# Zasada działania algorytmu mrówkowego

Zasadę działania algorytmu mrówkowego można przedstawić za pomocą następujących kroków:

1. Tworzenie początkowych śladów feromonowych – początkowa wartość feromonu dla każdej ścieżki wynosi 1.
2. Generowanie kolonii początkowej – na początku działania algorytmu generowane są mrówki, które przydzielane mają losowo atrakcje startowe.
3. Wybór następnej lokacji – nowe atrakcje są wybierane do czasu jednokrotnego odwiedzenia wszystkich atrakcji.
   1. Wybór probabilistyczny – wybierane jest droga, której wartość prawdopodobieństwa wyboru obliczanego ze wzoru poniżej jest największa.
   2. Selekcja ruletkowa – losowy wybór następnej atrakcji.
4. Aktualizacja śladów feromonowych – uwzględniany jest współczynnik wyparowywania oraz feromony pozostawione przez mrówki.
5. Aktualizacja najlepszego rozwiązania – sprawdzenie czy aktualne najlepsze rozwiązanie jest lepsze od ogólnego najlepszego rozwiązania.

* Wyżej wymieniane kroki są powtarzane do momentu zrealizowania jednego warunku stopu: liczba iteracji.

# Wybrane miejsca implementacji

Nasz program implementujący algorytm mrówkowy przyjmuje następujące wartości:

* Liczbę iteracji programu
* Współczynnik wyparowywania feromonu
* Liczba mrówek w kolonii
* Współczynnik alfa
* Współczynnik beta
* Tablice dystansów pomiędzy atrakcjami

Funkcja ant\_colony () przyjmuje powyższe wartości. Pierwszym etapem działania programu jest wygenerowanie tablicy śladów feromonowych oraz rozpoczęcie pętli, która wykonuje się określona liczbę iteracji, w której pierwszym etapem jest stworzenie początkowej kolonii mrówek.

def ant\_colony(iterations, evaporation\_rate, num\_ants, num\_attractions, alpha, beta, distances):  
 pheromone = initialize\_pheromone\_matrix(num\_attractions)  
 best\_ant = None  
  
 for i in range(iterations):  
 colony = create\_colony(num\_ants, num\_attractions)

def initialize\_pheromone\_matrix(size):  
 return np.ones((size, size))  
  
def create\_colony(num\_ants, num\_attractions):  
 return [Ant(num\_attractions) for \_ in range(num\_ants)]

Mrówki są obiektami klasy Ant posiadającej dwa atrybuty:

* memory – pamięć odwiedzonych atrakcji
* total\_distance – długość pokonanej trasy

oraz jedna metodę do aktualizowania przebytego dystansu update\_distance().

class Ant:  
 def \_\_init\_\_(self, num\_attractions):  
 self.memory = [random.randint(0, num\_attractions - 1)]  
 self.total\_distance = 0.0  
  
 def update\_distance(self, distances):  
 self.total\_distance = sum(  
 distances[self.memory[i], self.memory[i + 1]]  
 for i in range(len(self.memory) - 1)  
 )

Następnym krokiem jest rozpoczęcie iteracji po wszystkich mrówkach w kolonii oraz atrakcjach. Podczas iteracji sprawdzane jest prawdopodobieństwo losowego wyboru atrakcji.

for ant in colony:  
 for \_ in range(num\_attractions - 1):  
 probabilities = calculate\_probabilities(ant, pheromone, alpha, beta, distances)  
 rand = random.uniform(0, 1)  
 if rand <= 0.1:  
 next\_attraction = selection(probabilities)  
 ant.memory.append(next\_attraction)  
 else:  
 next\_attraction = max(probabilities, key=lambda x: x[1])  
 ant.memory.append(next\_attraction[0])

Funkcja calculate\_ probabilities () oblicza wartość prawdopodobieństwa wyboru danej atrakcji na podstawie intensywności feromonów oraz heurystyki.

def calculate\_probabilities(ant, pheromone, alpha, beta, distances):  
 current = ant.memory[-1]  
 unvisited = set(range(len(attractions\_list))) - set(ant.memory)  
 probabilities = []  
  
 for attraction in unvisited:  
 pheromone\_strength = pheromone[current][attraction] \*\* alpha  
 heuristic\_value = (1 / distances[current][attraction]) \*\* beta  
 probabilities.append((attraction, pheromone\_strength \* heuristic\_value))  
  
 total = sum(prob[1] for prob in probabilities)  
 probabilities = [(attraction, prob / total) for attraction, prob in probabilities]  
 return probabilities

Funkcja selection() pozwala na losowy wybór następnej atrakcji poprzez selekcje ruletkową

def selection(probabilities):  
 r = random.uniform(0, 1)  
 cumulative = 0  
 for attraction, prob in probabilities:  
 cumulative += prob  
 if r <= cumulative:  
 return attraction  
 return probabilities[-1][0]

Kolejnym etapem jest aktualizacja tabeli śladów feromonowych oraz sprawdzenie czy aktualna najlepsza trasa jest lepsza od ogólnej najlepszej trasy

update\_pheromones(pheromone, colony, evaporation\_rate)  
best\_ant = best(colony, best\_ant)

def update\_pheromones(pheromone, colony, evaporation\_rate):  
 pheromone \*= evaporation\_rate  
 for x in range (len(attractions\_list)):  
 for y in range (len(attractions\_list)):  
 for ant in colony:  
 pheromone[x][y] += 1 / ant.total\_distance

def best(colony, prevBestAnt):  
 bestAnt = prevBestAnt  
 tempBestAnt = min(colony, key=lambda ant: ant.total\_distance)  
 if tempBestAnt.total\_distance < bestAnt.total\_distance:  
 bestAnt = tempBestAnt

Po osiągnięciu warunku stopu zwracane jest najlepsza mrówka

return bestAnt

# Zasada działania programu

Zadaniem programu jest znalezienie najkrótszej trasy pomiędzy określonymi atrakcjami wczytanymi z pliku o określonej strukturze nrAtrakcji współrzędnaX współrzędnaY

Użytkownik wybiera plik z lokalizacjami atrakcji poprzez podanie nazwy pliku. Następnie program wywołuje funkcję, w której użytkownik podaje sześć parametrów: liczbę iteracji programu, współczynnik wyparowywania feromonu, liczba mrówek w kolonii, współczynnik alfa, współczynnik beta. Zadaniem funkcji jest przejście przez określoną liczbę iteracji, generując nowe kolonie mrówek, które poruszają się między atrakcjami szukając najkrótszej trasy. Program po spełnieniu warunku stopu kończy swoje działanie i wyświetla na ekranie długość najkrótszej trasy oraz wykres pokazujący tą trasę.

# Przeprowadzone eksperymenty

Algorytm dla wszystkich zestawów parametrów wejściowych zostanie uruchomiony dokładnie 5 razy, przy czym wyniki oraz średnie wyniki zostaną zaprezentowane w tabeli uruchamiając algorytm mrówkowy kolejno dla zestawów danych z 32 atrakcjami oraz 80 atrakcjami. W tabeli zawarte zostały także wartości ‘min’ oraz ‘max’, które odpowiadają odpowiednio najgorszym i najlepszym rozwiązaniom znalezionym dla danej ilości atrakcji.

Parametry wejściowe:

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 474,19 | 488,34 | 489,80 | 486,21 | 486,32 | 484,97 | 474,19 | 489,80 |
| 80 | 837,02 | 837,94 | 806,22 | 842,43 | 852.40 | 835,20 | 806,22 | 842,43 |

Obraz zawierający diagram, linia, mapa, tekst

Opis wygenerowany automatycznieObraz zawierający diagram, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

Większa ilość mrówek:

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 100
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 481,41 | 473,01 | 476,06 | 468,74 | 466,90 | 473,22 | 466,90 | 481,41 |
| 80 | 832,55 | 826,92 | 830,19 | 837,20 | 806,90 | 826,75 | 806,90 | 837,20 |

Obraz zawierający diagram, linia, mapa, tekst

Opis wygenerowany automatycznieObraz zawierający diagram, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

Mniejsza ilość mrówek:

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 25
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 482,32 | 490,45 | 472,75 | 466,78 | 489,42 | 480,42 | 466,78 | 490,45 |
| 80 | 859,07 | 826,15 | 833,63 | 822,32 | 859,61 | 840,15 | 822,32 | 859,61 |

Obraz zawierający diagram, linia, mapa, tekst

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

Większe prawdopodobieństwo wyboru losowej atrakcji przez mrówkę

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 531,45 | 546,39 | 467,35 | 518,60 | 512,66 | 515,66 | 467,35 | 546,39 |
| 80 | 1042,73 | 1078,45 | 1031,02 | 1014,75 | 1042,13 | 1042,10 | 1014,75 | 1078,45 |

Obraz zawierający linia, diagram

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres

Opis wygenerowany automatycznie

Większy współczynnik alfa

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 3
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 472,36 | 496,06 | 475,78 | 468,16 | 502,08 | 482,08 | 468,16 | 502,08 |
| 80 | 799,12 | 832,68 | 826,62 | 864,11 | 788,30 | 822,16 | 788,30 | 864,11 |

Obraz zawierający diagram, linia, Wykres

Opis wygenerowany automatycznieObraz zawierający diagram, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

Większy współczynnik beta

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 4
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 490,45 | 474,54 | 485,13 | 491,03 | 482,88 | 484,80 | 474,54 | 491,03 |
| 80 | 819,71 | 819,42 | 799,67 | 821,34 | 822,44 | 816,51 | 799,67 | 822,44 |

Obraz zawierający diagram, linia, mapa, tekst

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres

Opis wygenerowany automatycznie

Większa liczba iteracji

* Ilość iteracji = 100
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 489,64 | 483,84 | 470,30 | 454,82 | 483,59 | 476,43 | 454,82 | 489,64 |
| 80 | 822,62 | 880,75 | 810,93 | 841,00 | 790,46 | 841,31 | 810,93 | 880,75 |

Obraz zawierający diagram, linia, zrzut ekranu, mapa

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

Mniejsza liczba iteracji

* Ilość iteracji = 20
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.7
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 475,68 | 493,70 | 466,13 | 477,80 | 481,14 | 478,89 | 466,13 | 493,70 |
| 80 | 827,61 | 844,97 | 865,99 | 860,62 | 818,95 | 843,62 | 818,95 | 865,99 |

Obraz zawierający diagram, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

Mniejszy współczynnik wyparowania

* Ilość iteracji = 50
* Ilość mrówek w kolonii = 50
* Współczynnik wyparowania feromonu = 0.3
* Współczynnik alfa = 1
* Współczynnik beta = 2
* Współczynnik wyboru losowej atrakcji = 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba atrakcji | Otrzymane wyniki | | | | | | | |
|  | 1. | 2. | 3. | 4. | 5. | średnia | Min | Max |
| 32 | 489,71 | 479,82 | 460,71 | 484,72 | 484,37 | 479,86 | 460,71 | 489,71 |
| 80 | 841,82 | 867,59 | 795,67 | 830,35 | 841,55 | 835,39 | 795,67 | 867,69 |

Obraz zawierający diagram, linia, mapa, Wykres

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

1. Wpływ zmiany ilości mrówek

Zwiększenie ilości mrówek skutkuje poprawą stabilności wyników, redukuje zmienność i prowadzi do bardziej spójnych wyników. Większa kolonia mrówek wpływa na poprawę otrzymanych wyników. Kolonia składająca się ze 100 mrówek pokonuje najmniejszy średni dystans, aby odwiedzić każdą atrakcję. Z kolei kolonia składająca się z 25 mrówek charakteryzuje się większym rozrzutem wyników.

1. Większe prawdopodobieństwo wyboru losowej atrakcji

Zwiększenie prawdopodobieństwa wyboru losowej atrakcji ma znaczący wpływ na otrzymane wyniki. Po pierwsze jakość otrzymanych wyników jest dużo gorsza w porównaniu do pierwszego przeprowadzonego eksperymentu, mrówki pokonują dużą większą odległość zanim odwiedzą wszystkie atrakcje. Po drugie zwiększenie prawdopodobieństwa wyboru losowej atrakcji ma także wpływ na zróżnicowanie otrzymanych wyników. Różnice w minimalnych i maksymalnych wynikach są większe w porównaniu do pierwszego eksperymentu.

1. Zwiększenie współczynnika alfa

Zwiększenie współczynnika alfa pozwoliło na uzyskanie nieznacznie lepszych średnich wyników dla obu ilości atrakcji. Natomiast równocześnie przyczyniło się do zwiększenia zakresu otrzymanych wyników.

1. Zwiększenie współczynnika beta

Większa wartość współczynnika beta pozwoliła na uzyskanie lepszych wyników tylko w przypadku 80 atrakcji, natomiast większy współczynnik beta w przeciwieństwie do przypadku z większym współczynnikiem alfa nie charakteryzuje się brakiem stabilności otrzymanych wyników. Na podstawie tego można wywnioskować, że mrówki dużo chętniej wybierają te atrakcje, między którymi jest najmniejsza odległość, a nie wybierają tych tras na których znajduje się więcej feromonów, a co nie zawsze jest gwarancją najlepszego wyboru.

1. Wpływ zmiany ilość iteracji

Zwiększenie liczby iteracji skutkuje poprawą stabilności wyników, redukuje zmienność i prowadzi do bardziej spójnych wyników. Mniejsza liczba iteracji mrówek charakteryzuje się większym rozrzutem wyników. Z kolei większa liczba iteracji wpływa na poprawę otrzymanych wyników. Algorytm wykonujący się 100 razy zwraca najmniejszy średni dystans potrzebny, aby odwiedzić każdą atrakcję.

1. Zmniejszenie współczynnika wyparowania

Zmniejszenie współczynnika wyparowania pozwoliło na uzyskanie nieznacznie lepszych średnich wyników dla obu ilości atrakcji oraz jednocześnie przyczyniło się do zwiększenia zakresu otrzymanych wyników.