

# CRITICAL PATH SCHEDULING BY THE LINEAR PROGRAMMING METHOD

VEDAANTA AGARWALLA  
140001038

K SUDHARSAN  
140001014

Department of Computer Science and Engineering  
IIT INDORE

EMAIL : [cse140001038@iiti.ac.in](mailto:cse140001038@iiti.ac.in)  
[cse140001014@iiti.ac.in](mailto:cse140001014@iiti.ac.in)

## Linear Programming :

Linear programs have a linear objective function and linear constraints. These constraints include both equalities and inequalities. The feasible set is a polytope, a convex, connected set with flat, polygonal faces. Following is the standard form of a linear program:

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b, x \geq 0 \end{aligned}$$

where  $c$  and  $x$  are  $n$ -dimensional vectors,  $b$  is a  $m$ -dimensional vector and  $A$  is a  $m \times n$  full row rank matrix.

The first order necessary conditions for  $x$  to be a solution of the above problem are that there exists vectors  $\lambda$  and  $s$  such that

$$\begin{aligned} A^T \lambda + s &= c, \\ Ax &= b, \\ x &\geq 0, \\ s &\geq 0, \\ x_i s_i &= 0 \end{aligned}$$

## Critical path scheduling :

Critical path scheduling is a method for finding the fastest way of accomplishing a project which consists of many individual jobs. The jobs are interconnected according to certain sequence requirements, so that certain jobs must be done before others can be started, on the other hand, many jobs can proceed concurrently. Thus the jobs form an interlocking network. The object of the analysis is to find the shortest way to accomplish all the jobs, thus minimizing the time required to complete the project.

## Conversion Of A Critical Path Problem To A Linear Programming Problem :

Critical path scheduling applies to networks of jobs which obey certain sequencing and duration requirements. Its objective is to find the minimum time necessary to accomplish all the jobs.

In a scheduling problem, the individual jobs join at points called “nodes”. The times at which these points are reached can be considered as variables. The project starts at node 1, which is assigned the time zero. Then we let

$t_2$  = time at which point 2 is reached

$t_3$  = time at which point 3 is reached

.

.

$t_n$  = time at which the project is complete (that is, point  $n$  is reached).

Thus, for a project involving  $n$  nodes, we have  $n - 1$  variables. The restraints on these time variables may now be written in the form of linear inequalities. These arise from the sequencing and duration restrictions.

Letting  $t_1 = 0$ , the sequence restrictions may be expressed as

$$t_2 \geq t_1$$

$$t_3 \geq t_2$$

$$t_4 \geq t_3$$

.

.

.

$$t_n \geq t_1$$

The fact that each job has a certain minimum duration may also be expressed in the form of linear inequalities. For example, if job 1,2 has a minimum duration of say  $x_1$  days, we may write

$$t_2 - t_1 \geq x_1$$

And if job 2,3 has a minimum duration of  $x_2$  days, we may write

$$t_3 - t_2 \geq x_2$$

A similar inequality may be written for each job.

Also, if we wish to place a limit of  $x_n$  days maximum duration on the total project, we write

$$t_n \leq x_n$$

### Job Costs :

For simplification we suppose that the job cost is related linearly to the job duration, in such a way that decreasing duration means increasing cost . An actual job cost curve, of course, might be considerably more complicated.

Thus, the cost for any job  $i,j$  can be expressed as

$$C_{i,j} = A_{i,j} + (\text{slope})_{i,j} * (t_j - t_i)$$

where the constant  $A_{i,j}$  represents the intercept on the vertical axis.

The overall cost of the project is the sum of the individual job costs, each job having its individual time-cost slope. Thus the overall project cost is a linear function of the variables  $t_1, t_2 \dots t_n$ . The problem now is to find values of  $t_1, t_2 \dots t_n$  which minimize the total project cost. The constants  $A_{1,2}, A_{2,3}$ , etc., are irrelevant and do not enter into the problem, since they cannot affect the minimization process involving the variables  $t_1, t_2 \dots t_n$ .

## Simplex :

Each iterate generated by the simplex method is a basic feasible point of our problem. A vector  $x$  is a basic feasible point if it is feasible and there exists a subset  $\beta$  of the index set  $\{1,2,\dots,n\}$  such that

- $\beta$  contains exactly  $m$  indices
- $i \in \{1,2,\dots,n\} \setminus \beta \Rightarrow x_i = 0$
- The  $m \times m$  matrix  $B$  defined by  $B = [A_i]_{i \in \beta}$  is non-singular, where  $A_i$  is the  $i^{\text{th}}$  column of  $A$ .

Theorem:

(a) If the linear problem has a non empty feasible region, then there is at least one basic feasible point.

(b) If the linear problem has solutions, then at least one such solution is a basic feasible point.

(c) All basic feasible points for linear program are vertices of the feasible polytope  $\{x \mid Ax = b, x \geq 0\}$ , and vice-versa.

At a particular point in our simplex method we check if it satisfies the KKT condition. If it does we have found our solution and if it hasn't we need to decide which index to remove from the basis  $\beta$ . Consider the following procedure to do so:

$$\tilde{N} = \{1,2,\dots,n\} \setminus \beta$$

Just as  $B$  was the basic matrix for  $\beta$ , we use  $N$  for the non-basic matrix. Also partition vectors  $x, s$  and  $c$  according to the sets  $\beta$  and  $\tilde{N}$ .

From the KKT conditions we see that:

$$Ax = Bx_b + Nx_n = b$$

As the elements in non-basic set are zero, we have

$$x_b = B^{-1}b$$

Similarly we partition  $c$  into  $c_B$  and  $c_N$  components and use  $s_n = 0$  to obtain

$$\begin{aligned} B^T \lambda &= c_B \\ N^T \lambda + s_N &= c_N \end{aligned}$$

We can thus get that

$$s_N = c_N - N^T \lambda = c_N - (B^{-1}N)^T c_B$$

The only KKT condition that we have not enforced explicitly is the non negativity condition  $s \geq 0$ . The basic component of  $s$  certainly satisfies this condition. If the non-basic component also satisfies this then we have found our optimal solution and thus we can terminate our algorithm. However, if one or more of our element in this vector are negative then we chose an index  $q$  in  $\tilde{N}$  for which it was negative and get a new corresponding basis. We also ensure that our function value decreases by doing so.

$$x_B^+ = x_B - B^{-1}A_q x_q^+$$

We ensure that one element in our new basis has decreased to zero and swap that with our index  $q$  to get a new basis.

Thus we conclude by giving one step of simplex:

Given  $\mathcal{B}, \mathcal{N}, x_{\mathcal{B}} = B^{-1}b \geq 0, x_{\mathcal{N}} = 0$ ;

Solve  $B^T \lambda = c_{\mathcal{B}}$  for  $\lambda$ ,

Compute  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$ ;

if  $s_{\mathcal{N}} \geq 0$

**stop;**

Select  $q \in \mathcal{N}$  with  $s_q < 0$  as the entering index;

Solve  $Bd = A_q$  for  $d$ ;

if  $d \leq 0$

**stop;**

Calculate  $x_q^+ = \min_{i | d_i > 0} (x_{\mathcal{B}})_i / d_i$ , and use  $p$  to denote the minimizing  $i$ ;

Update  $x_{\mathcal{B}}^+ = x_{\mathcal{B}} - dx_q^+, x_{\mathcal{N}}^+ = (0, \dots, 0, x_q^+, 0, \dots, 0)^T$ ;

Change  $\mathcal{B}$  by adding  $q$  and removing the basic variable corresponding to column  $p$  of  $B$ .

## Interior Points Method :

Unlike the simplex method where we approach the solution through a sequence of iterates that move from vertex to vertex along the edges of the feasible polytope, the iterates generated by the interior point algorithms approach the solution from the interior of the polytope. We see the Primal-Dual Path following method here.

Consider the standard-form LP problem

$$\begin{aligned} \min f(x) &= c^T x \\ \text{subject to : } Ax &= b \\ x &\geq 0 \end{aligned}$$

where the matrix  $A$  is  $p \times n$  and is also full row rank. The dual problem is

$$\begin{aligned} \max h(\lambda) &= b^T \lambda \\ \text{subject to : } A^T \lambda + \mu &= c \\ \mu &\geq 0 \end{aligned}$$

The set  $\{x^*, \lambda^*, \mu^*\}$  is called a primal - dual solution if and only if  $x^*$  solves the primal and  $\{\lambda^*, \mu^*\}$  solves the dual.

We now define a central path. A set  $\{x, \lambda, \mu\}$  is a primal-dual solution if it satisfies the conditions

$$\begin{aligned} Ax &= b & \text{with } x &\geq 0 \\ A^T \lambda + \mu &= c & \text{with } \mu &\geq 0 \\ X\mu &= 0 \end{aligned}$$

where  $X = \text{diag}\{x_1, x_2, \dots, x_n\}$ .

The central path for a standard-form LP problem is defined as a set vectors  $\{x(\tau), \lambda(\tau), \mu(\tau)\}$  that satisfy the conditions

$$\begin{aligned} Ax &= b & \text{with } x &\geq 0 \\ A^T \lambda + \mu &= c & \text{with } \mu &\geq 0 \\ X\mu &= \tau e \end{aligned}$$

where  $\tau$  is a positive scalar parameter and  $e = [1 \ 1 \ \dots \ 1]^T$ .

For each fixed  $\tau > 0$  we can find the corresponding set  $\{x(\tau), \lambda(\tau), \mu(\tau)\}$  and as  $\tau$  varies, the corresponding points form a set of trajectories called the central path. Every point in this central path is strictly feasible. Hence it lies completely in the interior of the polytope and it approaches the primal-dual solution as  $\tau$  approaches 0.

Now let  $w_k = \{x_k, \lambda_k, \mu_k\}$  where  $x_k$  is strictly feasible for the primal and  $\{\lambda_k, \mu_k\}$  is strictly feasible for the dual. We need to find an increment vector  $\delta_w = \{\delta_x, \delta_\lambda, \delta_\mu\}$  such that the next iterate  $w_{k+1} = \{x_{k+1}, \lambda_{k+1}, \mu_{k+1}\} = \{x_k + \delta_x, \lambda_k + \delta_\lambda, \mu_k + \delta_\mu\}$  remains strictly feasible and approaches the central path.

$$\begin{aligned} A\delta_x &= 0 \\ A^T\delta_\lambda + \delta_\mu &= 0 \\ \Delta X\mu_k + X\delta_\mu + \Delta X\delta_\mu &= \tau_{k+1}e - X\mu_k \end{aligned}$$

We now ignore the second order term and thus we approximate it by the following system of linear equations

$$\begin{aligned} A\delta_x &= 0 \\ A^T\delta_\lambda + \delta_\mu &= 0 \\ M\delta_x + X\delta_\mu &= \tau_{k+1}e - X\mu_k \end{aligned}$$

We now solve these to get  $\delta_w$ . Therefore,

$$\begin{aligned} \delta_x &= -y - D\delta_\mu \\ \delta_\lambda &= YAy \\ \delta_\mu &= -A^T\delta_\lambda \end{aligned}$$

where

$$\begin{aligned} D &= M^{-1}X \\ Y &= (ADA^T)^{-1} \\ y &= x_k - \tau_{k+1}M^{-1}e \end{aligned}$$

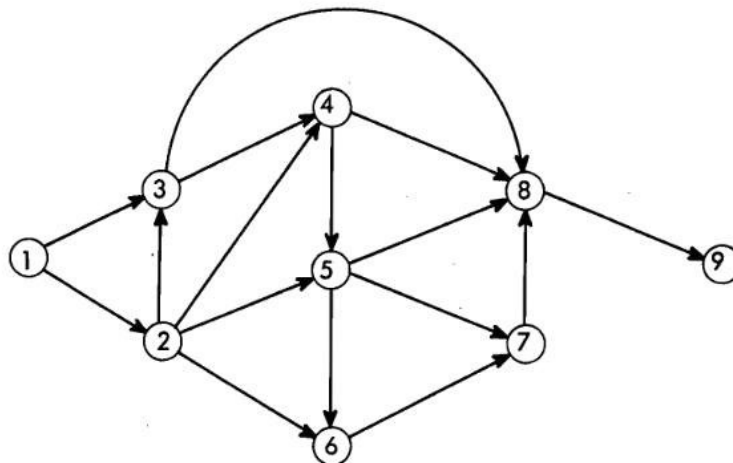
Also let

$$\tau_{k+1} = (1 - \delta/\sqrt{n})\tau_k$$

Since  $0 < \delta/\sqrt{n} < 1$ , we can say that  $\tau_k = (1 - \delta/\sqrt{n})\tau_0$  approaches 0 as  $k$  approaches  $\infty$ . So, the above method converges to the primal-dual solution.

## Sample Problem :

There are nine nodes and sixteen jobs in the network.



The duration and sequencing restrictions for this project were written as follows:

$$t_1 = 0 \text{ (by definition)}$$

$$\begin{array}{llll} t_2 - t_1 \geq 4 & t_5 - t_2 \geq 2 & t_5 - t_4 \geq 7 & t_8 - t_5 \geq 7 \\ t_3 - t_1 \geq 9 & t_6 - t_2 \geq 8 & t_8 - t_4 \geq 11 & t_7 - t_6 \geq 3 \\ t_3 - t_2 \geq 5 & t_4 - t_3 \geq 1 & t_6 - t_5 \geq 6 & t_8 - t_7 \geq 6 \\ t_4 - t_2 \geq 4 & t_8 - t_3 \geq 1 & t_7 - t_5 \geq 12 & t_9 - t_8 \geq 5 \end{array}$$

In addition, an upper limit of 50 days was placed on the over-all duration of the project :

$$t_9 \leq 50$$

The restrictions were converted to equalities by adding nonnegative slack variables.  
The new set of restrictions is

$$\begin{array}{lll} -t_2 + u_1 = -4 & -t_3 + u_2 = -9 & +t_2 - t_3 + u_3 = -5 \\ +t_2 - t_4 + u_4 = -4 & +t_2 - t_5 + u_5 = -2 & +t_2 - t_6 + u_6 = -8 \\ +t_3 - t_4 + u_7 = -1 & +t_3 - t_8 + u_8 = -4 & +t_4 - t_5 + u_9 = -7 \\ +t_4 - t_8 + u_{10} = -11 & +t_5 - t_6 + u_{11} = -6 & +t_5 - t_7 + u_{12} = -12 \\ +t_5 - t_8 + u_{13} = -7 & +t_6 - t_7 + u_{14} = -3 & +t_7 - t_8 + u_{15} = -6 \\ +t_8 - t_9 + u_{16} = -5 & & \end{array}$$

Job Costs :

Job	Minimum Duration, days	Cost Slope, \$	Job	Minimum Duration, days	Cost Slope, \$
1,2	4	-0.40	4,5	7	-0.60
1,3	9	-0.20	4,8	11	-0.10
2,3	5	-0.30	5,6	6	-0.30
2,4	4	0	5,7	12	-0.30
2,5	2	-0.40	5,8	7	0
2,6	8	-0.20	6,7	3	-0.40
3,4	1	-0.10	7,8	6	-0.20
3,8	4	0	8,9	5	-0.10

The job costs were expressed as follows:

$$\begin{array}{l} C_{1,2} = A_{1,2} - 0.40(t_2 - t_1) \\ C_{1,3} = A_{1,3} - 0.20(t_3 - t_1) \\ C_{2,3} = A_{2,3} - 0.30(t_3 - t_2) \\ . \\ . \\ . \\ C_{8,9} = A_{8,9} - 0.10(t_9 - t_8) \end{array}$$

Writing out and adding up all these gives the total cost

$$C = K + 0.50t_2 - 0.40t_3 + 0.60t_4 - 0.40t_5 - 0.10t_6 - 0.50t_7 - 0.20t_8 - 0.10t_9$$

where K represents the sum of the A's.

One additional cost factor was added: a penalty of \$10 per day on the over-all duration of the project. This was done in order to force the solution towards minimum project duration. Adding  $+10t_9$  to the cost equation above gives

$$C = K + 0.50t_2 - 0.40t_3 + 0.60t_4 - 0.40t_5 - 0.10t_6 - 0.50t_7 - 0.20t_8 + 9.90t_9$$

This total cost function C is the function that is to be minimized during the linear programming calculation. As mentioned before, the constant K has no bearing on the values of the this, and was omitted.

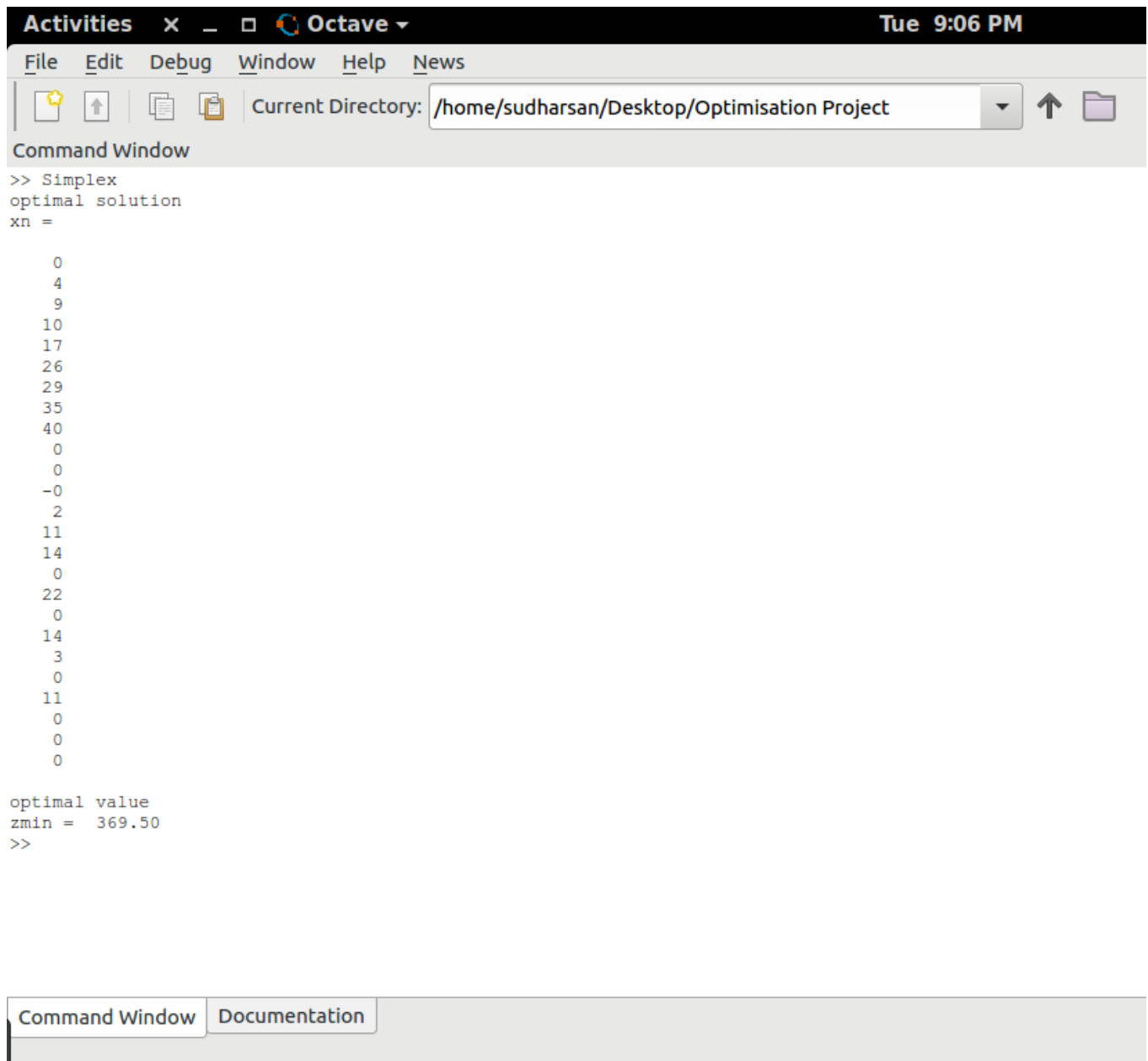
The operation of putting the problem into linear programming form has now been completed. The variables have been assigned, the restrictions have been written, and the cost function has been defined in terms of the variables.



## Results and Observations :

The following figure shows the result obtained by solving this linear programming problem using both the Simplex and Interior Points method.

Solution obtained using Simplex Method :



The screenshot shows the Octave Command Window interface. The title bar indicates the application is 'Octave' and the time is 'Tue 9:06 PM'. The menu bar includes 'File', 'Edit', 'Debug', 'Window', 'Help', and 'News'. Below the menu bar is a toolbar with icons for file operations and a 'Current Directory' field showing '/home/sudharsan/Desktop/Optimisation Project'. The main area is the 'Command Window', which displays the following text:

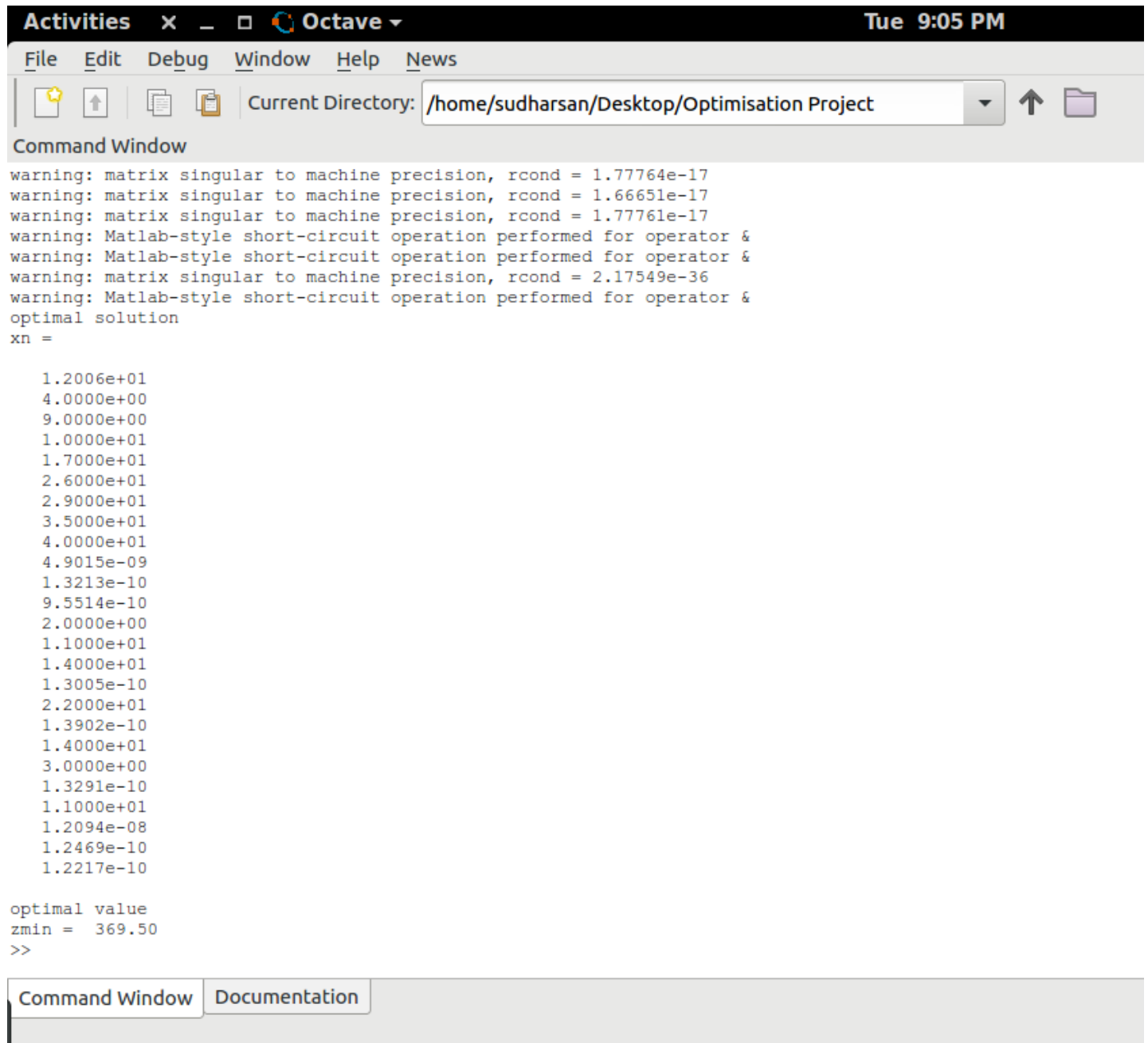
```
>> Simplex
optimal solution
xn =

    0
    4
    9
   10
   17
   26
   29
   35
   40
    0
    0
   -0
    2
   11
   14
    0
   22
    0
   14
    3
    0
   11
    0
    0
    0

optimal value
zmin = 369.50
>>
```

At the bottom of the window, there are two tabs: 'Command Window' and 'Documentation'.

Solution obtained using Interior Points Method :



The screenshot shows the Octave software interface. The title bar indicates 'Activities', 'x', a window icon, and 'Octave'. The system clock shows 'Tue 9:05 PM'. The menu bar includes 'File', 'Edit', 'Debug', 'Window', 'Help', and 'News'. The toolbar contains icons for saving, opening, and navigating. The 'Current Directory' is set to '/home/sudharsan/Desktop/Optimisation Project'. The 'Command Window' displays the following output:

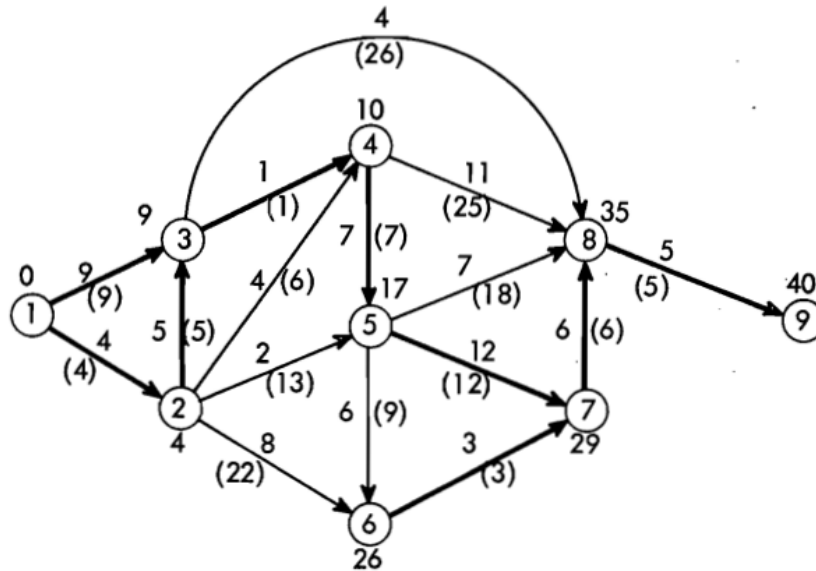
```
warning: matrix singular to machine precision, rcond = 1.77764e-17
warning: matrix singular to machine precision, rcond = 1.66651e-17
warning: matrix singular to machine precision, rcond = 1.77761e-17
warning: Matlab-style short-circuit operation performed for operator &
warning: Matlab-style short-circuit operation performed for operator &
warning: matrix singular to machine precision, rcond = 2.17549e-36
warning: Matlab-style short-circuit operation performed for operator &
optimal solution
xn =

    1.2006e+01
    4.0000e+00
    9.0000e+00
    1.0000e+01
    1.7000e+01
    2.6000e+01
    2.9000e+01
    3.5000e+01
    4.0000e+01
    4.9015e-09
    1.3213e-10
    9.5514e-10
    2.0000e+00
    1.1000e+01
    1.4000e+01
    1.3005e-10
    2.2000e+01
    1.3902e-10
    1.4000e+01
    3.0000e+00
    1.3291e-10
    1.1000e+01
    1.2094e-08
    1.2469e-10
    1.2217e-10

optimal value
zmin = 369.50
>>
```

At the bottom, there are two tabs: 'Command Window' (active) and 'Documentation'.

Using the solutions obtained, we get the critical path and the time at which each node is reached. The over-all project duration was 40 days, and the total cost was \$369.50 using both the methods. A solution by hand, using the ordinary critical path method, showed a minimum project time of 40 days. This solution agreed in every respect with the computer solution, showing that the minimum cost schedule in this case corresponds to the minimum over-all time. This is a consequence of the large penalty (\$10 per day) placed on over-all project time.



Times at nodes :

$t_1 = 0$	$t_2 = 4$
$t_3 = 9$	$t_4 = 10$
$t_5 = 17$	$t_6 = 26$
$t_7 = 29$	$t_8 = 35$
$t_9 = 40$	

## References :

[1] Royes Salmon, "Critical Path Scheduling by the Linear Programming Method", Oak Ridge National Laboratory, August 30, 1962.

\*\*\*\*\*