Rozwiązanie problemu Capacitated Vehicle Routing Problem przy pomocy algorytmu mrówkowego

Piotr Bródka

Kwiecień 2019

1 Wstęp

Istnieją w informatyce problemy trudno obliczalne (za takie można uznać problemy NP-trudne). Takim problemem jest na przykład problem komiwojażera i jego modyfikacje. Wyznaczenie wyniku ma zazwyczaj w takich problemach złożoność wykładniczą, co oznacza, że w praktyce obliczenie dokładnego rozwiązania nie jest możliwe, gdy wielkość danych wejściowych nie jest bardzo mała. W tej sytuacji z pomocą przychodzą algorytmy aproksymacyjne, w szczególności algorytmy inspirowane biologią. W niniejszej pracy pokażę, jak wykorzystać inteligencję rojową (konkretnie - algorytm mrówkowy).

2 Opis problemu - CVRP

Opis tego algorytmu należy zacząć od przypomnienia problemu komiwojażera. Chodzi w nim o znalezienie cyklu Hamiltona o najmniejszej sumie wag. Tłumacząc na przykładzie, wyobraźmy sobie, że mamy n miast. Każde połączone z każdym, wiemy jakie są odległości między nimi. Musimy odwiedzić wszystkie miasta w taki sposób, by sumaryczna przejechana odległość była jak najmniejsza.

Problem CVRP jest rozszerzeniem problemu komiwojażera. W każdym z miast mamy zapotrzebowanie na jakąś ilość produktu. Komiwojażer rusza z punktu startowego (np. fabryki) i musi dostarczyć do miast produkty w wymaganej ilości. Mamy wielu komiwojażerów (dostawców). Problemem do rozwiązania jest minimalizacja drogi przejechanej przez dostawców.

3 Opis algorytmu - Algorytm mrówkowy

Na początek opis algorytmu będzie odnosił się do podstawowej wersji problemu komiwojażera. Wyobraźmy sobie, że w miastach rozkładane jest jedzenie i mamy stado mrówek. Mrówki chcą odwiedzić wszystkie miasta, bo jest tam jedzenie (dla każdej mrówki jedzenie z danego miasta znika po odwiedzeniu tego miasta). Mrówki rozpylają na swojej trasie feromony, jest to sposób kontaktu wewnątrz stada. Na trasie najkrótszej koncentracja feromonów będzie największa i będzie zachęcała mrówki do poruszania się tą trasą (inne mrówki "zachęcają" do wyboru tej trasy). Oczywiście, należy pamiętać o tym, że feromony zanikają (parują). W związku z tym, ostatecznie na trasach nieuczęszczanych nie będzie feromonów w ogóle.

3.1 Opis matematyczny

To, w jaki sposób mrówka k będzie poruszała się od miasta i do j (przy założeniu, że jest aktualnie w mieście i) będzie zgodne z następującym rozkładem prawdopodobieństwa:

$$p_{ij}^k(t) = \frac{(r_{ij}(t))^\alpha * (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (r_{il}(t))^\alpha * (\eta_{il})^\beta}$$

To znaczy, że miasto z największą wartością nie będzie na pewno wybrane, ale będzie miało największe prawdopodobieństwo wyboru.

η_{ij} to odwrotność odległości między miastami i oraz j. r_{ij} to intensywność feromonów na odcinku między miastami

Wydaje się, że ważnym aspektem będzie dobre wyważenie między współczynnikami α i β. Zauważmy, że:

1. Jeśli $\alpha = 0$, algorytm zdegeneruje się do algorytmu zachłannego, będzie szedł do tego miasta, które jest w danej chwili najbliżej.
2. Jeśli $\beta = 0$, algorytm będzie działal losowo, utknie w wyborze drogi, która może nie być optymalna.

Mrówka zostawia na wszystkich odcinkach (i,j) przebytej przez siebie trasy porcję feromonu:

$$\Delta r_{ij}^k(t) = \frac{Q}{L^k(t)}$$

Gdzie Q to stały parametr, a $L^k(t)$ to długość drogi przebytej przez mrówkę k w iteracji t. Zakładamy parowanie (zanikanie) feromonu. Zmianę ilości feromonu w czasie opisuje wzór:

$$r_{ij}(t+1) = (1-\rho) * r_{ij}(t) + \Delta r_{ij}(t)$$

Gdzie $\Delta r_{ij}(t) = \sum_{k=1}^{m} \Delta r_{ij}^k(t)$. m to liczba mrówek.

3.2 Modyfikacja dla problemu CVRP

Dla problemu CVRP, algorytm trzeba zmodyfikować, by był zgodny z założeniami problemu. W każdej iteracji puszczany wiele mrówek. Niezależnie szukają trasy. Wówczas mrówka kontynuuje trasę, bazując na koncentracji feromonów. W przypadku, gdy zostawimy cały ładunek, musimy zakończyć rozwożenie i wrócić do fabryki (punktu początkowego).

(Jak zostało wskazane w poprzedniej sekcji) Aktualizacja feromonów w danej iteracji następuje po przebyciu trasy przez wszystkie mrówki (tzn. nie robimy tak: mrówka pierwsza idzie, zostawia feromon, mrówka druga idzie po grafie z zauktualizowanym przez mrówkę pierwszą feromonem. . .). Jak sugeruje praca [3], najlepiej jest, gdy licza mrówek jest równa (lub zbliżona do) ilości miast. U mnie w programie te liczby są równe.

Uwaga implementacyjna: Na początku, wartości feromonów powinny być niezerowymi małymi wartościami z rozkładu jednostajnego.

4 Opis danych wejściowych, weryfikacja rezultatów

Początkowym pomysłem było generowanie własnych danych i "zbenchmarkowanie" swojego rozwiązania za pomocą biblioteki Google Or-Tools. Okazuje się, że w Internecie można znaleźć gotowe dane i benchmarki do sprawdzenia działania algorytmu. Skorzystam z drugiego rozwiązania. Odpowiednie dane można znaleźć na przykład tutaj: https://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm.old

5 Technologie

Do napisania programu został użyty język Python3. Z pomocniczymi bibliotekami: numpy oraz networkx (do wizualizacji rozwiązań)

6 Wyniki

Algorytm znajduje rozwiązania niewiele gorsze od optymalnych przy podstawowych ustawieniach: $\alpha = 1$, $\beta = 1$, $\rho = 0.8$, $Q = 1$, 500 iteracji. Oto wyniki dla poszczególnych przykładów:

	path	ratio
0	A-n32-k5	1.0618
1	A-n33-k5	1.0581
2	A-n33-k6	1.0570
3	A-n34-k5	1.0807
4	A-n36-k5	1.1106
5	A-n37-k5	1.2146
6	A-n37-k6	1.0694
7	A-n38-k5	1.1000
8	A-n39-k5	1.1273
9	A-n39-k6	1.1512
10	A-n44-k7	1.1037

Table 1: Porównanie wyników. Ratio to stosunek długości, zwróconej przez mój algorytm do długości z benchmarku.

6.1 Przykładowa wizualizacja rozwiązań

Poniżej przedstawione jest rozwiązanie optymalne dla przykładu **A-n33-k5** oraz poniżej - rozwiązanie znalezione przez mój algorytm.

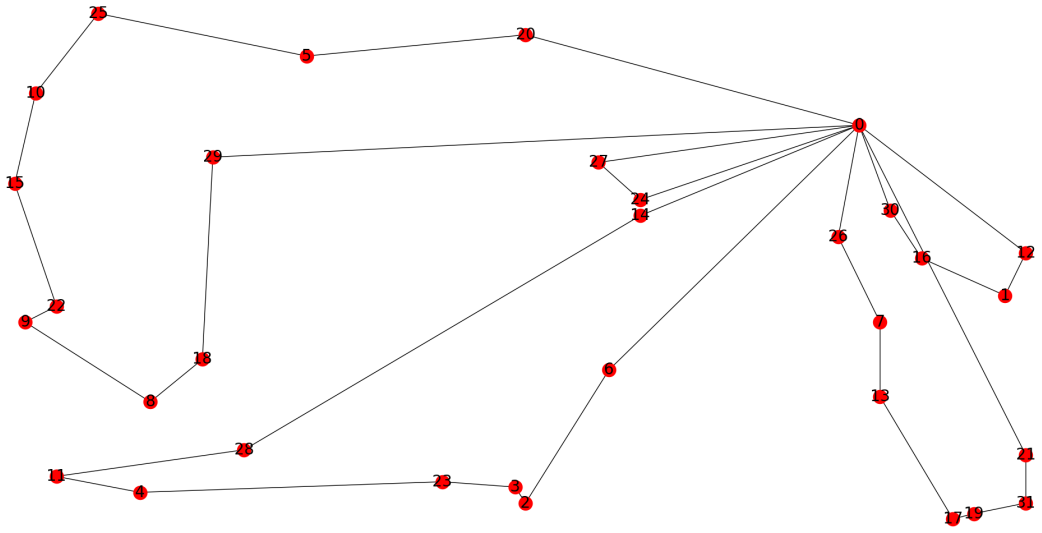

Figure 1: Rozwiązanie optymalne

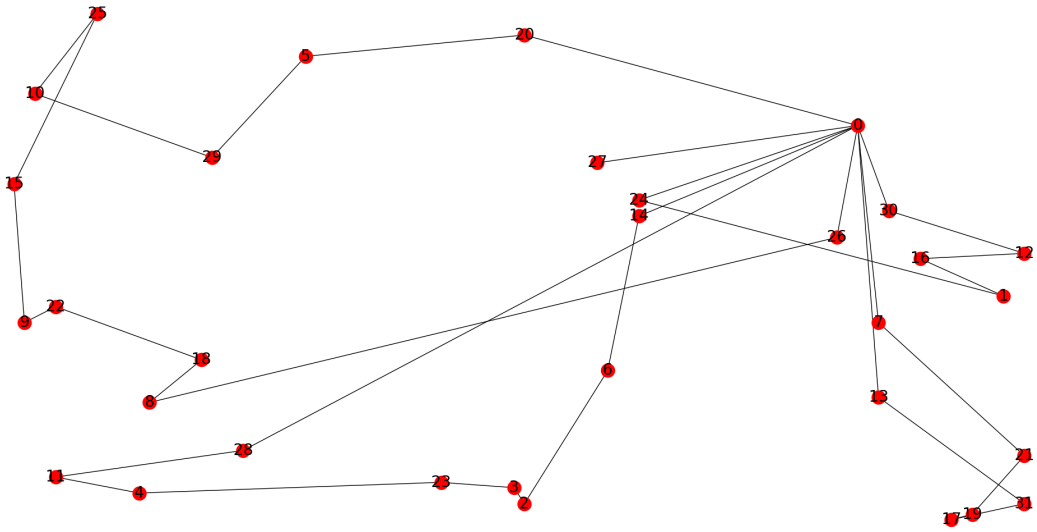

Figure 2: Rozwiązanie, znalezione przez mój algorytm

6.2 Dostrajanie zmiennych

Działanie algorytmu zależy od czterech zmiennych.

1. Liczba iteracji
2. α - wykładnik przy potędze ilości feromonu (do obliczania rozkładu prawdopodobieństwa)
3. β - wykładnik przy potędze odwrotności odległości (do obliczania rozkładu prawdopodobieństwa)
4. $(1 - \rho)$ - tyle feromonu zostaje po każdej iteracji
5. Q - taka wartość feromonu, dzielona przez długość trasy jest wydzielana na trasie przez mrówkę.

Badanie wszystkich ich kombinacji (nawet w danych zakresie) byłoby karkołomnym zadaniem. Praca [3] sugeruje użycie następującego zestawu zmiennych:

$$\alpha = 2, \beta = 5, 1 - \rho = 0.8, Q = 80$$

Oto wyniki:

	path	ratio
0	A-n32-k5	1.260574
1	A-n33-k5	1.090007
2	A-n33-k6	1.118934
3	A-n34-k5	1.081439
4	A-n36-k5	1.136319
5	A-n37-k5	1.243153
6	A-n37-k6	1.153935
7	A-n38-k5	1.185502
8	A-n39-k5	1.077709
9	A-n39-k6	1.176752
10	A-n44-k7	1.103405

Table 2: Porównanie wyników dla zestawu parametrów, zaproponowanych w [3]

Okazuje się, że wyniki wcale nie są bardzo dobre, jak moglibyśmy się spodziewać. Są gorsze od uprzednio osiągniętych Zauważmy, że przy tych ustawieniach maksymalny błąd osiągnął 26%. Potwierdza to fakt, że dobór parametrów nie jest łatwym zadaniem.

6.3 Jak ilość iteracji wpływa na jakość rozwiązania?

Sprawdźmy, jak dla początkowo dobranych danych wygląda zależność optymalnego rozwiązania od ilości iteracji (dla pierwszego ustawienia parametrów i pierwszego zbioru testowego):

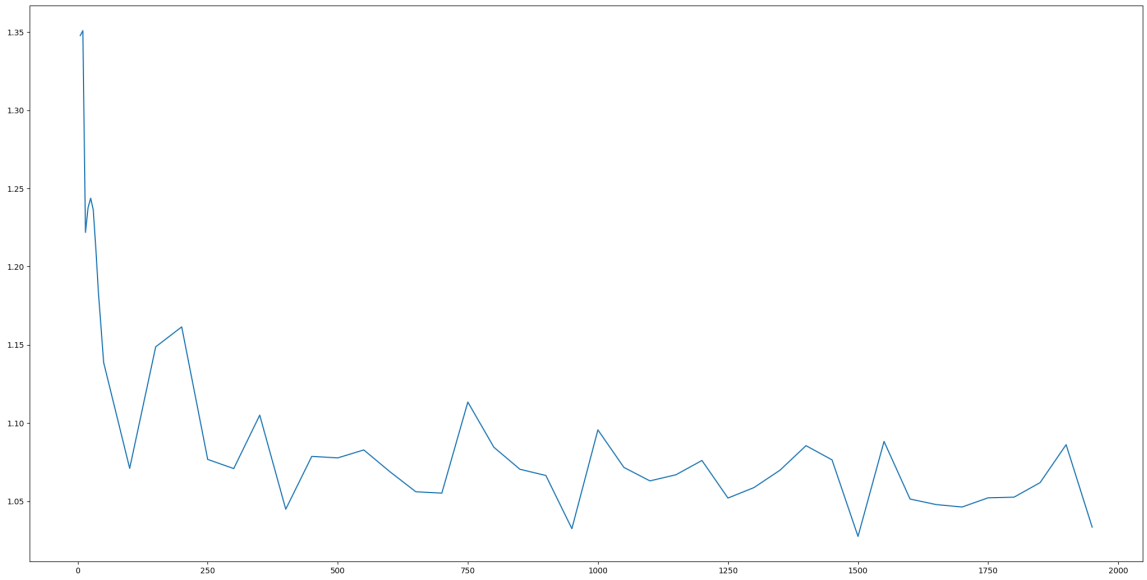

Figure 3: Spadek stosunku długości mojego rozwiązania do rozwiązania optymalnego w zależności od ilości iteracji

Widać, że stosunek długości osiągniętego rozwiązania do długości rozwiązania optymalnego maleje z czasem, jednak nie jest to spadek jednostajny. Do uzyskania dobrego wyniku wystarcza około 200 iteracji. Potem wraz ze wzrostem ilości iteracji, zyski w jakości wyniku są nieznaczne.

7 Wnioski

Użycie algorytmu mrówkowego jest dobrym pomysłem na optymalizację problemów trudno obliczalnych. Sukcees tej metody zależy w dużym stopniu od doboru parametrów algorytmu.

References

[1] Capacitated Vehicle Routing Problem
`https://developers.google.com/optimization/routing/cvrp`

[2] Dreo, Petrowski, Siarry, Taillard, Springer 2006, *Metaheuristics for Hard Optimization: Methods and Case Studies*

[3] Tan, W.F., L.S. Lee, Z.A. Majid H.V. Seow *Ant Colony Optimization for Capacitated Vehicle Routing Problem*
`https://thescipub.com/pdf/10.3844/jcssp.2012.846.852`