Lab7c

Wojciech Klusek 305943 grupa C

21 stycznia 2023

Spis treści

1 Zadanie

Zadaniem było minimalnej odległości między dwoma wypukłymi rozłącznymi wielokątami. Rozważmy dwa zbiory punktów $P = \{p_1, ..., p_r\}$ oraz $Q = \{q_1, ..., q_s\}$ definiujące wierzchołki dwóch wypukłych rozłącznych wielokątów (w R^2, r + s = m). Zadaniem było znaleźć minimalną odległość wielokątów, tzn. $min(||p - q||)$, gdzie $p = \sum_{i=1}^{r} \lambda_i p_i$ oraz $q = \sum_{i=1}^{s} \lambda_i q_i$. Powyższy problem można sprowadzić do następującego zadania programowania kwadratowego:

$$\min_{x \in \Omega}(x^T D x)$$
$$\text{gdzie } \Omega = \begin{cases} \sum_{i=1}^{r} x_i = 1, \\ \sum_{i=r+1}^{m} x_i = 1, \\ x \geqslant 0 \end{cases}$$

Macierz C jest konstruowana w taki sposób, aby zawierała różnice współrzędnych między punktami P i Q. Macierz ta ma wymiar $2 \times (r+s)$, gdzie pierwsze r kolumn odpowiada współrzędnym punktów z P, a kolejne s kolumn - współrzędnym punktów z Q. Każdy wiersz macierzy C odpowiada jednemu wymiarowi: pierwszy wiersz dla współrzędnej x, a drugi wiersz dla współrzędnej y. Tak więc, macierz C jest zdefiniowana jako:

$$C = \begin{bmatrix} P_{x_1} & P_{x_2} & \cdots & P_{x_r} & -Q_{x_1} & -Q_{x_2} & \cdots & -Q_{x_s} \\ P_{y_1} & P_{y_2} & \cdots & P_{y_r} & -Q_{y_1} & -Q_{y_2} & \cdots & -Q_{y_s} \end{bmatrix}$$

gdzie P_{x_i} i P_{y_i} to odpowiednio współrzędne x i y punktów z P, a Q_{x_j} i Q_{y_j} to współrzędne x i y punktów z Q.

Macierz D, będąca iloczynem $C^T C$, jest macierzą kwadratową o wymiarach $(r+s) \times (r+s)$. Elementy macierzy D są sumami kwadratów i iloczynów współrzędnych punktów z P i Q:

$$D = \begin{bmatrix} P_{x_1}^2 + P_{y_1}^2 & \cdots & -P_{x_1}Q_{x_1} - P_{y_1}Q_{y_1} & \cdots \\ \vdots & \ddots & \vdots & \\ -P_{x_1}Q_{x_1} - P_{y_1}Q_{y_1} & \cdots & Q_{x_1}^2 + Q_{y_1}^2 & \cdots \\ \vdots & & \vdots & \ddots \end{bmatrix}$$

Gdzie sumy są brane przez wszystkie punkty P i Q. Elementy na głównej przekątnej macierzy D odpowiadają sumom kwadratów współrzędnych tych samych punktów, a elementy poza główną przekątną reprezentują iloczyny współrzędnych różnych punktów.

2 Wyprowadzenie wzorów

Powyższe zadanie możemy przekształcić do barierowej postaci zadania:

$$\min_{x\in\Omega}\left(x^T Dx - \mu\sum_{i=1}^{m}\ln x_i\right)$$

Funkcja Lagrange'a dla powyższego zadania wygląda następująco:

$$L(x,y) = x^T Dx - \mu\sum_{i=1}^{m}\ln x_i + y^T(b - Ax)$$

Warunki Kuhna-Tuckera wyglądają następująco:

$$\begin{cases}\nabla_x L(x,y) = Dx + r_k\frac{1}{x_j} - (A^T y)_j = 0, j = 1,2,...,n\\ \nabla_y L(x,y) = Ax - b = 0\end{cases}$$

Z warunków Kuhna-Tuckera otrzymujemy:

$$\begin{cases}Dx - r_k X^{-1}e - A^T y = 0\\ Ax - b = 0\end{cases}$$

$$\begin{cases}Dx - z - A^T y = 0\\ XZe = r_k e\\ Ax - b = 0\\ x,y,z > 0\end{cases}$$

3 Opis methody punktu wewnętrznego

Dla każdej iteracji:

- **Warunek Stopu**: Sprawdzamy, czy rozwiązanie spełnia powyższe warunki KKT. Jeśli tak, ustawiamy `exitflag` na 1 i przerywamy pętlę.
- **Obliczenie kierunków**:
  - Ustalamy wartość r jako $\sigma\frac{z^T x}{n+m}$.
  - Tworzymy diagonalne macierze X i Z z wektorów x i z.
  - Rozwiązujemy poniższy układ równań aby znaleźć kierunki dla x, y, i z:

$$\begin{bmatrix}-(X^{-1}Z + D) & A_{eq}^T\\ A_{eq} & 0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \begin{bmatrix}-A_{eq}^T y - r(X^{-1}e) + Dx\\ b_{eq} - A_{eq}x\end{bmatrix}$$

- **Obliczenie długości kroku** (α):

- Obliczamy wartości β_x i β_z:

$$\beta_x = -\beta \left(\frac{x}{\Delta x} \right) \quad , \quad \Delta x < 0$$

$$\beta_z = -\beta \left(\frac{z}{\Delta z} \right) \quad , \quad \Delta z < 0$$

- Długość kroku α jest minimum z 1 i najmniejszych wartości β_x i β_z.

- **Aktualizacja rozwiązania**: Uaktualniamy x, y, i z stosując obliczone kierunki i długość kroku.

4 Rozwiązanie

Rozwiązanie składa się z 7 plików:

1. `quadprog_solution.m` - Rozwiązanie z wykorzystaniem funkcji `quadprog`. Przyjmuje jako wejście:

   - `P` - macierz z punktami wielokąta `P`,
   - `Q` - macierz z punktami wielokąta `Q`,

   Funkcja zwraca wektor `x` będący rozwiązaniem zadania, `dist` odległość między znalezioną parą punktów, `p` będący punktem na wielokącie `P`, `q` będący punktem na wielokącie `Q` oraz `exitflag` określającą czy rozwiązanie zostało znalezione.

2. `IPM.m` - Rozwiązanie z wykorzystaniem metody punktu wewnętrznego. Przyjmuje jako wejście:

   - `P` - macierz z punktami wielokąta `P`,
   - `Q` - macierz z punktami wielokąta `Q`,

   Funkcja zwraca wektor `x` będący rozwiązaniem zadania, `dist` odległość między znalezioną parą punktów, `p` będący punktem na wielokącie `P`, `q` będący punktem na wielokącie `Q`, `exitflag` określającą czy rozwiązanie zostało znalezione oraz `it` określającą liczbę iteracji.

3. `dane.m` - Losowo generuje punkty będące wierzchołkami wielokątów. Przyjmuje jako wejście:

   - `r` - liczba wierzchołków wielokąta P,
   - `s` - liczba wierzchołków wielokąta Q,

   Funkcja zwraca macierz `P` i `Q` zawierające punkty rozłącznych wielokątów.

4. `draw.m` - Rysuje wielokąty oraz linię między nimi. Przyjmuje jako wejście:

   - `P` - macierz punktów z wielokąta `P`,

- `Q` - macierz punktów z wielokąta `Q`,
- `p` - punkt z wielokąta `P`,
- `q` - punkt z wielokąta `Q`,

5. `get_initial_form.m` - Zwraca podstawową formę zadania. Przyjmuje jako wejście:

   - `P` - macierz punktów z wielokąta `P`,
   - `Q` - macierz punktów z wielokąta `Q`,

   Zwraca macierz `D`, lewą stronę ograniczeń równościowych `Aeq`, prawą stronę ograniczeń równościowych `beq` oraz dolne ograniczenie `lb`.

6. `get_points.m` - Zwraca punkty na podstawie rozwiązania zadania kwadratowego. Przyjmuje jako wejście:

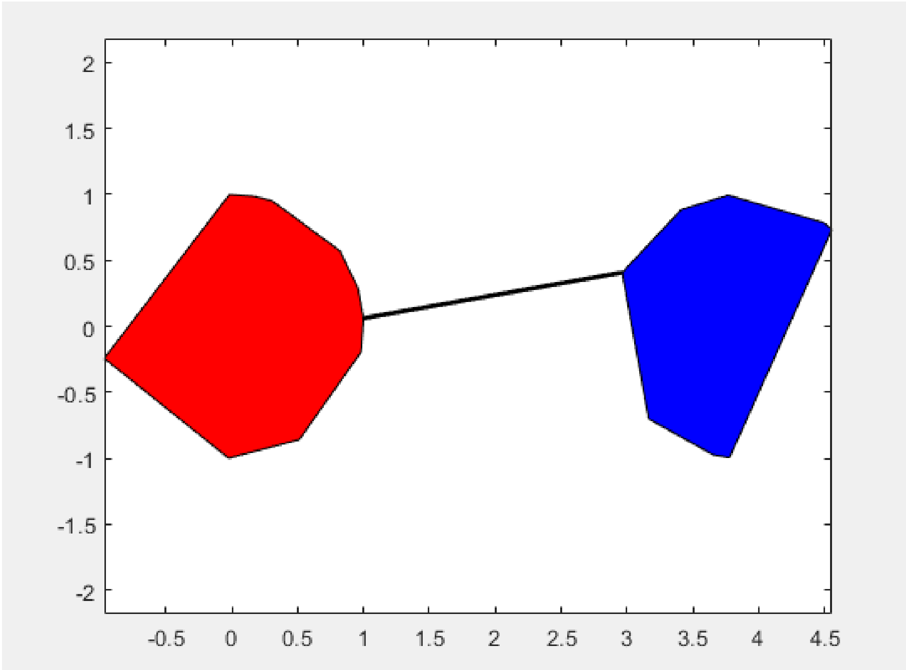
   - `x` - rozwiązanie zadania,
   - `P` - macierz punktów z wielokąta `P`,
   - `Q` - macierz punktów z wielokąta `Q`,

   Funkcja zwraca punkt `p` z wielokąta `P`, punkt `q` z wielokąta `Q` oraz odległość między nimi - `dist`.

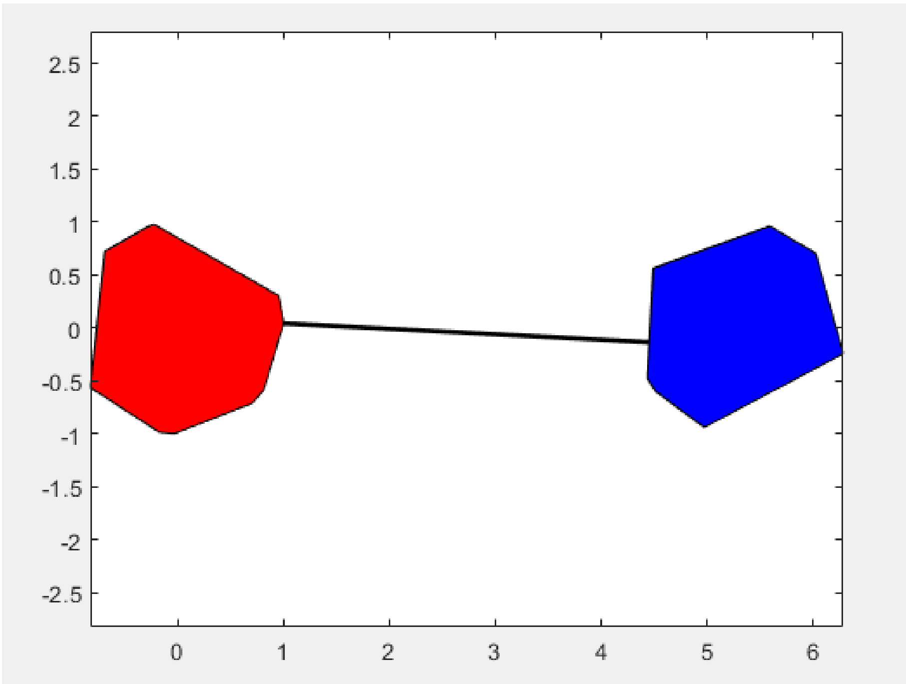
7. `test.m` - Porównuje rozwiązanie wykorzystujące `quadprog` z rozwiązaniem wykorzystującym metodę punktu wewnętrznego.

5 Przykłady

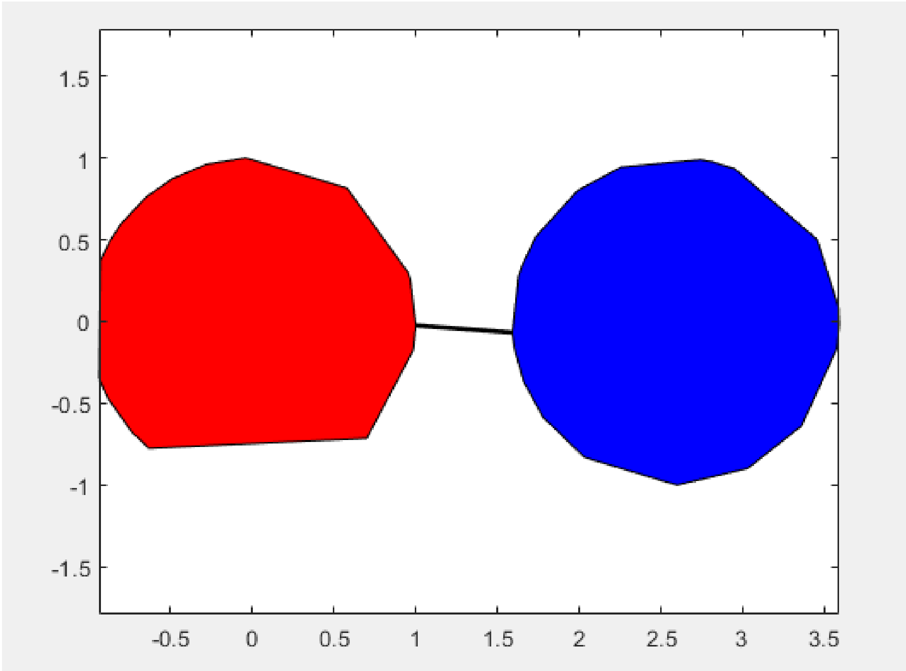
Poniżej znajdują się wizualizacje najmniejszych odległości między wielokątami z wykorzystaniem metody punktu wewnętrznego.



2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-0.5 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5



2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0 1 2 3 4 5 6



6 Test dokładności rozwiązania

Do testu dokładności wykorzystane zostały następujące zmienne:

- Liczba punktów w wielokątach - 20,
- Liczba różnych wejść - 100,
- Ziarno generatora liczb losowych - 1,
- Dokładność obliczeń - $1e - 6$,

Poniżej przedstawione zostały wyniki dla powyższych parametrów:

- Dokładność: 100% (określa jak często algorytm zwrócił tą samą wartość co `quadprog`).
- Średnia liczba iteracji IPM: 13.33.
- Średnia liczba iteracji quadprog: 13.33.

7 Wnioski

Optymalizacja problemu polegającego na znalezieniu pary punktów, które minimalizują odległość między wielokątami, może być skutecznie przekształcona w zadanie optymalizacji funkcji kwadratowej. Algorytm, wykorzystujący metodę punktu wewnętrznego do rozwiązywania problemów optymalizacji funkcji kwadratowej z ograniczeniami, osiągnął takie same wyniki co wbudowana funkcja `quadprog` przy średniej liczbie iteracji wynoszącej 13.33.

8 Oświadczenie o samodzielności

Oświadczam, że niniejsza praca stanowiąca podstawę do uznani osiągnięcia efektów uczenia się z przedmiotu Programowanie Matematyczne została wykonana przeze mnie samodzielnie.

Wojciech Klusek 305934