowosci`. Im waga losowości większa, tym mrówki będą mniej uwagi zwracały na feromony.

d) Po wybraniu miasta docelowego przez mrówkę, mrówka przechodzi tam zostawiając swój ślad feromonowy o wartości `zostawiany_feromon` podzielny przez długość ścieżki. Wielkość zostawionego feromonu nie może być większa niż 1.0. Jeżeli jest większa, to wynik jest przycinany do 1.0.

e) W ten sposób każda z mrówek wykonuje swój krok.

f) Po tym jak każda z mrówek wykona swój krok, jest wykonywane odparowywanie feromonów o wartość `ulatnianie_feromonu`. Ilość feromonów na ścieżce nie może spaść poniżej 0.0. Jeżeli jest mniej to wynik jest przycinany do wartości 0.0.

g) Jeżeli mrówki nie przeszły po wszystkich miastach to idź do kroku c).

h) Jeżeli mrówki przeszyły po wszystkich miastach to sprawdź czy ich rozwiązania są lepsze od obecnego. Jeżeli są lepsze to zapamiętaj je. Jeżeli nie osiągnięto wymaganej liczby iteracji `liczba_iteracji` algorytmu to idź do punktu b).

i) Algorytm skończony, mamy najlepsze rozwiązanie.

5. Wyniki

Do testów wylosowane trzy grafy, przy użyciu następujących komend:

```
a) algorytm_mrowkowy wylosuj 15 30 90
b) algorytm_mrowkowy wylosuj 25 30 90
c) algorytm_mrowkowy wylosuj 60 30 90
```

Komendy powodują wylosowanie grafów o wielkości 15, 25, 60 z rozpiętościami ścieżek o rozkładzie jednorodnym od 30 do 90.

Dla każdego grafu wykonano jeden przebieg algorytmu zachłannego, oraz po 5 przebiegów algorytmu mrówkowego i losowego.

Komenda dla uruchomienia algorytmu zachłannego była następująca:

```
algorytm_mrowkowy zachlanny nazwa_grafu
```

Komenda dla algorytmu mrówkowego:

```
algorytm_mrowkowy mrowkowy -n nazwa_grafu -i 200 -u poziom_ulatniania -l
waga_losowości -z zostawiany_feromon
```

Co oznacza, że algorytm mrówkowy miał wykonać 200 iteracji, ilość ulatnianego feromonu na poziomie `poziom_ulatniania`, waga losowości na poziomie `waga_losowości` oraz ilość zostawianego feromonu na poziomie `zostawiany_feromon`. Parametry należy dobrać indywidualnie dla każdego grafu, tak aby w macierzy feromonów, wykrystalizowała się jedna silna ścieżka.

Komenda dla algorytmu losowego wyglądała następująco:

```
algorytm_mrowkowy mrowkowy -n nazwa_grafu -i 200 -u 0.0 -l 1.0 -z 0.0
```

Uruchomienie algorytmu mrówkowego z zadanymi parametrami spowoduje, że mrówki nie będą korzystać z feromonów i będą przemierzać graf w sposób losowy.

Wyniki zostały zawarte w poniższych tabelach:

Liczba miast	**Algorytm zachłanny**
15	588
25	885
60	1971

Graf 15 miast Poziom ulatniania: 0.003, Waga losowości: 0.05, Zostawiany feromon: 9.65		
Numer próby	**Algorytm mrówkowy**	**Algorytm losowy**
1	577	669
2	555	706
3	553	636
4	556	635
5	556	671
Średnia	559,4	663,4

Graf 25 miast Poziom ulatniania: 0.005, Waga losowości: 0.1, Zostawiany feromon: 10.65		
Numer próby	**Algorytm mrówkowy**	**Algorytm losowy**
1	913	1159
2	871	1138
3	929	1162
4	935	1172
5	945	1202
Średnia	918,2	1166,6

Graf 60 miast Poziom ulatniania: 0.007, Waga losowości: 0.15, Zostawiany feromon: 11.65		
Numer próby	**Algorytm mrówkowy**	**Algorytm losowy**
1	2141	3006
2	2203	3086
3	2182	3025
4	2198	3083
5	2221	3040
Średnia	2189	3048

6. Wnioski

Jak widać, algorytm mrówkowy działa. Dla grafu z 15 miastami udało się uzyskać wynik lepszy od algorytmu zachłannego. Natomiast dla grafów 25 miast, oraz 60 miast wynik nie był lepszy, ale był przybliżony.

Porównując algorytm mrówkowy do algorytmu losowego, ten pierwszy okazał się zawsze lepszy, co nie dziwi.

Wydaje się, że podstawową wadą algorytmu mrówkowego, jest trudność w doborze prawidłowych wartości parametrów, których wartość zależą między innymi od wielkości grafu.

Myślę, że dobór lepszych parametrów mógłby sprawić, że w miastach 25 i 60 algorytm mrówkowy miałby lepszy wynik od algorytmu zachłannego.