PROJECT REPORT

Transportation Problem - North West Corner Rule

ABSTRACT:

Transportation problem is an optimization problem in operations research, logistics, supply chain management and more. It involves finding the most efficient way to transport goods from a set of sources to set of destinations while minimizing the cost involved or maximizing the total profit. Our project focuses on solving the transportation problem using the method of North West Corner Rule.

The objective is to optimize the allocation while considering all the given parameters such as, supply, demand, and costs. North West corner rule provides an initial feasible solution for transportation problems. North west corner rule works by allocating values from the northwest corner of the matrix and repeating the same for the rest of the matrix until all demands and supplies are met.

We have used python and streamlit in our project. We input the relevant data that is needed to calculate the optimal solution such as, demand, supply and transportation costs. Then we apply optimization techniques to obtain the basic feasible solution.

ALGORITHM:

i. Initialize the transportation matrix (C) with the costs of transporting one unit of goods from each source to each destination.
ii. Initialize the supply (S) and demand (D) arrays with the respective supply quantities and demand quantities for each source and destination.
iii. Create an empty solution matrix (X) to store the quantities to be transported.
iv. Set the current source index (i) to 0 and the current destination index (j) to 0.
v. Repeat the following steps until all supply and demand are satisfied:
    a. Find the minimum between the supply at the current source (S[i]) and the demand at the current destination (D[j]).
    b. Assign this minimum value to the corresponding cell in the solution matrix (X[i][j]).
    c. Subtract the assigned quantity from both the supply and demand: S[i] -= assigned quantity and D[j] -= assigned quantity.
    d. If the supply at the current source becomes zero (S[i] == 0), move to the next source: i += 1.
    e. If the demand at the current destination becomes zero (D[j] == 0), move to the next destination: j += 1.
vi. Calculate the total cost by multiplying the transportation costs (C) with the solution matrix (X) element-wise and summing them.
vii. Return the solution matrix (X) and the total cost.

CODE:

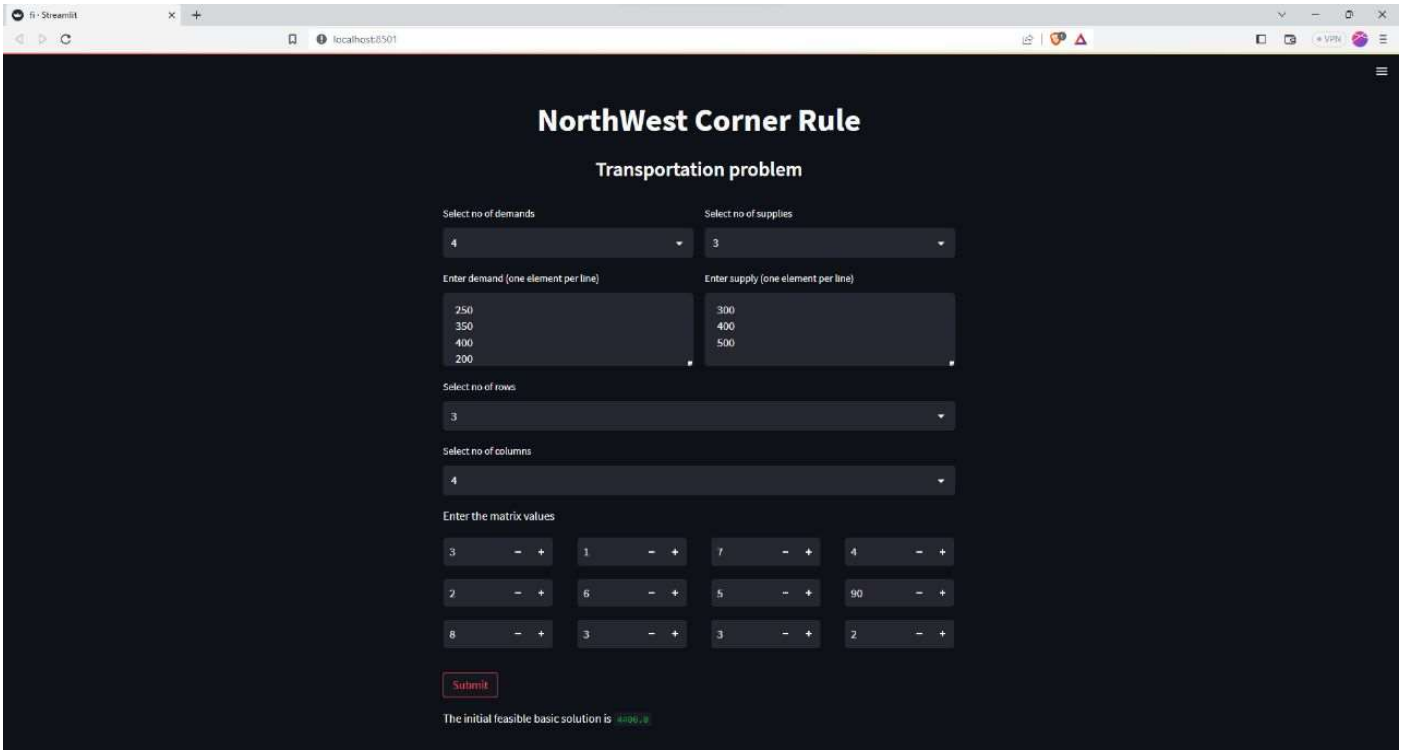
```
import streamlit as st
import numpy as np

centered_header_style = """
   <style>
   .centered-header {
      text-align: center;
   }
   </style>
"""
centered_button_style = """
   <style>
   .centered-button {
      font-size: 20px;
      padding: 10px;
      border-radius: 5px;
      display: flex;
      justify-content: center;
   }
   </style>
"""

def make_grid(col, row):
   grid = [0]*col
   for i in range(col):
      with st.container():
         grid[i] = st.columns(row, gap="medium")
   return grid

# Render centered header
st.markdown(centered_header_style, unsafe_allow_html=True)
st.markdown('<h1 class="centered-header">NorthWest Corner Rule</h1>', unsafe_allow_html=True)
st.markdown('<h3 class="centered-header">Transportation problem</h3>', unsafe_allow_html=True)
```

```
st.markdown("\n\n")

col1, col2 = st.columns(2)

# Place content in the first column
with col1:
    demand = st.selectbox('Select no of demands', ['', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], key='demand')

# Place content in the second column
with col2:
    supply = st.selectbox('Select no of supplies', ['', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], key='supply')

row = st.selectbox('Select no of rows', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], key='row')
col = st.selectbox('Select no of columns', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], key='col')

with col1:
    input1 = st.text_area("Enter demand (one element per line)")

    # Convert the input string into a NumPy array
    dema = input1.strip().split("\n")
    demarray = np.array([float(line) for line in dema if line.strip()])

with col2:
     array2 = st.text_area("Enter supply (one element per line)")
     sup = array2.strip().split("\n")
     suparray = np.array([float(line) for line in sup if line.strip()])

st.write("Enter the matrix values")
mygrid = make_grid(row, col)

l=[]
l1=[]
for i in range(row):
```

```
    for j in range(col):
        a=mygrid[i][j].number_input(":", key=(i*10+j), label_visibility="collapsed", value=0, step=0)
        l.append(a)
    l1.append(l)
    l=[]

st.markdown(centered_button_style, unsafe_allow_html=True)
if st.button('Submit'):

    # supply list
    supply=list(suparray)

    # demand list
    demand=list(demarray)

    # GRID values
    grid=l1

    startRow,startColm,solution=0

    # Check if indices are within valid range
    while(startRow<len(grid) and startColm<len(grid[0])):

        # if demand is greater than supply
        if startRow<len(supply) and supply[startRow]<=demand[startColm]:
            solution+=supply[startRow]*grid[startRow][startColm]
            # subtract the value of supply from the demand
            demand[startColm]-=supply[startRow]
            startRow+=1

        # if supply is greater than demand
        elif startColm<len(demand):
            solution+=demand[startColm]*grid[startRow][startColm]
            # subtract the value of demand from the supply
```

```
            supply[startRow]-=demand[startColm]
            startColm+=1
        else:
            break

    st.write("The initial feasible basic solution is", solution)
```

OUTPUT:



INFERENCE:

The aim of the Northwest Corner Method in transportation problems is to provide an initial feasible solution for allocating quantities from sources to destinations. It helps in establishing a starting point for the transportation problem by allocating units from the northwest corner and moving sequentially to other corners. The method aims to satisfy the supply and demand constraints while minimizing transportation costs or optimizing transportation quantities. However, it does not guarantee the optimal solution and may require further refinement using more advanced optimization techniques.