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)

$$S = \begin{bmatrix} 20 & 30 & 50 \end{bmatrix}$$
$$D = \begin{bmatrix} 30 & 10 & 30 & 30 \end{bmatrix}$$
$$C = \begin{bmatrix} 5 & 4 & 3 & 2 \\ 7 & 6 & 5 & 3 \\ 6 & 5 & 4 & 3 \end{bmatrix}$$

- Test Case 2 (valid)

$$S = \begin{bmatrix} 100 & 200 & 300 \end{bmatrix}$$
$$D = \begin{bmatrix} 50 & 500 & 50 \end{bmatrix}$$
$$C = \begin{bmatrix} 10 & 7 & 8 \\ 3 & 2 & 4 \\ 2 & 4 & 6 \end{bmatrix}$$

- Test Case 3 (valid)

$$S = \begin{bmatrix} 10 & 10 & 10 & 10 & 10 \end{bmatrix}$$
$$D = \begin{bmatrix} 10 & 10 & 10 & 10 & 10 \end{bmatrix}$$
$$C = \begin{bmatrix} 2 & 3 & 1 & 4 & 5 \\ 4 & 1 & 2 & 3 & 2 \\ 3 & 2 & 5 & 1 & 2 \\ 1 & 4 & 3 & 2 & 1 \\ 2 & 3 & 2 & 1 & 4 \end{bmatrix}$$

- Test Case 4 (valid)

$$S = \begin{bmatrix} 50 & 70 & 120 & 80 & 60 \end{bmatrix}$$
$$D = \begin{bmatrix} 100 & 120 & 70 & 90 \end{bmatrix}$$
$$C = \begin{bmatrix} 2 & 3 & 1 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 2 & 5 & 1 \\ 1 & 4 & 3 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix}$$

- Test Case 5 (invalid)

$$S = \begin{bmatrix} 20 & 40 & 10 \end{bmatrix}$$
$$D = \begin{bmatrix} 30 & 10 & 10 \end{bmatrix}$$
$$C = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 1 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

5 Code for Testing

```
import os
import sys
import numpy as np

sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

from north_west_corner import north_west_corner
from russel import russel
from vogel import vogel

test_cases = [
    dict(
        S=np.array([20, 30, 50], dtype=np.float32),
        D=np.array([30, 10, 30, 30], dtype=np.float32),
        C=np.array(
            [
                [5, 4, 3, 2],
                [7, 6, 5, 3],
                [6, 5, 4, 3],
            ],
            dtype=np.float32,
        ),
        valid=True,
    ),
    dict(
        S=np.array([100, 200, 300], dtype=np.float32),
```

```
        D=np.array([50, 500, 50], dtype=np.float32),
        C=np.array(
            [
                [10, 7, 8],
                [3, 2, 4],
                [2, 4, 6],
            ],
            dtype=np.float32,
        ),
        valid=True,
    ),
    dict(
        S=np.array([10, 10, 10, 10, 10], dtype=np.float32),
        D=np.array([10, 10, 10, 10, 10], dtype=np.float32),
        C=np.array(
            [
                [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],
            ],
            dtype=np.float32,
        ),
        valid=True,
    ),
    dict(
        S=np.array([50, 70, 120, 80, 60], dtype=np.float32),
        D=np.array([100, 120, 70, 90], dtype=np.float32),
        C=np.array(
            [
                [2, 3, 1, 4],
                [4, 1, 2, 3],
                [3, 2, 5, 1],
                [1, 4, 3, 2],
                [2, 3, 2, 1],
            ],
            dtype=np.float32,
        ),
        valid=True,
    ),
    dict(
        S=np.array([20, 40, 10], dtype=np.float32),
        D=np.array([30, 10, 10], dtype=np.float32),
        C=np.array(
            [
                [2, 3, 1],
                [4, 1, 2],
                [3, 2, 5],
            ],
```

```
            dtype=np.float32,
        ),
        valid=False,
    ),
]


def test_no_negative_supply():
    for test_case in test_cases:
        S = test_case["S"]
        D = test_case["D"]
        C = test_case["C"]
        valid = test_case["valid"]
        solved, X = north_west_corner(S, C, D)
        assert valid == solved
        if valid:
            assert np.allclose(S, np.sum(X, axis=1))
            assert np.allclose(D, np.sum(X, axis=0))


def test_vogel():
    for test_case in test_cases:
        S = test_case["S"]
        D = test_case["D"]
        C = test_case["C"]
        valid = test_case["valid"]
        solved, X = vogel(S, C, D)
        assert valid == solved
        if valid:
            assert np.allclose(S, np.sum(X, axis=1))
            assert np.allclose(D, np.sum(X, axis=0))


def test_russel():
    for test_case in test_cases:
        S = test_case["S"]
        D = test_case["D"]
        C = test_case["C"]
        valid = test_case["valid"]
        solved, X = russel(S, C, D)
        assert valid == solved
        if valid:
            assert np.allclose(S, np.sum(X, axis=1))
            assert np.allclose(D, np.sum(X, axis=0))
```