10 grudnia 2019

Politechnika Wrocławska

Ewolucyjny algorytm dla nieliniowego zadania transportowego

Wydział Podstawowych Problemów Techniki

Piotr Berezowski

Opiekun pracy: dr hab. Paweł Zieliński

Plan prezentacji

- Cel pracy
- Wprowadzenie
  - Zadanie transportowe
  - Algorytmy ewolucyjne
- Implementacja
- Wersja równoległa
- Wyniki
- Podsumowanie

Wprowadzenie

Cel pracy

1. Implementacja algorytmu ewolucyjnego dla zadania transportowego.
2. Analiza eksperymentalna.

Zadanie transportowe

- Mamy zdefiniowane n punktów nadania i m punktów odbioru.
- Każdy z punktów nadania ma określoną podaż.
- Każdy punkt odbioru ma określony popyt.
- Znaleźć plan optymalnego transportu.

Zadanie transportowe

Funkcja celu:

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(x_{ij})$$

Ograniczenia:

$$\sum_{j=1}^{m} x_{ij} = supply(i), \text{ dla } i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = demand(j), \text{ dla } j = 1, 2, \ldots, m$$

$$x_{ij} \quad 0, \text{ dla } i = 1, 2, \ldots, n \text{ i } j = 1, 2, \ldots, m$$

Algorytmy metaheurystyczne

Algorytmy ewolucyjne

Implementacja

Implementacja

- Reprezentacja rozwiązania w postaci macierzy.
- Najbardziej intuicyjny sposób reprezentacji dla omawianego problemu.

	s_1	s_2	s_3	s_4	s_5	*demand*
d_1	0.0	7.0	5.0	0.0	0.0	12.0
d_2	5.0	0.0	0.0	0.0	5.0	10.0
d_3	3.0	0.0	0.0	0.0	0.0	3.0
d_4	0.0	0.0	0.0	3.0	7.0	10.0
d_5	2.0	0.0	0.0	10.0	0.0	12.0
supply	10.0	7.0	5.0	13.0	12.0	

Przykładowe rozwiązanie.

Implementacja

Inicjalizacja:

- Generujemy losową permutację wszystkich indeksów macierzy rozwiązania.
- Wstawiamy po kolei dla komórki o indeksie (i, j) $val = \min(demand[i], supply[j])$.
- Zmniejszamy odpowiednio popyt i podaż o wartość val.

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	0.0	0.0	8.0
d_2	0.0	0.0	0.0	12.0
supply	5.0	8.0	7.0	

Niezainicjalizowana macierz

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	0.0	0.0	8.0
d_2	**5.0**	0.0	0.0	**7.0**
supply	**0.0**	8.0	7.0	

Inicjalizujemy indeks $(\mathbf{2},\mathbf{1})$

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	0.0	**7.0**	**1.0**
d_2	5.0	0.0	0.0	7.0
supply	0.0	8.0	**0.0**	

Inicjalizujemy indeks $(\mathbf{1},\mathbf{3})$

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	0.0	7.0	1.0
d_2	5.0	0.0	**0.0**	**7.0**
supply	0.0	8.0	**0.0**	

Inicjalizujemy indeks $(\mathbf{2},\mathbf{3})$

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	0.0	7.0	1.0
d_2	5.0	**7.0**	0.0	**0.0**
supply	0.0	**1.0**	0.0	

Inicjalizujemy indeks $(2,2)$

Implementacja

Przykład:

- Macierz 2×3
- Permutacja indeksów:
  $[(2,1),(1,3),(2,3),(2,2),(1,2),(1,1)]$

	s_1	s_2	s_3	*demand*
d_1	0.0	**1.0**	7.0	**0.0**
d_2	5.0	7.0	0.0	0.0
supply	0.0	**0.0**	0.0	

Inicjalizujemy indeks $(\mathbf{1},\mathbf{2})$

Implementacja

- Operator krzyżowania
  - Selekcja metodą ruletki.
  - Kombinacja wypukła dwóch rodziców.
  - Ograniczenia zadania są spełnione przez dzieci, jeśli są spełnione przez rodziców.
- Operator mutacji
  - Wybieramy losową podmacierz z macierzy rozwiązania.
  - Inicjalizujemy na nowo podmacierz.

Wersja równoległa

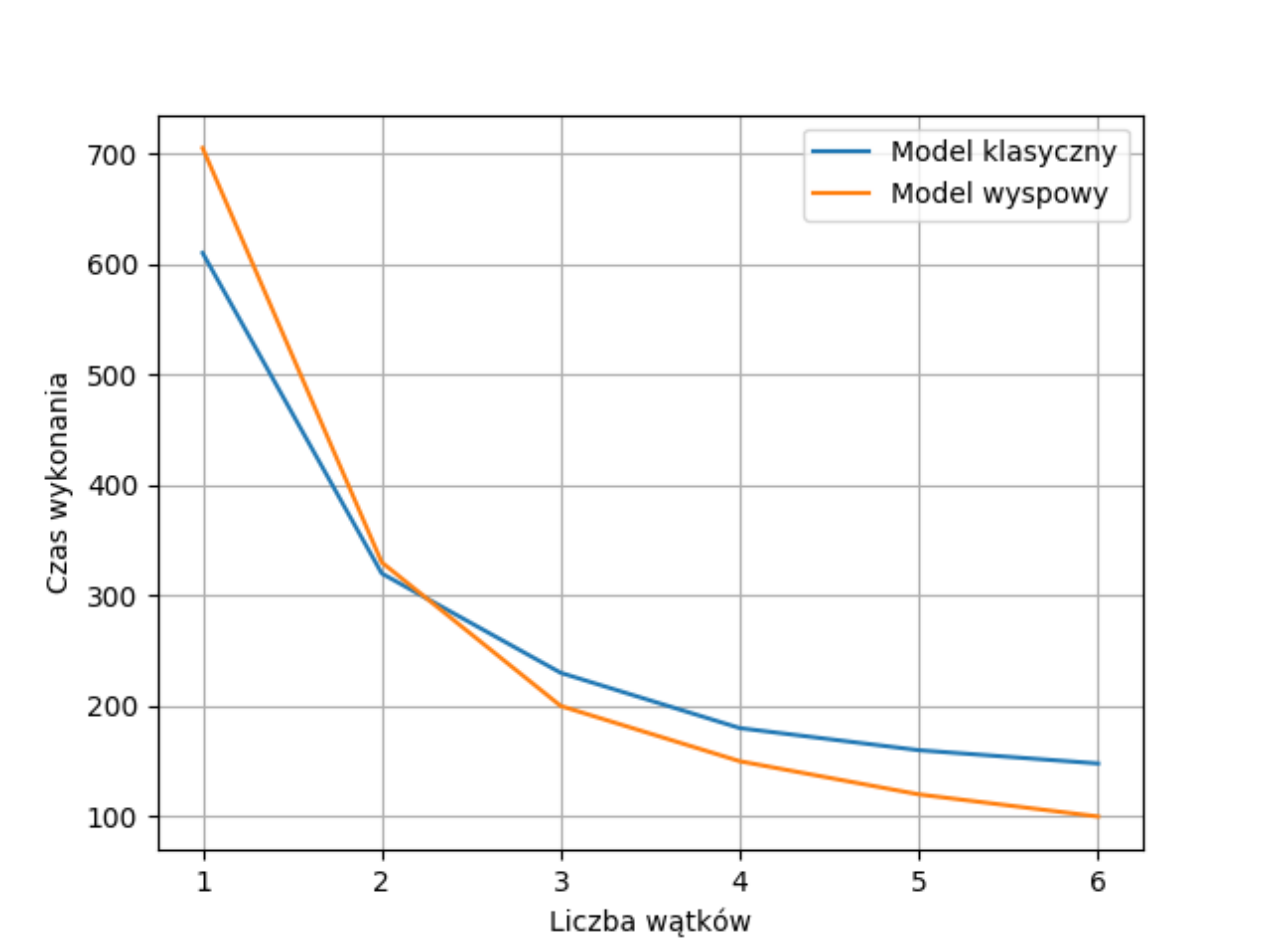
Modele ewolucji

Dodano dwa modele ewolucji populacji:

- Klasyczny
  - Zrównoleglenie na poziomie operatorów i funkcji przystosowania.
- Wyspowy
  - Podział na populacje częściowe.
  - Każda populacja ewoluuje niezależnie od innych przez określoną liczbę pokoleń.

Wyniki

Czas znalezienia rozwiązania



Zależność czasu znalezienia rozwiązania od liczby wątków

Podsumowanie