

Report of Programming Task 3 of the course "Introduction to Optimization" - Fall 2024

Nikita Zagainov, Ilyas Galiev, Arthur Babkin, Nikita Menshikov,
Sergey Aitov

September 2024

1 Team Leader Report

- Team leader: Nikita Zagainov — 5
Managed team work, Implemented North-West corner method, Wrote report
- Team member 1: Ilyas Galiev — 5
Wrote QA tests for all algorithms, Contributed to the implementation of the Russel's approximation algorithm
- Team member 2: Arthur Babkin — 5
Implemented Russel's approximation algorithm
- Team member 3: Nikita Menshikov — 5
Implemented Vogel's approximation algorithm
- Team member 4: Sergey Aitov — 5
Contributed to the implementation of the Russel's approximation algorithm, Developed test cases for all algorithms

2 Link to the product

[Project source code](#)

3 Programming language

Python

4 Testing Results

All the algorithm we implemented were tested on 4 inputs. The acceptance criteria for solvable problems was the following:

$$\sum_{j=1}^n X_{ij} = S_i, \quad \forall i$$

$$\sum_{i=1}^m X_{ij} = D_j, \quad \forall j$$

where X_{ij} is the amount of goods transported from i -th source to j -th destination, S_i is the supply of i -th source, and D_j is the demand of j -th destination.

The following test cases were used:

- Test Case 1 (valid)

$$S = [20 \quad 30 \quad 50]$$

$$D = [30 \quad 10 \quad 30 \quad 30]$$

$$C = \begin{bmatrix} 5 & 4 & 3 & 2 \\ 7 & 6 & 5 & 3 \\ 6 & 5 & 4 & 3 \end{bmatrix}$$

- Test Case 2 (valid)

$$S = [100 \quad 200 \quad 300]$$

$$D = [50 \quad 500 \quad 50]$$

$$C = \begin{bmatrix} 10 & 7 & 8 \\ 3 & 2 & 4 \\ 2 & 4 & 6 \end{bmatrix}$$

- Test Case 3 (valid)

$$S = [10 \quad 10 \quad 10 \quad 10 \quad 10]$$

$$D = [10 \quad 10 \quad 10 \quad 10 \quad 10]$$

$$C = \begin{bmatrix} 2 & 3 & 1 & 4 & 5 \\ 4 & 1 & 2 & 3 & 2 \\ 3 & 2 & 5 & 1 & 2 \\ 1 & 4 & 3 & 2 & 1 \\ 2 & 3 & 2 & 1 & 4 \end{bmatrix}$$

- Test Case 4 (valid)

$$S = [50 \quad 70 \quad 120 \quad 80 \quad 60]$$

$$D = [100 \quad 120 \quad 70 \quad 90]$$

$$C = \begin{bmatrix} 2 & 3 & 1 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 2 & 5 & 1 \\ 1 & 4 & 3 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix}$$

- Test Case 5 (invalid)

$$S = [20 \quad 40 \quad 10]$$

$$D = [30 \quad 10 \quad 10]$$

$$C = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 1 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

5 Code for Testing

```

1 import os
2 import sys
3 import numpy as np
4
5 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
6
7 from north_west_corner import north_west_corner
8 from russel import russel
9 from vogel import vogel
10
11 test_cases = [
12     dict(
13         S=np.array([20, 30, 50], dtype=np.float32),
14         D=np.array([30, 10, 30, 30], dtype=np.float32),
15         C=np.array(
16             [
17                 [5, 4, 3, 2],
18                 [7, 6, 5, 3],
19                 [6, 5, 4, 3],
20             ],
21             dtype=np.float32,
22         ),
23         valid=True,
24     ),
25     dict(
26         S=np.array([100, 200, 300], dtype=np.float32),

```

```

27     D=np.array([50, 500, 50], dtype=np.float32),
28     C=np.array(
29         [
30             [10, 7, 8],
31             [3, 2, 4],
32             [2, 4, 6],
33         ],
34         dtype=np.float32,
35     ),
36     valid=True,
37 ),
38 dict(
39     S=np.array([10, 10, 10, 10, 10], dtype=np.float32),
40     D=np.array([10, 10, 10, 10, 10], dtype=np.float32),
41     C=np.array(
42         [
43             [2, 3, 1, 4, 5],
44             [4, 1, 2, 3, 2],
45             [3, 2, 5, 1, 2],
46             [1, 4, 3, 2, 1],
47             [2, 3, 2, 1, 4],
48         ],
49         dtype=np.float32,
50     ),
51     valid=True,
52 ),
53 dict(
54     S=np.array([50, 70, 120, 80, 60], dtype=np.float32),
55     D=np.array([100, 120, 70, 90], dtype=np.float32),
56     C=np.array(
57         [
58             [2, 3, 1, 4],
59             [4, 1, 2, 3],
60             [3, 2, 5, 1],
61             [1, 4, 3, 2],
62             [2, 3, 2, 1],
63         ],
64         dtype=np.float32,
65     ),
66     valid=True,
67 ),
68 dict(
69     S=np.array([20, 40, 10], dtype=np.float32),
70     D=np.array([30, 10, 10], dtype=np.float32),
71     C=np.array(
72         [
73             [2, 3, 1],
74             [4, 1, 2],
75             [3, 2, 5],
76         ],

```

```

77         dtype=np.float32,
78     ),
79     valid=False,
80 ),
81 ]
82
83
84 def test_no_negative_supply():
85     for test_case in test_cases:
86         S = test_case["S"]
87         D = test_case["D"]
88         C = test_case["C"]
89         valid = test_case["valid"]
90         solved, X = north_west_corner(S, C, D)
91         assert valid == solved
92         if valid:
93             assert np.allclose(S, np.sum(X, axis=1))
94             assert np.allclose(D, np.sum(X, axis=0))
95
96
97 def test_vogel():
98     for test_case in test_cases:
99         S = test_case["S"]
100         D = test_case["D"]
101         C = test_case["C"]
102         valid = test_case["valid"]
103         solved, X = vogel(S, C, D)
104         assert valid == solved
105         if valid:
106             assert np.allclose(S, np.sum(X, axis=1))
107             assert np.allclose(D, np.sum(X, axis=0))
108
109
110 def test_russel():
111     for test_case in test_cases:
112         S = test_case["S"]
113         D = test_case["D"]
114         C = test_case["C"]
115         valid = test_case["valid"]
116         solved, X = russel(S, C, D)
117         assert valid == solved
118         if valid:
119             assert np.allclose(S, np.sum(X, axis=1))
120             assert np.allclose(D, np.sum(X, axis=0))

```
