Chapter 5

Sensitivity Analysis of Assignment Problem
Chapter 5
Sensitivity Analysis of Assignment Problem

5.1 Introduction

The assignment problem (AP) is one of the most studied, well solved and important problems in mathematical programming. It is of great importance for its application in manufacturing, education, planning etc. It is well known that AP is special case of linear programming problem. Let there be \( n \) resources which are to be assigned to \( n \) activities with resource capacity and activity equal to unity each. The problem consist in determining the assignment of resources to activities so as to minimize the overall cost / time in such a way that each resource can associate with one and only one activity. Let the cost of assigning the \( i^{th} \) resource to the \( j^{th} \) activity be \( c_{ij} \). Let \( x_{ij} \) denote the assignment of the \( i^{th} \) resource to the \( j^{th} \) activity. The AP is mathematically defined as

\[
\text{Min } z = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij} \tag{5.1.1}
\]

s.t.

\[
\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, 2, \ldots, n \tag{5.1.2}
\]

\[
\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n \tag{5.1.3}
\]

and \( x_{ij} = 0 \) or \( 1 \) for all \( i, j = 1, 2, \ldots, n \) \tag{5.1.4}

5.2 Solution Approaches

In this chapter we study the solution approach and the methodology used in various approaches for solving the AP.

5.2.1 Simplex Method Approach

Due to it's totally unimodular property of constraint matrix, an AP can be treated both as an LP, and also as a 0-1 integer programming problem and a network problem. Some particular algorithms have been developed which can solve an AP very efficiently. Almost all of these algorithms proceed directly on the cost matrix \( C \) where, without loss of generality, we assume \( c_{ij} \geq 0 \). After obtaining the optimal solution, most decision-maker performs sensitivity analysis or post optimality analysis. In an LP, the objective of sensitivity analysis is to analyze the effect of the changes of the objective function coefficients (ofc) and the effect of changes of right-hand-side (rhs) elements on
the optimal value of the objective function, as well as the validity ranges of these effects. The implementation of sensitivity analysis is based primarily on basic optimal solutions. In contrast, degeneracy is a complex phenomenon that may negatively influence the computation of optimal basic solutions and sensitivity analysis[27]. Unfortunately, AP is inherently an LP model with a high degree of primal degeneracy, and a primal degenerate optimal extreme point usually corresponds to several different optimal bases. Therefore, it may be tedious to determine the ranges of parameter values for which a given optimal assignment remains optimal for an AP. In fact, Gal [20] pointed out that when degeneracy occurs, the traditional sensitivity analysis approach in textbooks or commercial LP-packages to determine sensitivity ranges of parameters yields impractical results. Apparently though AP can be solved by Simplex and its variants, the traditional sensitivity analysis is not suitable for AP.

5.2.2 Network Simplex

Approach here is to adapt the simplex network optimization through network simplex method. This provides unification of the various problems but maintains all inefficiencies of the simplex, such as pivotal degeneracy, as well as inflexibility to handle side constraints. Even ordinary, one change at a time, ordinary sensitivity analysis limits, long available in the simplex, are not easily available in network simplex[1].

5.2.3 TP Algorithm

As AP is a sub class of a standard TP, the TP algorithms can be used to find the optimal solutions. However, the transportation algorithm is not generally recommended for solving the assignment problem due to presence of degeneracy in every basic feasible solution. The degeneracy can lead to nonproductive basis changes and possibly cycling in the transportation algorithm.[32].

5.2.4 Specialized Algorithm

In other approaches in solving the AP consist in using specialized algorithms based on special structure of a specific problem. These solution algorithms are not unified as each algorithm uses a different strategy to exploit the special structure of a specific problem. As a result small variation in problem structure renders the algorithm unusable for obtaining the solution to other problems. Also these algorithms do not provide the sufficient information necessary for performing the post optimality analysis of AP[1].
5.2.5 Hungarian Assignment Algorithm

Hungarian Assignment is widely used algorithm to solve AP, although efficient one but it suffers with drawback, that, it is not easy draw the number of lines to cover zeros and secondly, it lacks information necessary for performing post optimality analysis. The changes in cost parameters are very important from AP point of view and the Hungarian algorithm in fact lacks the ability to test the validity of the current optimal solution with changes in cost parameters without resolving the problem.

5.3 Assignment algorithm

We first present and discuss the basic principal and the algorithm for Hungarian method.

5.3.1 Basic Principle

The AP is special case of LPP and is traditionally solved using the Hungarian method due to D. König. The basic principle of the method is that the optimal assignment is not affected if a constant is added or subtracted from any row or column of the assignment cost matrix. In essence the solution procedure is to subtract a sufficiently large cost from the various rows or columns in such a way that an optimal assignment is found by inspection. The algorithm is initiated by examining each row/column of the cost matrix to identify the smallest element; this is then subtracted from all the elements in that row/column. This produces a cost matrix containing at least one zero element in each row/column. Now the feasible assignment is made using cells with zero costs and if it is possible then we have an optimal assignment. This is because the cost elements \( c_{ij} \) are nonnegative and the minimum value of the objective function cannot be less than zero. Hence, an assignment with zero cost has to be optimal.

5.3.2 Hungarian Algorithm

This is most widely used and popular algorithm for solving AP. The various steps of the Hungarian Assignment algorithm are

1. Determine the cost table from the given problem.
2. Locate the smallest element in each row of the given cost matrix and then subtract the same from each element of that row.
3. For the modified matrix obtained from step 2, locate the smallest element of each column and then subtract the same from each element of that column.
4. As a result of step 3 and 4 each row and column have at least one zero element. Search for an optimal assignment as follows
(a) Examine the rows successively until a row with a single zero is found. Encircle this zero and cross off all other zeros in its column. Continue in this manner until all the rows have been taken care of.

(b) Repeat the procedure for each column of the modified matrix.

(c) If a row and/or column has two or more zeros and one cannot be chosen by inspection then assign arbitrary any one of these zeros and cross off all other zeros of that row/column.

(d) Repeat (a) through (c) above successively until the chain of encircling zeros and crossing off ends.

5. If the number of assignment is equal to \( n \) i.e. the order of cost matrix, an optimum solution is reached. However, if the number of assignment is less than \( n \) then go to next step.

6. Draw the minimum number of horizontal and/or vertical lines to cover all the zeros of the reduced matrix.

7. Find the smallest element of the reduced matrix not covered by any of the lines. Subtract this element from all uncovered element and add the same to all the elements lying at the intersection of any tow lines.

8. Goto step 5 and repeat this procedure until an optimum solution is attained.

9. Stop.

Even though the Hungarian method is an efficient method to solve the AP, to draw minimum number of lines to cover all the zeros in the reduced cost matrix is not an easy task and also it does not provide useful information to perform cost sensitivity analysis to a decision maker.

5.4 Push-and-Pull Method

From the point of view of sensitivity analysis the best approach would be that of Simplex method. Simplex method has necessary information required for the sensitivity analysis in form of basis inverse matrix but as the AP solutions are highly degenerate, the Simplex approach will provide very misleading sensitivity analysis for the AP. TP algorithm also suffers from the same drawback. In other approaches the final solution lacks the necessary information or extra computational work needs to be done for performing the sensitivity analysis of AP. As pointed out by Adlakha V. and Arsham H. [6], the literature still lacks managing cost uncertainties for a degenerate final solution which is very common in AP problems as well as transportation problem. In view of this Adlakha V. and Arsham H. [1] proposed a single unified algorithm that solves both
Assignment problem as well as Transportation problem providing useful information to perform cost sensitivity analysis to a decision maker. The Simplex like algorithm is presented, discussed and illustrated with numerical example.

5.4.1 The Solution Algorithm

The Push-and-Pull algorithm is a Simplex type pivotal algorithm for general solution of AP as well as TP. The name identifies the two main strategies that are adopted in this approach to reach the optimum solution avoiding the use of slack, surplus or artificial variables commonly used in the Simplex method. The algorithm works in four phases, which are 1) Cost matrix reduction 2) Initialization phase 3) Push phase 4) Pull phase. Non iterative phase one and two takes care of initialization preliminaries and iterative phases three and four pushes and if necessary pulls back towards optimal vertex. The working of each major phase is presented.

The matrix reduction phase helps in identifying lowest cost variables which are ready to enter the basis and also decides the redundant constrains to be eliminated and thus facilitates a good start for the initialization phase.

The elimination of the redundant constraint in earlier phase helps in generating unit column vectors which can be readily entered into basis variable set, partially filling up basis variable set without any iterations or computations. This phase generates an initial tableau, which contains some basic variables; the columns corresponding to these variables are deleted. Hence, this phase does the job of providing good platform to start the iterative process with an optimized initial tableau.

The purpose of push phase is to complete the partially filled basic variable set generated in initialization phase. The filling of basic variable set is done using the known simplex like criterion for entering variable while maintaining the optimality condition. At this stage variables are only pushed to the basic variable set without any replacement. Thus, in this phase, pushing towards optimum vertex by maintaining feasibility and optimality as much as possible is achieved. In this sense the push phase is a basic variable set augmentation process that develops a basic solution, which may or may not be feasible and if feasible then it is optimal.

The strategy used in earlier phase uses modified simplex entering variable criterion to enter one variable at a time into an open row, rather than replacing a variable, while moving toward a vertex that is close to the optimal vertex. This strategy pushes towards an optimal solution, which may result in pushing too far into non-feasibility. The last phase, if needed pulls the current infeasible solution back towards
feasibility while maintaining optimality condition using the famous dual simplex
criterion. This method in fact is fine blend of primal and dual Simplex criterion.

To implement the algorithm to obtain optimum solution for the given AP the
problem need to be presented in Simplex like tableau as displayed below.

<table>
<thead>
<tr>
<th>Basis Column</th>
<th>Columns for Coefficients of Constraints</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis Variable</td>
<td>Non Basic Variables</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>Reduced Cost coefficients</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.1.1

5.4.2 Push-and-Pull Pivotal Algorithm

The Simplex type pivotal algorithm to solve AP using following terminology is
presented

GJP; Gauss-Jordan pivoting.

BV; basic variable.

BVS; basic variable set.

NB; non-basic variable.

PR; pivot row.

PC; pivot column.

PE; pivot element.

RHS; right hand side.

C/R; column ratio given by RHS element / Pivot Column element.

OR; open row, a row not yet assigned to a BV and

? ; label for an open row.

Various steps of Push and Pull algorithm to find optimal solution are provided
below

Step 1 Reduced cost matrix

1.1 Determine the smallest cost from each row and subtract it from the elements of
the respective row. Now, determine the smallest cost from each column and
subtract the same from the respective column. Accumulate the effect of row and
column reductions into the base cost.

1.2 Identify the row/s and/or column/s with the most zeros in the reduced cost
matrix. Eliminate the corresponding constraint/s.
Step 2 Initialization phase
2.1 Set up the initial simplex tableau using a row for each constraint and a column for each variable. Enter reduced $c_{ij}$'s in the last row of tableau under corresponding columns and negative entry for base cost under the RHS column.

2.2 Identify unit vector column and label the row containing entry ‘1’ with the name of the variable for the column. Label remaining rows as open rows by placing? mark.

2.3 Delete column corresponding to Basic Variable from the tableau.

Step 3 Push phase
3.1 Perform BVS iteration termination test as under If open row (?) exists then continue to bring in basic variables by proceeding to step 3.2 otherwise proceed to Pull phase.

3.2 Selection of Basic variable
3.2.1 Find smallest cost $c_{ij}$ from last row, the column corresponding to smallest $c_{ij}$ is pivot column (PC). Resolve ties arbitrarily, preferably consider left most $c_{ij}$

3.2.2 Select the open rows as candidate rows where variable corresponding to PC can be entered.

3.2.3 Find the ratios as rhs entry for open row / Entry in the PC column
3.2.4 Choose the minimum but positive ratio corresponding to open row. If minimum positive ratios do not exist then choose the ratio with the smallest absolute value.

3.2.5 Identify the element corresponding to intersection of pivot column and pivot row as Pivot Element (PE). If the pivot element is zero then select the next best $c_{ij}$

3.2.6 Replace the open row label with entering variable name
3.2.7 Obtain new tableau elements performing pivoting operations. Remove the pivot column from the tableau. Loop back to step (3.1)

Step 4 Pull Phase (Feasibility phase)
4.1 Perform Iteration termination test. If RHS $\geq 0$, the tableau is optimal, and hence proceed to step (4.4) otherwise proceed to (4.2)

4.2 Perform Selection of Pivot Element
4.2.1 Determine row with the most negative RHS and denote it as pivot row (PR).
4.2.2. Identify column with a negative element in pivot row.

4.2.3. Determine ratios $c_{ij}$/ negative elements in the pivot row. If more than one column has –ve entry then choose the column with smallest $c_{ij}$ (If tie still exist then resolve it arbitrarily). Label the column as pivot column (PC).

4.2.4. Element at intersection of PR and PC is pivot element (PE)

4.3 Perform Pivoting operations.

4.3.1. Save the pivot column

4.3.2. Perform pivoting operations using PE.

4.3.3. Replace variable corresponding to PR with that of PC.

4.3.4. Replace new PC with old pivot column.

4.3.5. Loop-back to step (4.1)

4.4 Identify Optimal Assignment Plan

4.4.1. Identify all $x_{ij} = 1$ from the basic variable set and prepare the optimal assignment plan accordingly

The proposed algorithm provides a clear indication of the presence of alternate optimal solutions upon termination. Clearly, different alternate solutions give the same cost. The decision maker, however, has the option of deciding which optimal solution to implement on the basis of all the other factors involved. To identify the presence of alternate optimal solution, inspect the cost row at the optimum tableau, an entry of zero in the cost row corresponding to non basic variable indicates the variable to enter the basis to provide an alternate optimal solution.

Ex:5.1. Consider the assignment problem of assigning tasks to men on one to one basis with following cost matrix.

<table>
<thead>
<tr>
<th>Men</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>18</td>
<td>26</td>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>B</td>
<td>13</td>
<td>28</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>C</td>
<td>38</td>
<td>19</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>D</td>
<td>19</td>
<td>26</td>
<td>24</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5.4.2.1

Consider the 4 x 4 assignment problem above. Application of step 1.1 i.e. row reduction yields
Table 5.4.2.2

Selection of column minimum and subtraction from respective column yields

<table>
<thead>
<tr>
<th>Task</th>
<th>Men</th>
<th>M₁</th>
<th>M₂</th>
<th>M₃</th>
<th>M₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>15</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>23</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>16</td>
<td>14</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.2.3

Now, we apply step 1.2 of Push and Pull algorithm to decide elimination of unwanted constraints. According to step 1.2, amongst the rows and column, maximum zeros (3 in number) occur in the last column and hence last constraint should be deleted, according to algorithm. We slightly deviate from this rule and select row 3 and 4 which has two zeros each. The reason is, if, we delete last constraint then only three basic variables are available at outset in bvs, however, if we make choice of 3rd and 4th constraint for deletion then 4 basic variables are available at outset in bvs which results in reducing pivoting operations required to enter the variable in bvs. The reduced cost cᵢⱼ's are entered in cost row and the sum of reductions from row and column as (11+13+15+10+4+1=54) as negative entry under rhs column is entered. The initial table after incorporating this change in step 1.2 is shown below.

Table 5.4.2.4

<table>
<thead>
<tr>
<th>bvs</th>
<th>x₁₁</th>
<th>x₁₂</th>
<th>x₁₃</th>
<th>x₁₄</th>
<th>x₂₁</th>
<th>x₂₂</th>
<th>x₂₃</th>
<th>x₂₄</th>
<th>x₃₁</th>
<th>x₃₂</th>
<th>x₃₃</th>
<th>x₃₄</th>
<th>x₄₁</th>
<th>x₄₂</th>
<th>x₄₃</th>
<th>x₄₄</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>13</td>
<td>23</td>
<td>0</td>
<td>9</td>
<td>12</td>
<td>13</td>
<td>0</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

129
Deletion of 2nd and 3rd constraint results in unit column vectors corresponding to variables x_{21}, x_{23}, x_{32}, x_{34}. Now application of step 2.2 results in inclusion of x_{21}, x_{23}, x_{32}, x_{34} at 3rd, 4th, 5th and 6th row respectively in bvs. Row 1 and 2 are open row and hence filled with ? label. Now, application of 2.3 results in deletion of columns corresponding to variables x_{31}, x_{33}, x_{32}, and x_{34} from the initial table. Initial phase ends here. We note that, calculations are not needed in obtaining the basic variables, thus a procedure that helps in inclusion of more number of basic variables in the basic variable set during initialization phase will reduce the number of iterations required to reach the optimum solution. Reduced initial table is shown below.

<table>
<thead>
<tr>
<th>bvs</th>
<th>x_{11}</th>
<th>x_{12}</th>
<th>x_{13}</th>
<th>x_{14}</th>
<th>x_{21}</th>
<th>x_{22}</th>
<th>x_{24}</th>
<th>x_{31}</th>
<th>x_{33}</th>
<th>x_{34}</th>
<th>x_{41}</th>
<th>x_{42}</th>
<th>x_{43}</th>
<th>x_{44}</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>[1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>i</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x_{21}</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{22}</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{23}</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{24}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>11</td>
<td>13</td>
<td>23</td>
<td>2</td>
<td>9</td>
<td>12</td>
<td>13</td>
<td>0</td>
<td></td>
<td>-54</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.2.5

We now enter the push phase, open rows are checked as per step 3.1, two open rows are there. Selection of variable x_{14} to enter the bvs at first row is made in accordance with step 3.2, pivoting operations are performed according to step 3.3 and bvs is augmented with basic variable x_{14} and column corresponding to x_{14} deleted, which gives the following reduced table.

<table>
<thead>
<tr>
<th>bvs</th>
<th>x_{11}</th>
<th>x_{12}</th>
<th>x_{13}</th>
<th>x_{14}</th>
<th>x_{21}</th>
<th>x_{22}</th>
<th>x_{24}</th>
<th>x_{31}</th>
<th>x_{33}</th>
<th>x_{34}</th>
<th>x_{41}</th>
<th>x_{42}</th>
<th>x_{43}</th>
<th>x_{44}</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_{14}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>[1]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x_{21}</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{22}</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{23}</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_{24}</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c_{4}</td>
<td>7</td>
<td>11</td>
<td>5</td>
<td>11</td>
<td>13</td>
<td>23</td>
<td>2</td>
<td>9</td>
<td>12</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
<td>-54</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.2.6

Row 2 is still open row. Now, application of 3.2 results in selection of x_{44} to enter at open row and application of 3.3 augments x_{14} to bvs with deletion of corresponding column from the table. The resulting table is shown below.
The basic variable set is now completely filled and hence push phase ends and we observe that rhs column contains negative value corresponding to variable $x_{34}$ and according to iteration termination test (step 4.1) , we need to proceed to step 4.2. According to step 4.2 most negative rhs entry occurs at last row corresponding to variable $x_{34}$ and hence we find ratios $c_j / a_{ij}$ for all $a_{ij} < 0$ from last row as $(-7/1, -11/1, -5/1, -9/1, -12/1, -13/1)$. Now the smallest ratio corresponds to column $x_{13}$ which is -5, this column is pivot column and so it is saved according to step 4.3. Now pivoting operations are performed, variables $x_{34}$ and $x_{13}$ are swapped and saved column $x_{13}$ is restored under label $x_{34}$ as shown under , to provide the following table.

<table>
<thead>
<tr>
<th>bvs</th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$x_{13}$</th>
<th>$x_{21}$</th>
<th>$x_{22}$</th>
<th>$x_{23}$</th>
<th>$x_{24}$</th>
<th>$x_{31}$</th>
<th>$x_{32}$</th>
<th>$x_{33}$</th>
<th>$x_{34}$</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{14}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$x_{44}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$x_{21}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$x_{22}$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$x_{23}$</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>$x_{34}$</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>5</td>
<td>11</td>
<td>13</td>
<td>23</td>
<td>2</td>
<td>9</td>
<td>12</td>
<td>13</td>
<td>-54</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4.2.7

We, now, observe that bvs is complete as well as rhs column has positive values and hence optimum solution is reached. The optimum solution is $x_{13}=1, x_{23}=1, x_{33}=1, x_{44}=1$ and hence the assignment plan according to optimal solution is assign task A to $M_3$, task B to $M_1$, task C to $M_2$ and task D to $M_4$. The optimum cost involved according
to this assignment plan is \( 17 + 13 + 19 + 10 = 59 \). Observe that optimum solution is primal degenerate as \( x_{33} \) and \( x_{14} \) equal to 0.

5.5 Computational Complexity of Push-and-Pull Algorithm

During any step of the algorithm the Simplex like table used in the algorithm has entries as 1, 0, -1, except \( c_i \) entries occurring in the last row. This fact can be systematically utilized to reduce the amount of pivoting computations during the push and pull phase of the algorithm. Let us denote the table entries in the \( j^{th} \) row and \( k^{th} \) column by \( a_{jk} \). Let the pivot element \( a_{ik} \) be at intersection of \( i^{th} \) row and \( k^{th} \) column, then the entry in the new table denoted by \( \hat{a}_{ij} \) is given by \( \hat{a}_{ij} = a_{ij} - (a_{ij} \cdot a_{ik}) / a_{ik} \). Now, as per rules of bfs augmentation phase or pull phase \( a_{ik} \) cannot be zero. Hence, the entry \( \hat{a}_{ij} \) will change only when \( a_{ij} \cdot a_{ik} \neq 0 \) i.e. if \( a_{ij} \) or \( a_{ik} \) is zero then the \( \hat{a}_{ij} \) remains unchanged.

In particular if \( a_{ik} = 0 \) then entire \( j^{th} \) row remains unchanged and hence can be transferred to new table as it is without performing any pivoting operation. Thus, after determining the pivotal element, performing following steps will reduce the computations drastically.

1. Identify the rows with zero elements in the pivot column.
2. Copy these rows as it is without performing any pivoting operations as these entries remain unchanged.
3. In the rows containing non-zero element in pivot row, note columns with zero entry these can be copied directly and pivoting be performed on rest entry.

On implementation of these modification we note the following points,
- Observe that every pivot operation requires \(-, *, /\) i.e. 3 arithmetic operations for each new entry \( \hat{a}_{ij} \). Thus if there are \( m \) rows containing zero element in pivot column and with \( c \) columns in current tableau then total of \( 3 \times m \times c \) arithmetic operations can be avoided while obtaining entries of the new table. This can be very large for large size problems.
- The above observation can be effectively used for resolving the tie-breaker choice for deciding the pivot column. During pivot selection column, generally left hand rule is used, in view 5.5 a criterion for selection of pivot column on the basis of number of zeros contained in pivot column can be used. Whenever tie occurs during selection of pivot column, in push or pull phase of the method, resolve the tie on the basis of number of zeros contained in column, choose a pivot column as column with maximum number of zeros.
• During initialization phase constraints are deleted based on the criterion of row or column with maximum zeros after reduction of cost matrix, with this deletion, unit vectors are generated and variables corresponding to these unit vectors can be readily entered in bvs without any computational efforts. Thus, more the number of basic variables included during the initialization phase less the number of iterations required to fill up the bvs during push phase. In view of this any choice of deletion of constraints that result in more number of variables in the bvs will be beneficial as it will reduce the number of iterations required to complete bvs during push phase of algorithm. Selection based on this logic has resulted in inclusion of 4 basic variables instead of 3 basic variables in bvs during initialization phase. This will be very helpful as inclusion of each individual variable in bvs during initialization phase saves entire pivotal calculations for generating new table entries.

5.6 Managing Cost Uncertainties

In AP model assignment of activities to equal number of resources is done primarily on the basis of cost associated with it. In real life we observe that the cost parameters are subjected to fluctuations in the market, they are more or less uncertain. The uncertainty in the cost parameter may arise due to,
• Ever changing currency exchange rate
• A time lapse between the development and implementation of the model.
Uncertainty in estimated cost is unavoidable and therefore managers are concerned with the stability of the optimal solution under uncertainty of the estimated cost parameters. Manager needs to consider his decision under these uncertainty. The current approaches to deal with the cost uncertainty includes Scenario Analysis, Worst-case Analysis, Monte-Carlo Approaches and Re-optimization. Neither of this approach produces any managerially useful prescriptive ranges for sensitivity analysis. Also, the prerequisite for above approaches is anticipation of possible scenarios and need for anticipated direction of changes in cost parameters. Adlakha V. and Arsham H[1] have suggested perturbation analysis approach for cost parameters $c_{ij}$ which help to:
• Determine the responsiveness of the solution to changes or errors in cost parameters.
• Adapt a model to new environment with an adjustment in these parameters.
• Provide systematic guidelines for allocating scarce organizational resources by using the sensitive information.
• Determine a cost perturbed region in which the current strategic decision is still valid.
Although all aspects of perturbation analysis are readily available for LP models, even ordinary sensitivity analysis is rarely performed on AP for the reason that the final solutions generated by solution methods do not contain enough information to perform perturbation analysis. The Push-and-Pull algorithmic method provides this information and the perturbation analysis is possible. Perturbation approach using Push-and-Pull method provides managerially useful prescriptive ranges on cost parameters for the sensitivity analysis, parametric programming and simultaneous changes without anticipation of possible scenarios and need for anticipated direction of changes in cost parameters.

5.6.1 Perturbation Analysis

From a managerial point of view, the PA results provide an assessment and analysis of the stability of the optimal strategy by monitoring the admissible range that preserves it. This permits evaluation of the impact of uncertainty in the data. PA gives a manager more leverage in allocating scarce resources. Monitoring the admissible ranges of PA that preserve the current optimal strategy aids in determining how resources should be applied. Information about departure from these limits allows a manager to anticipate the consequences and to pre-determine back-up strategies. The results empower a manager to assess, analyze, monitor, and manage various types of cost uncertainties in all phases of the model, namely design, solution, and implementation.

The perturbed objective function for the given AP is defined as

\[
\text{Min } z = \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} + c'_{ij})x_{ij}
\]

with the same constrains and the admissibility condition \( c'_{ij} \geq c_{ij} \). Now, the Push-and-Pull method is implemented to obtain the optimal solution and as soon as the optimal solution to original problem is obtained, the perturbation analysis can be immediately started. The necessary components for cost perturbation analysis from the final table are shown below

<table>
<thead>
<tr>
<th>bvs</th>
<th>( x_n )</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_n )</td>
<td>( [A] = B^{-1}N )</td>
<td>( b )</td>
</tr>
<tr>
<td>Reduced Cost</td>
<td>( c'_n = c_n - c_n[A] )</td>
<td>Optimum Cost</td>
</tr>
</tbody>
</table>

Table 5.6.1.1

For the purpose of perturbation analysis the initial table is partitioned into B, the basic variable coefficient, and N, the non basic variable coefficients, as they appear in the final table and the rhs column.
To find critical region, that is the largest set of perturbed costs for which the current solution remains optimal, we must find the allowable changes in the cost coefficients. The necessary and sufficient condition to maintain the current optimal strategy is that \( c_{ij} > 0 \), where 0 stands for a zero vector with the appropriate dimension.

Let \( \theta \) denote the set of perturbed cost \( c'_{ij} \) to maintain optimality.

\[
\theta = \{ c'_{ij} \mid c_{ij} > 0 \text{ and } c'_{ij} \geq -c_{ij} \}
\]

(5.6.1.1)

The set \( \theta \) is non-empty since it contains the origin \( c'_{ij} = 0 \), for all \( i, j \). This set can be used to check whether the given perturbed cost coefficients have the same optimal solution as the original problem. The procedure for obtaining the perturbed set is illustrated with the example below.

**Ex: 5.1 (Continued)** The critical region for the Ex:5.1 as per 5.6.1.1 is shown below

\[
\theta = \{ c'_{ij} \mid 2 + c'_{11} - c'_{12} + c'_{21} - c'_{22} - c'_{13} + c'_{14} - c'_{15} + c'_{31} - c'_{32} + c'_{33} - c'_{34} + c'_{35} - c'_{41} + c'_{42} - c'_{43} + c'_{44} - c'_{45} + c'_{51} - c'_{52} + c'_{53} - c'_{54} + c'_{55} \geq 0, \text{and all other } c'_{ij} \geq 0 \}
\]

(5.6.1.1)

The set \( \theta \) can be used to check whether specific changes in the values of \( c_{ij} \) will lead to the same optimal solution as original problem or not. For example let the perturbed cost vector be \( c + c' \), then values of \( c'_{ij} \) of vector \( c' \) can be substituted in set \( \theta \) and one can easily check whether the current solution is still optimal or not. The perturbation analysis discussed above is the most general one and it can handle the simultaneous and independent changes in the cost parameters.

**5.7 Parametric Sensitivity Analysis**

The study of parametric sensitivity analysis involves the analysis of simultaneous changes in a given direction and it is of great importance whenever there is dependency among the cost parameters. This analysis can be considered as simultaneous changes in a given direction. The cost vector, \( c \), is composed of elements, \( c_{ij} \) signifying the cost of assigning the \( i^{th} \) resource to the \( j^{th} \) activity. Let \( P \) be a perturbation vector consisting of elements specifying a perturbation in each cost element. Introducing a scalar parameter \( \delta \geq 0 \), we would like to find out how far we can move in the direction of \( P \), \( \delta \) being the step size, while still maintaining optimality of the current optimal assignment. The steps followed to perform parametric sensitivity analysis are.
1. Identify $P_N$ and $P_B$, the sub-vectors of $P$ corresponding to the non basic and basic variables, as they appear in the final tableau of the original problem.

2. Define set $S = \{ i \rightarrow j \mid (P_N - P_B \times [A])_{ij} < 0 \}$

3. If $S = \emptyset$, then $\delta' = \infty$. Go to step 5.

4. Calculate $\delta' = \min \{ -(old \ c^*_{N_i,j}) / (P_N - P_B \times [A])_{ij} \}$ over all $(i, j) \in S$.

5. Determine $\delta' = \min \{ \delta', \min(\delta | c_9 - \delta P_9 \geq 0) \}$.

6. New $c^*_{N_x} = old \ c^*_{N_x} + \delta( P_N - P_B \times [A])$ for any $\delta \in [0, \delta']$.

Various steps of the algorithm are illustrated by using example Ex:5.1 from (5.6.1) Let the cost vector $c$ in the example be perturbed along vector $P = [1, -1, 1, 1, -2, -1, 1, 1, 1, 2, 1, 1, -1, 1]$ and hence the perturbed cost coefficient vector is $[18 + \delta, 26 + \delta, 28 - 26, 14 + \delta, 18 + \delta, 19 + \delta, 15 + 28, 19 + \delta, 26 + \delta, 24 - \delta, 19 + \delta]$ The perturbed coefficients for non basic variables $P_N = \left[\begin{array}{cccccccccccc}
\ x_{11}, & x_{12}, & x_{23}, & x_{24}, & x_{25}, & x_{26}, & x_{27}, & x_{28}, & x_{29}, & x_{30}, & x_{31}, & x_{32}, & x_{33}, & x_{34}, & x_{35}, \end{array}\right] = \left[\begin{array}{cccccccccccc}
1, & -1, & 2, & -2, & 1, & 1, & 1, & 1, & 1, & -1, & 1, & -1, & \end{array}\right]$ and the perturbed coefficients for basic variables are $P_B = \left[\begin{array}{cccccccccccc}
\ x_{14}, & x_{24}, & x_{25}, & x_{26}, & x_{27}, & x_{28}, & x_{29}, & x_{30}, & x_{31}, & x_{32}, & x_{33}, & x_{34}, & x_{35}, \end{array}\right] = \left[\begin{array}{cccccccccccc}
1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & \end{array}\right]$.

$\therefore P_N - P_B \times [A] = [\left[\begin{array}{cccccccccccc}
1, & -1, & 2, & -2, & 1, & 1, & 1, & 1, & 1, & -1, & 1, & -1,1, & \end{array}\right] - \left[\begin{array}{cccccccccccc}
1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, \end{array}\right] \times [A]$

$= \left[\begin{array}{cccccccccccc}
1, & -1, & 2, & -2, & 1, & 1, & 1, & 1, & 1, & -1, & 1, & -1, & \end{array}\right] - \left[\begin{array}{cccccccccccc}
1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, \end{array}\right]$.

$= \left[\begin{array}{cccccccccccc}
1, & -1, & 2, & -2, & 1, & 1, & 1, & 1, & 1, & -1, & 1, & -1, & \end{array}\right] - \left[\begin{array}{cccccccccccc}
3, & -1, & 1, & -1, & -1, -1, & 3, & 3, & 1, & \end{array}\right]$

$= \left[\begin{array}{cccccccccccc}
\ x_{11}, & x_{12}, & x_{23}, & x_{24}, & x_{25}, & x_{26}, & x_{27}, & x_{28}, & x_{29}, & x_{30}, & x_{31}, & x_{32}, & x_{33}, & x_{34}, & x_{35}, \end{array}\right] = \left[\begin{array}{cccccccccccc}
-2, & -4, & 3, & -3, & 2, & 0, & 2, & -2, & -2, & -2, & -2, & -2, & -2, & -2, & -2, & -2, \end{array}\right]$

Now $(P_N - P_B \times [A]) < 0$ yields $S = \{1, 2, 3, 4 \} \therefore \delta' = \min \{ -2, -4, 3, -3, 2, 0, 2, -2, -2 \} \delta' = \min \{ 1, 3, 5, 6, 6, 6, 6, 6, 6 \}$ $\delta' = 1$

As per step 5 of algorithm $\delta' = \min(1, \min(26, 14, 24))$ $\therefore \delta = 1$ $\therefore$ New $c^*_{N_x} = old \ c^*_{N_x} + \delta( P_N - P_B \times [A])$ for any $\delta \in (0, 1)$ The elements of new $c^*_{N_x}$ for $\delta = 1$ are $\text{New } c^*_{N_x} = [2, 6, 5, 11, 18, 23, 2, 4, 7, 8] - [-2, -4, 3, -3, 2, 0, 2, -2, -2, -2]$
New $c_N^* = [0.2, 8.8, 20.23, 4.2, 5.6]$

Thus for $\delta=1$ the cost row entry corresponding to non basis vector $x_{ii}$ is zero and thus one can obtain an alternate optimal solution as variable $x_{ii}$ can be brought into basis.

### 5.8 Ordinary Sensitivity Analysis

Ordinary sensitivity analysis is very popular and readily available in LP. Ordinary sensitivity analysis type analysis for AP can be carried out easily by using parametric perturbation analysis algorithm since ordinary sensitivity analysis is a special case of parametric analysis. To, obtain upper sensitivity limit for cost $c_{ij}$, we can consider perturbed cost vector as a unit row vector and for lower sensitivity limit perturbed cost vector can be considered as its negative. The step size $\delta$ is the amount of increase or decrease in that direction.

Other way of obtaining these sensitivity limits for particular $c_{ij}$ is, by setting all other cost except $c_{ij}$ to zero in the set $\Theta$ given by (5.6.1.1). Let $c_pq$ be the cost coefficient for which upper and lower sensitivity limits are to be obtained. Now set $c_{i} = 0$ except for $i=p$ and $j=q$ in the set of equations contained in $\Theta$ and then solve the resulting equations which gives us the upper and lower limits for $c_{pq}$, holding all other cost coefficients constant. The given cost coefficient $c_{pq}$ can vary within these sensitivity limits so that the optimal solution to the problem does not change. The allowable ranges for the different $c_{ij}$'s are shown below

<table>
<thead>
<tr>
<th>Cost</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
<th>Cost</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{11}$</td>
<td>-2</td>
<td>$\infty$</td>
<td>$c_{31}$</td>
<td>-23</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{12}$</td>
<td>-6</td>
<td>$\infty$</td>
<td>$c_{32}$</td>
<td>-19</td>
<td>6</td>
</tr>
<tr>
<td>$c_{13}$</td>
<td>-5</td>
<td>2</td>
<td>$c_{33}$</td>
<td>-2</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{14}$</td>
<td>-4</td>
<td>5</td>
<td>$c_{34}$</td>
<td>-5</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{21}$</td>
<td>-13</td>
<td>2</td>
<td>$c_{41}$</td>
<td>-4</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{22}$</td>
<td>-11</td>
<td>$\infty$</td>
<td>$c_{42}$</td>
<td>-7</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{23}$</td>
<td>-2</td>
<td>2</td>
<td>$c_{43}$</td>
<td>-8</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$c_{24}$</td>
<td>-18</td>
<td>$\infty$</td>
<td>$c_{44}$</td>
<td>-10</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5.8.1
5.9 Degenerate Optimal Solutions

The set $\theta$ constructed in (5.6.1.1) is used to construct the critical region, that is, the largest set of perturbed costs for which the current solution remains optimal, is valid for a non-degenerate solution only i.e. when each rhs value is greater than zero. On the contrary, the optimal solution to the assignment problem is highly primal degenerate. In view of this the procedures for carrying out sensitivity analysis need to be modified in respect of degeneracy phenomenon. Also, when the optimal solutions are primal degenerate valid upper and lower bounds for the cost coefficients may not be evaluated using only one primal degenerate tableau. Thus, for degenerate optimal solution the procedure for obtaining the largest set $\theta$ for carrying out parametric and sensitivity analysis is modified and this is done by obtaining distinct optimal tableau for each degenerate row along with corresponding set $\theta_i$. Finally, to obtain valid cost sensitivity limits in case of degenerate solution the set $\theta$ is obtained by considering intersection all $\theta_i$.

5.9.1 Algorithm for Degeneracy

The usual procedure with modification is presented below

1. Let $Z$ denote the number of zeros in the rhs of current optimal tableau. Now set $l = Z + 1$ and $i = 0$
2. Let $i = i + 1$. Now determine set $\theta_i$ for the current optimal tableau.
3. If $i = l$, go to step 8
4. Select a degenerate row to exit, denote it by PR
5. Now, find a column with largest negative ratio, given by cost row element $i$ pivotal row element.
6. Variable corresponding to PR moves out of basis and variable corresponding to most negative ratio column enters the basis. Obtain next distinct table by performing pivoting operations.
7. Go to step 2
8. Determine $\theta = \bigcap \theta_i \quad i \in I$

The determination of set $\theta$ (step 2 of algorithm) for each distinct tableau suggested in the algorithm is time consuming and tedious. We suggest a modification in constructing the set $\theta$. This procedure helps in obtaining the range for cost coefficient based on the reduced cost for non basic variables. The procedure given below be implemented on each distinct tableau on individual cost coefficient for obtaining sensitivity limits.
1. If the cost coefficient corresponds to non basic and let reduced cost for that variable be \( \bar{c}_j \) then its sensitivity range is \((-\bar{c}_j \rightarrow \infty)\).

2. If the cost coefficient corresponds to basic variable then let element in row of basic variable in the distinct tableau be \( x_i \) and reduced cost in the \( j^{th} \) column be \( \bar{c}_j \) then consider ratio \( \frac{\bar{c}_j}{x_i} \) for all the non basic variable columns. Then the lower and upper critical limits in this case are given by,

   \[
   \text{lower critical limit} = \begin{cases} 
   \max \{ \frac{\bar{c}_j}{x_i} \text{ for } x_i < 0 \} \\
   -\infty \text{ if there does not exist } x_i < 0 
   \end{cases}
   \]

   \[
   \text{upper critical limit} = \begin{cases} 
   \min \{ \frac{\bar{c}_j}{x_i} \text{ for } x_i < 0 \} \\
   \infty \text{ if there does not exist } x_i > 0 
   \end{cases}
   \]

The set \( \theta \) can also be used for parametric or ordinary cost sensitivity analysis. In case of parametric analysis \( \delta' \) for each distinct table is obtained while in case of sensitivity analysis upper and lower cost coefficients for each distinct table are evaluated and finally the intersection of these limits give the final valid upper and lower cost coefficient limits for the given assignment problem. As the valid lower and upper cost coefficients are available the 100% rule for simultaneous increase or decrease of all cost coefficients is easily available.

**Illustration:** The procedure adopted above is illustrated with numerical Ex:5.1 of 5.6.1. The optimum tableau for the problem is

| bvs | \( x_{11} \) | \( x_{12} \) | \( x_{14} \) | \( x_{21} \) | \( x_{22} \) | \( x_{23} \) | \( x_{31} \) | \( x_{32} \) | \( x_{33} \) | \( x_{41} \) | \( x_{42} \) | \( x_{43} \) | rhs |
|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------|
| \( x_{14} \) | 0           | 0           | 1           | 0           | 1           | 0           | 0           | 0           | -1          | -1          | -1          | 0           | ------- |
| \( x_{44} \) | 0           | 0           | 0           | 0           | 0           | 0           | 0           | 1           | 1           | 1           | 1           | 1           | ------- |
| \( x_{21} \) | 1           | 0           | 0           | 0           | 0           | 0           | 1           | 1           | 1           | 1           | 1           | 1           | ------- |
| \( x_{32} \) | 0           | 1           | 0           | 1           | 0           | 0           | 0           | 0           | 1           | 0           | 0           | 1           | ------- |
| \( x_{33} \) | -1          | -1          | 1           | 0           | 1           | 0           | 1           | -1          | -1          | 0           | 0           | 1           | ------- |
| \( x_{13} \) | 1           | 1           | -1          | 0           | -1          | 0           | 1           | 1           | 1           | 1           | 1           | 1           | ------- |
| 2   | 6           | 5           | 11          | 18          | 23          | 2           | 4           | 7           | 8           | -59         | 3           | 13          | ------- |

**Table 5.9.1.1**

We observe that final tableau is primal optimum and the valid sensitivity limits may not be obtained from the single tableau. The degenerate rows in above table are row 1 and row 5 and hence to obtain valid sensitivity analysis for cost coefficients we obtain distinct tableau corresponding to these degenerate row as per the given algorithm.
Consider first degenerate row, in this case variable $x_{14}$ moves out and first row is pivotal row. Now, consider the negative ratios given by cost row element / pivot row element corresponds to non basic variable $x_{11}, x_{42}, x_{43}$ as \{ -4/1, -7/1, -8/1 \} and as largest negative ratio -4 and this corresponds to variable $x_{41}$, hence this variable enters the basis set in place of $x_{14}$. This gives us a distinct optimal table with same optimal cost

$$
\begin{array}{cccccccccccc}
\text{bvs} & x_{11} & x_{12} & x_{23} & x_{24} & x_{31} & x_{32} & x_{33} & x_{34} & x_{42} & x_{43} & \text{rhs} \\
x_{14} & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 1 & 1 & 0 \\
x_{44} & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
x_{31} & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & -1 & -1 & 1 \\
x_{32} & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
x_{23} & -1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 \\
x_{13} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
c_{ij} & 2 & 6 & 9 & 11 & 22 & 23 & 2 & 4 & 3 & 4 & -59 \\
\end{array}
$$

Table 5.9.1.2

The above degenerate optimal tableau provides the critical region $\theta_2$ as

$$
\theta_2 = \{ 2 + c'_{11} c'_{21} + c'_{31} \geq 0, 6 + c'_{11} c'_{21} + c'_{31} \geq 0, 9 + c'_{11} c'_{21} + c'_{31} \geq 0, 11 + c'_{21} c'_{31} \geq 0, 22 + c'_{21} c'_{31} \geq 0, 4 + c'_{21} c'_{31} \geq 0, 23 + c'_{21} c'_{31} \geq 0, 24 + c'_{21} c'_{31} \geq 0, 25 + c'_{21} c'_{31} \geq 0, 26 + c'_{21} c'_{31} \geq 0, 27 + c'_{21} c'_{31} \geq 0, 28 + c'_{21} c'_{31} \geq 0, 29 + c'_{21} c'_{31} \geq 0, 30 + c'_{21} c'_{31} \geq 0, 31 + c'_{21} c'_{31} \geq 0, 32 + c'_{21} c'_{31} \geq 0, 33 + c'_{21} c'_{31} \geq 0, 34 + c'_{21} c'_{31} \geq 0, 35 + c'_{21} c'_{31} \geq 0, 36 + c'_{21} c'_{31} \geq 0, 37 + c'_{21} c'_{31} \geq 0, 38 + c'_{21} c'_{31} \geq 0, 39 + c'_{21} c'_{31} \geq 0, 40 + c'_{21} c'_{31} \geq 0, 41 + c'_{21} c'_{31} \geq 0, 42 + c'_{21} c'_{31} \geq 0, 43 + c'_{21} c'_{31} \geq 0, 44 + c'_{21} c'_{31} \geq 0 \} \). The ordinary sensitivity limits for $c_{ij}$ are now obtained by setting all other cost coefficients to value zero, shown in the next table.

Consider fifth degenerate row, in this case variable $x_{23}$ moves out. Now the negative ratios corresponds to non basic variable $x_{11}, x_{12}, x_{41}, x_{42}$ as \{ -2, -6, -4, -7 \} and as largest negative ratio -2 corresponds to variable $x_{11}$ and hence this variable enters the basis in place of $x_{23}$. This gives us a distinct optimal tableau with same optimal cost as above table

$$
\begin{array}{cccccccccccc}
\text{bvs} & x_{23} & x_{12} & x_{34} & x_{24} & x_{31} & x_{32} & x_{33} & x_{34} & x_{42} & x_{43} & \text{rhs} \\
x_{14} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 & -1 & -1 & 0 \\
x_{44} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
x_{31} & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & -1 & 0 & 1 \\
x_{32} & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
x_{11} & -1 & -1 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 & 0 \\
x_{13} & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
c_{ij} & 2 & 4 & 7 & 11 & 20 & 23 & 4 & 2 & 5 & 8 & -59 \\
\end{array}
$$

Table 5.9.1.3
The above degenerate optimal tableau provides the critical region \( \theta \), as
\[
\theta = \{2 + c_{12} + c_{13} + c_{14}, c_{15} + c_{11} \geq 0, 4 + c_{14} + c_{12} + c_{16} + c_{15} + c_{11} \geq 0, 7 + c_{16} + c_{14} + c_{12} + c_{15} + c_{11} \geq 0, 11 + c_{15} + c_{16} \geq 0, 20 + c_{16} + c_{14} + c_{12} + c_{15} + c_{11} \geq 0, 23 + c_{15} + c_{16} \geq 0, 4 + c_{14} + c_{12} + c_{16} + c_{15} + c_{11} \geq 0, 2 + c_{16} + c_{14} + c_{12} + c_{15} + c_{11} \geq 0, 5 + c_{15} + c_{16} + c_{14} + c_{12} + c_{16} + c_{15} + c_{11} \geq 0, 8 + c_{14} + c_{12} + c_{16} + c_{15} + c_{11} \geq 0 \}.
\]
The ordinary sensitivity limits for \( c_{ij} \) are now obtained by setting all other cost coefficients to value zero, shown in the table 5.9.1.4.

The valid ordinary sensitivity limits for the various cost coefficients are now obtained by considering the intersection of \( \theta_1 \), \( \theta_2 \), \( \theta_3 \). Thus \( \theta = \theta_1 \cap \theta_2 \cap \theta_3 \) along with the individual limits for the three distinct optimal tableau and valid optimum value are shown in the table below.

<table>
<thead>
<tr>
<th>Cost coefficient</th>
<th>( \theta_1 )</th>
<th>( \theta_2 )</th>
<th>( \theta_3 )</th>
<th>( \theta_1 \cap \theta_2 \cap \theta_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{11} )</td>
<td>-2 to ( \infty )</td>
<td>-2 to ( \infty )</td>
<td>-2 to 2</td>
<td>-2 to 2</td>
</tr>
<tr>
<td>( c_{12} )</td>
<td>-6 to ( \infty )</td>
<td>-6 to ( \infty )</td>
<td>-4 to ( \infty )</td>
<td>-4 to ( \infty )</td>
</tr>
<tr>
<td>( c_{13} )</td>
<td>-5 to 2</td>
<td>-( \infty ) to 2</td>
<td>-( \infty ) to 2</td>
<td>-5 to 2</td>
</tr>
<tr>
<td>( c_{14} )</td>
<td>-4 to 5</td>
<td>-4 to ( \infty )</td>
<td>-2 to 7</td>
<td>-2 to 5</td>
</tr>
<tr>
<td>( c_{21} )</td>
<td>-( \infty ) to 2</td>
<td>-3 to 2</td>
<td>-4 to 2</td>
<td>-3 to 2</td>
</tr>
<tr>
<td>( c_{22} )</td>
<td>-11 to ( \infty )</td>
<td>-11 to ( \infty )</td>
<td>-11 to ( \infty )</td>
<td>-11 to ( \infty )</td>
</tr>
<tr>
<td>( c_{23} )</td>
<td>-2 to ( \infty )</td>
<td>-2 to 2</td>
<td>-2 to ( \infty )</td>
<td>-2 to ( \infty )</td>
</tr>
<tr>
<td>( c_{24} )</td>
<td>-18 to ( \infty )</td>
<td>-22 to ( \infty )</td>
<td>-20 to ( \infty )</td>
<td>-18 to ( \infty )</td>
</tr>
<tr>
<td>( c_{31} )</td>
<td>-23 to ( \infty )</td>
<td>-23 to ( \infty )</td>
<td>-23 to ( \infty )</td>
<td>-23 to ( \infty )</td>
</tr>
<tr>
<td>( c_{32} )</td>
<td>-( \infty ) to 6</td>
<td>-( \infty ) to 3</td>
<td>-( \infty ) to 4</td>
<td>-( \infty ) to 3</td>
</tr>
<tr>
<td>( c_{33} )</td>
<td>-2 to ( \infty )</td>
<td>-2 to ( \infty )</td>
<td>-4 to ( \infty )</td>
<td>-2 to ( \infty )</td>
</tr>
<tr>
<td>( c_{34} )</td>
<td>-5 to ( \infty )</td>
<td>-9 to ( \infty )</td>
<td>-7 to ( \infty )</td>
<td>-5 to ( \infty )</td>
</tr>
<tr>
<td>( c_{41} )</td>
<td>-4 to ( \infty )</td>
<td>-4 to 3</td>
<td>-2 to ( \infty )</td>
<td>-2 to 3</td>
</tr>
<tr>
<td>( c_{42} )</td>
<td>-7 to ( \infty )</td>
<td>-3 to ( \infty )</td>
<td>-5 to ( \infty )</td>
<td>-3 to ( \infty )</td>
</tr>
<tr>
<td>( c_{43} )</td>
<td>-8 to ( \infty )</td>
<td>-4 to ( \infty )</td>
<td>-8 to ( \infty )</td>
<td>-4 to ( \infty )</td>
</tr>
<tr>
<td>( c_{44} )</td>
<td>-( \infty ) to 4</td>
<td>-( \infty ) to 4</td>
<td>-( \infty ) to 2</td>
<td>-( \infty ) to 2</td>
</tr>
</tbody>
</table>

Table 5.9.1.4
5.10 Discussion and Conclusions

The assignment problem for n resources to n activities with the brief discussion regarding the various solution approaches and there shortcomings in regards to sensitivity analysis of assignment problem and the Hungarian method in detail is presented. In a fast changing global market cost uncertainties prevail which necessitates the post optimal analysis of the assignment problem. A desirable tool is to construct a perturbation set of cost coefficient which ensures the stability of an optimal solution under such uncertainties. As AP is special class LPP, Simplex approach can be used to carry out this type of analysis but as AP posses highly degenerate optimal solution and may provide misleading results. In view of this we have studied the algorithm called the Push-and-Pull proposed by Adlakha V. and Arsham H[1]. The algorithm solves the AP efficiently and also provides useful information which can be used to make useful and valid managerial decisions. The important features of the solution algorithm are noted as under,

1. Push-and-Pull algorithm is a general purpose unified algorithm to solve classical AP and TP problems. The algorithm provides a single treatment for both problems in place of usual Hungarian and Stepping Stone methods used for solving these problems.

2. Perturbation analysis can be efficiently carried out as all the information needed to carry out perturbation analysis of cost coefficients is readily available in final tableau. Consistent managerial decisions can be taken by making use of scenario, parametric and sensitivity analysis, and monitoring various types of cost uncertainties encountered in real life situation.

3. The algorithm is free from pivotal degeneracy.

4. The algorithm is computationally efficient as the bvs augmentation phase provides a warm start by choosing initial basic feasible solution in the proximity to optimal solution. It also avoids use of slack, surplus, artificial variables as in case of LP method and thus further reduces the computational complexity.

5. The degeneracy phenomenon which is common in AP and it gives rise to various problems; the algorithm has proper subroutine to deal with primal degenerate solutions.
The algorithm is discussed in respect of computational complexity and modification is suggested to avoid pivotal calculations in generation of new table and the tie breaking choice based on this is provided. A modification in making elimination of redundant constraint is suggested and this has resulted in reduction of number of iterations which is illustrated.

The computation of perturbation set $\theta$, parametric sensitivity algorithm and ordinary sensitivity ranges are illustrated with an example. The algorithm for the computation of valid sensitivity information at primal degenerate solutions is discussed and modification is suggested in computation of set $\theta$. The results are illustrated with a numerical example.

We conclude that Push-and-Pull algorithm for the solution of Assignment problem is easy, computationally efficient algorithm that allows for simultaneous, independent, or dependent change of the cost coefficient from the estimated values.