Computational analysis of Transportation Problem in MATLAB

MATH 3000 Final Project

Shelby Joji
ID: 000722012

April 30, 2018

Abstract

Optimization methods in mathematics aims at finding out the best solution to a problem from several different alternatives. Through this research project, I have investigated a special type of optimization problem called transportation problem. Objective of every transportation problem is to minimize the cost of transportation by finding a feasible solution. After surveying four different approaches, I have chosen North west corner rule and Vogel's approximation method for further analysis. Both algorithms were investigated mathematically and were implemented using MATLAB. In addition to it, a comparison of both methods was performed to further analyze the effectiveness in finding an optimal solution and thereby minimizing the cost. Algorithms were tested on 15 different transportation problems to compare the execution time. Comparisons were demonstrated in MATLAB plots.

1 Introduction

Transportation problem was first systematically formulated by an American mathematician named Frank Lauren Hitchcock[3]. During the same period, Tjalling Koopmans got involved in the same problem making the problem to be commonly referred as Hitchcock-Koopmans Transportation problem[3]. A general case of Transportation problem is as described below.

Definition 1. Consider m sources and n destinations. Let s_i be the amount of supply available at the source i and let d_j be the demand of supply required at the destination j. Assume that $s_i, d_j > 0$ and there exists a cost of transportation c_{ij} for transporting supply from source i to destination j for every i and j. In addition to it assume that the problem is balanced. Then,

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j \tag{1.1}$$

Let x_{ij} be a non-negative integer that represents the number of supply transported from source i to destination j. Under these, assumptions we can model the problem as

$$\begin{aligned} Minimize \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ subject\ to \quad & \sum_{i=1}^{m} x_{ij} = d_j \quad \ni j = 1, 2, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} = s_i \quad \ni i = 1, 2, \ldots, m \\ & x_{ij} \geq 0 \end{aligned} \tag{1.2}$$

Transportation problem has application in various industries ranging from optimizing public transportation system to formulating genetic algorithms[4]. Let us consider a simple example of general case transportation problem.

Example 1.1. Consider the case of a manufacturing industry with factories and warehouses. Each warehouse requires certain supply of items from factories. We can formulate the problem in a transportation tableau with each cell containing cost of shipping from factory i to warehouse j.

	Warehouse 1	Warehouse 2	Warehouse 3	Supply
Factory 1	2	7	4	5
Factory 2	3	3	1	8
Factory 3	5	4	7	7
Factory 4	1	6	2	14
Demand	7	9	18	34

2 Algorithms to solve transportation problem (Survey)

There are several approaches to find optimal cost transportation problem. In all such solution, we aims to find $m+n-1$ basic variables that satisfies all the constraints. Following are the algorithms to solve general case of a transportation problem

1. The North West Corner Rule (NWCR): In this method, there is exactly m+n-1 basic variables as solution.NWCR is simple and easy to implement. Method starts form the north west cell of tableau to assign basic variables and a row or column is increased with each iterations completing the required n umber of basic solutions. One disadvantage of this method is that it is not cost sensitive thereby providing poor solutions ans higher optimal costs[1].

2. Least cost Method (LCM): As the name states, LCM depend minimizing on the unit cost of transportation at each step. The algorithm calculates feasible solution by assigning as much possible supply on the

cheapest route available. Once the assignment is made, it moves to the next cheapest route after updating the demand and supply. This method repeats until there exits no more supply or demand. Since LCM focuses on cost, we obtain better accurate solution compared to NWCR[8]. When multiple routes are found, LCM tend to break ties arbitrarily which result in no so accurate solutions[7].

3. Vogel's Approximation Method (VAM): In a sense, VAM is an improved version of LCM and not computationally simple compared to LCM. Approach is to calculate the penalties of row and column by taking the difference of lowest among the two minimum cost.In each iterations, cell is chosen by selecting largest penalties to ensure least cost. Therefore, VAM ensures minimum cost than LCM[2]. As complexity of computations involved increases, VAM consumes more time to deliver the result,

4. Best Candidate Method (BCM): Compared to VAM, BCM provides solution in least computational time and the optimal solution is at least better than VAM. It makes uses of fewer iteration and thereby saving computational time. This method find best candidate having best combination of cost and supply for each row and column. Once the assignment is made, BCM looks for next best combination until all assignments are made[4].

3 Two Algorithms

3.1 The North West Corner Rule (NWCR)

Northwest corner rule is one of the basic and simple approach in solving a general case transportation problem. This method is useful for finding the initial basic feasible solution. The first cell or the northwest cell in the cost matrix is selected as the first assignment to allocate supplies[1].

- Select the Northwest cell ij where $i = 1$ and $j = 1$
- Assign as many supplies from source i to destination $j \ni x_{ij} = min\{s_i, d_j\}$
- Update the supplies at the source s_i as $s_i = s_i - x_{ij}$ and destinations as $d_j = d_j - x_{ij}$

- If $s_i = 0$, then $i = i + 1$ and if $d_j = 0$, then $j = j + 1$. Repeat the set for the next cell.

Once the assignment is complete, there will be exactly m+n-1 basic feasible solution. As we have seen in the above algorithm, NCWR does not consider cost of transportation at any point which often results in poor solution and higher costs[6]. In fact NCWR serves as the starting point for more advance methods like Modified distribution algorithms (MODI)[6].

```
Algorithm Northwest Corner rule is
FUNCTION Input: 2D Cost matrix c with
                a number of source,
                b number of destination,
                1D source matrix s, and
                1D destination matrix d
        Output: Optimal cost z and
                2D Basic feasible solution matrix x

define x as matrix with a columns and b rows
for each i from 1 to a
    for each j from 1 to b
        while supply and demand at cell i and j is
               larger than zero check
               if demand is larger than supply
                assign all supplies to x(ij)
                d(j) <- d(j)-s(i)
                set s(i)as zero;

               else if supply is larger than demand
                assign all demands to x(ij)
                s(i)<-s(i)-d(j)
                set d(j)as zero;
               else
                set x(ij) as either supply or demand
                set d(j) and s(i)as zero
               endif
          end while
       end for
end for
return x and z as the product of c and x
end function
```

Figure 1: Pesudo-code for North West Corner Rule

3.2 Vogel's Approximation Method (VAM)

VAM was proposed by W.R. Vogel in 1958 which is an improved version of minimum cost method[5]. This is considered as one of the best approaches in finding initial basic feasible solutions which obtain approximate optimal solutions. VAM uses the following steps.

- Calculate penalties for each row and column. Penalties are calculated by finding the difference between lowest and next lowest cost for each row and column.VAM allows to select same costs multiple times. Suppose h and k are cell number for the respective row and column. Let P_i be row penalty and P_j be column penalty. Then,

$$P_i = |c_{ih} - c_{ik}|$$

$$P_j = |c_{hj} - c_{kj}|$$

- Find the lowest cost among all $max\{p_i, p_j\}$. If it is not unique find the cell with maximum allocation of $min\{s_i, d_j\}$.

- Once the assignment is complete, adjust the s_i and d_j. If the assignment is complete in either row or column, remove it from the cost matrix and repeat procedure until m+n-1 iterations are complete.

```
Algorithm Vogels approximation method is
FUNCTION Input: 2D Cost matrix c with
                a number of source,
                b number of destination,
                1D source matrix s, and
                1D destination matrix d
        Output: Optimal cost z and
                2D Basic feasible solution matrix x

define x as zero matrix with a columns and b rows
create a copy of c
for each iteration from 1 to a+b-1 do

    %calculate minimum cost of rows
    Define 2 zero column matrix of length a and
           row matrix of length a for position
    for each i from 1 to a
       define minimum as infinity value
       for each j from 1 to b
        if c(i,j)< minimum
            set minimum as the cost cell
            assign position to position matrix
        endif
      end for
      assign the ith row of column matrix as minimum value
    end for

    %calculate the second minimum cost of rows
    for each i from 1 to a
       define minimum as infinity value
       for each j from 1 to b
          if j not equal to ith position of position matrix
            check if c(i,j)< minimum set minimum as the cost cell
            endif
          endif
      end for
    end for
      assign the ith row of column matrix as minimum value
    end

    Repeat the same Algorithm to obtain minimum and
     second minimum cost of columns

    %calculate penalty on row and column
    Set row penalty rp as zero matrix of length a
    Set column penalty cp as zero matrix of length b but transposed
    Set rp as difference between of column matrix for row
    Set cp as difference between of the computed row matrix for column

%find the maximum value of row penalty and its index
    Set maximum row penalty Mrp as zero
    for each i from 1 to a
        check if ith rp >= Mrp
            Set Mrp as rp at this position and
            save index position to Irp
        endif
    end for

    Repeat the Algorithm to find the maximum value of column
    Penalty Mcp and its index Icp
    Define I and signal as zero
    oc=c(:,Icp);
    or=c(Irp,:);
    % if column has larger or equal penality
    Check if Mcp>=Mrp
         Find the Ith position of minimum cost in the column Icp
         If supply is larger than demand
               assign all demand to x(I,Icp)
               s(I) <- s(I)-d(Icp)
               set d(Icp)as zero
               signal to remove columns
         else if demand is larger than supply
               assign all supply to x(I,Icp)
               d(Icp) <- d(Icp)-s(I)
               set s(I)as zero
               signal to remove rows
         else
               assign all supply to x(I,Icp)
               set d(Icp)as 0
               set s(I)as zero
               signal to remove rows and columns
        end if
        %next elimination is performed
        if signal is to remove columns
            assign column Icp of cost matrix as infinity
        else if signal is to remove rows
            assign row I of cost matrix as infinity
        else if signal to remove rows and columns
            assign column Icp and row I of cost matrix as infinity
        end if
    Check else if Find the Ith position of minimum cost in the row Irp
          Repeat the Algorithm to assign values to x and
          remove elements from cost matrix
    end if
end for
return x and z as the product of c and x
end function
```

Figure 2: Pesudo-code for Vogel's Approximation Method

One drawback of Vogel's algorithm is that it does not ensure lowest cost in degenerate cases where there are similar higher penalties[2]. It can be improved by finding logical costs instead of settling for the first minimum cost but this tend to increases the complexity and run time of the problem.

4 Comparison

4.1 Analytical comparison

As we have seen in the algorithms, NCWR does not consider cost matrix in computing the basic feasible solution. Whereas VAM allocates to least cost route by finding penalties.Therefore NCWR must yield higher optimal cost compared to VAM.

Table 1: Implementation of algorithms on example 1.1

(a) NCWR

2	7	4	
5			5
3	3	1	
2	6		8
5	4	7	
	3	4	7
1	6	2	
		14	14
7	9	18	34

(b) VAM

2	7	4	
3	2		5
3	3	1	
		8	8
5	4	7	
	7		7
1	6	2	
4		10	14
7	9	18	34

On implementing both methods on the example 1.1, we obtain results as described in Table 1. With NCWR, we obtain an optimal cost of 102 meanwhile VAM yields a better solution with an optimal cost of 80.

4.2 Run-time Comparison

I ran both NCWR and VAM on 15 randomly created general case transportation problem. We obtain the following results as plotted in MATLAB plot (Figure) given below.

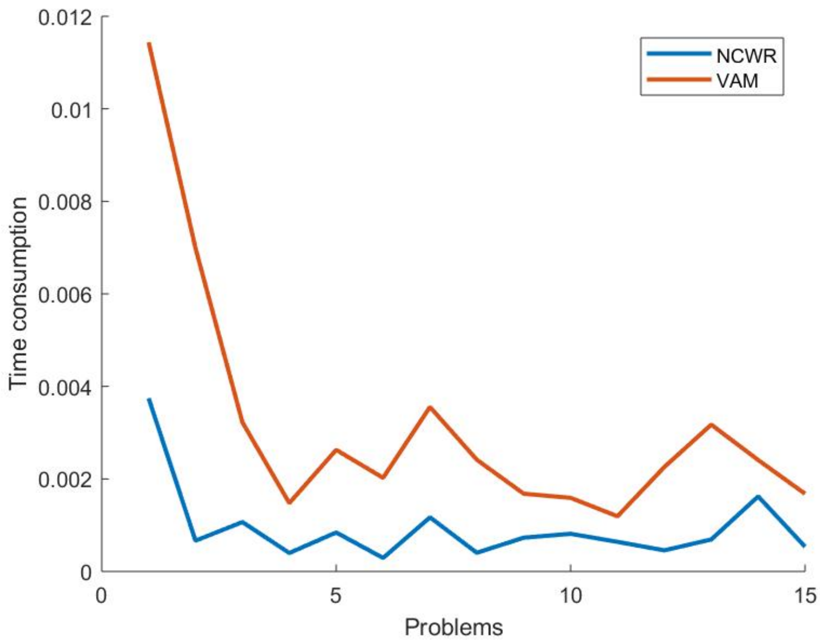

Figure 3: Runtime comparison of Northwest Corner Rule(NCWR) and Vogel's Approximation Method (VAM)

References

[1] M S. Bazaraa, J J Jarvis, and H D Sherali, *The transportation and assignment problems*, Wiley-Blackwell, 2011.

[2] Utpal Das, Md. Ashraful Babu, Aminur Khan, Md Abu Helal, and Md Uddin, *Logical development of vogel's approximation method (ld-vam): An approach to find basic feasible solution of transportation problem* **Volume 3** (201402), pp.42–48.

[3] D R Fulkerson, *Hitchcock transportation problem*, 1956. An optional note.

[4] A Hlayel and M A. Alia, *Solving transportation problems using the best candidates method* **2** (201210), 23–30.

[5] Z Juman, Hoque M, and M Buhari, *A study of transportation problem and use of object oriented programming*, 3rd International Conference on Applied Mathematics and Pharmaceutical Sciences (201304), 353–354.

[6] Bettina Klinz and Gerhard J. Woeginger, *The northwest corner rule revisited* **159** (201107), 1284–1289.

[7] H A Taha, *Operations research an introduction*, 8th ed., Pearson Education, Inc., Upper Saddle River, NJ, 07458, 2007.

[8] Md Uddin, Aminur Khan, Chowdhury Kibria, and Iliyana Raeva, *Improved least cost method to obtain a better ibfs to the transportation problem* **6** (201601), 1–20.

Appendices

A Matlab Code for North West Corner Rule

```
function [x,y]= nwcf(a,s,b,d,c)
x=zeros(a,b);
for i= 1:a
    for j= 1:b
        while s(i)>0 && d(j)>0 % we don't want loop to check if allocation is complete
            if d(j)>s(i)
                x(i,j)=s(i);
                d(j)=d(j)-s(i);
                s(i)=0;

            elseif s(i)>d(j)
                x(i,j)=d(j) ;
                s(i)=s(i)-d(j);
                d(j)=0;
            else
                x(i,j)=d(j);
                d(j)=0;
                s(i)=0;
            end

        end

    end
end
y=sum(sum(c.*x));
end
```

Figure 4: Matlab implementation of NWCR

B Matlab Code for Vogel's Approximation Method

```matlab
function[x,z]=vsam(a,s,b,d,c1)
x=zeros(a,b);
c=c1;
for k=1:a+b-1

    %row difference
    %define array to hold the least two minimum numbers in the row and array to hold the position of minimum
    mr1=zeros(a,1);
    mr2=mr1;
    mc1=zeros(1,a);
    for i=1:a
      minr=inf;
      for j=1:b % finding minimum in the ith row and its position
        if c(i,j)<minr
            minr=c(i,j);
            mc1(i)=j; % we get the position of lowest element
        end
      end
      mr1(i)=minr;
    end
    clear i;
    clear j;
    %Since we get the position of lowest element, we can find the next lowest
    for i=1:a
      minr=inf;
      for j=1:b % finding minimum in the ith row and its position
          if j~=mc1(i) %if position of next lowest element is not the lowest
            if c(i,j)<minr
                minr=c(i,j);
            end
          end
      end
      mr2(i)=minr;
    end

    %column difference
    %define array for the least two minimum numbers in the column and another array to hold the position of minimum in each column
    mc1=zeros(1,b);
    mc2=mc1;
    posr=zeros(1,b);
    for i=1:b
      minc=inf;
      for j=1:a % loop to find minimum in the ith column and its position
        if c(j,i)<minc
            minc=c(j,i);
            posr(i)=j; % we get the position of lowest element
        end
      end
      mc1(i)=minc;
    end

    %Since we get the position of lowest element, we can find the next
    %lowest in the column
    for i=1:b
      minc=inf;
      for j=1:a % finding minimum in the ith column and its position
          if j~=posr(i) %if position of next lowest element is not the lowest
            if c(j,i)<minc
                minc=c(j,i);
            end
          end
      end
      mc2(i)=minc;
    end

    %penalty
    rp=zeros(1,a);
    cp=zeros(b,1);
    rp=abs(mr2-mr1);%row penalty
    cp=abs(mc2-mc1);%coulmn penalty

    %find the maximum value of row penalty and its index


    Mrp=0;
    for i=1:a
        if rp(i)>=Mrp
            Mrp=rp(i);
            Irp=i;
        end
    end

    %largest row penalty R and index iminrow
    Mcp=0;
    for j=1:b
        if cp(j)>=Mcp
            Mcp=cp(j);
            Icp=j;
        end
    end

    I=0;
    M=0;
    signal=0;
    oc=c(:,Icp);
    or=c(Irp,:);
    % if column has larger or equal penality
    if Mcp>=Mrp
        [M,I]=min(oc);
        if s(I)>d(Icp)
             x(I,Icp)=d(Icp);
             s(I)=s(I)-d(Icp);
             d(Icp)=0;
             Signal=0;% column needs to be eliminated
        elseif s(I)<d(Icp)
             x(I,Icp)=s(I);
             d(Icp)=d(Icp)-s(I);
             s(I)=0;
             signal=1;% row needs to be eliminated
        else
             x(I,Icp)=d(Icp);
             s(I)=0;
             d(Icp)=0;
             signal=2;% row and column needs to be eliminated
        end
        %next elimination is performed
        if signal==0
            c(:,Icp)=inf;
        elseif signal==1
            c(I,:)=inf;
        elseif signal==2
            c(:,Icp)=inf;
            c(I,:)=inf;
        end

    else%hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

        [M,I]=min(or);
        if s(Irp)>d(I)
             x(Irp,I)=d(I);
             s(Irp)=s(Irp)-d(I);
             d(I)=0;
             Signal=0;% column needs to be eliminated
        elseif s(Irp)<d(I)
             x(Irp,I)=s(Irp);
             d(I)=d(I)-s(Irp);
             s(Irp)=0;
             signal=1;% row needs to be eliminated
        else
             x(Irp,I)=d(I);
             s(Irp)=0;
             d(I)=0;
             signal=2;% row and column needs to be eliminated
        end
        %next elimination is performed
        if signal==0
            c(:,I)=inf;
        elseif signal==1
            c(Irp,:)=inf;
        elseif signal==2
            c(:,I)=inf;
            c(Irp,:)=inf;
        end
    end
end
z=sum(sum(c1.*x));
end
```

Figure 5: Matlab implementation of VAM