Multi-objective Time-Cost Optimization for Construction Activities

Group no. 1

Aniruddha Dave (120104020)

Ninad Pate (120104045)

Tarun Sahu (120104075)

Multi-objective Time-Cost Optimization for Construction Activities

Construction companies often face the problem of optimizing their resource use. They must decide on the quantity of labour, materials, equipment's to be used as well as the construction techniques that should be implemented. Time and cost of the project varies according to the selection of above mentioned resources. A project takes a longer duration to complete if the resources used are less expensive and vice-versa. Such problems are difficult to solve and they usually have multiple solutions. In this project, an example of such a construction problem is considered and optimized using genetic algorithm.

Objective Function:

$$F(x) = w_c \frac{C_{max} - C}{C_{max} - C_{min} + d} + w_t \frac{t_{max} - t}{t_{max} - t_{min} + d}$$

Where, x= number of the solution in current generation,

F(x) = fitness of the x^{th} solution in the current generation,

w_c, w_t = weights for cost and time,

c= Total cost of xth solution in the current generation,

t= Total Time of the xth solution in the current generation and

d= random constant value between 0 and 1

Weight values are assigned as follows
1.when tmax is not equal to tmin and cman is not equal to cmin
vc =cmin/(cmax-cmin),vt=tmin/(tmax-tmin)

V=vc+vt

Wc=vc/v wt =vt/v

2. when cmax = cmin and tmax=tmin
wc=wt=.5

3.when cmax is not equal to cmin and tmax is equals to tmin
wc=.1 wt=.9
4. when cmax is equal to cmin and tmax is not equals to tmin
wc=.9 wt=.1

Details for the problem:

Activity Number	Activity Description	Precedence	Option	Duration (Days)	Cost(in thousands)
1	Survey		1	5	10
			2	7	8
			3	8	6
2	Excavation	1	1	15	40

			2	18	37
			3	20	35
			4	23	31
			5	25	29
3	Forms & Rebars	1	1	10	35
			2	13	31
			3	17	27
4	Precast Beams and Columns	1	1	22	80
			2	25	75
			3	28	72
5	Foundation	2,3	1	25	65
			2	28	62
			3	32	60
			4	37	57
6	Transportation of Precast materials to site	4	1	6	20
			2	8	16
			3	9	14
7	Assembly of precast materials	5,6	1	20	45
			2	24	42
			3	27	39
			4	31	35

Chromosome: $X_1X_2X_3X_4X_5X_6X_7$

Here, X_1, X_2, X_3, X_4, X_5, X_6, X_7 are the options for activities 1,2,3,4,5,6 and 7 respectively and $X_1X_2X_3X_4X_5X_6X_7$ is the combination of options selected for construction.

For e.g., 1111111 means that option 1 is selected for all the 7 construction activities.

They take the values from 1-5. Each value is associated with the time and cost of the particular option.

Constraints:

The constraints for X_1, X_2, X_3, X_4, X_5, X_6 and X_7 are based on the options available for the specific construction activity. They are as follows:

$1<X_1<3$

$1<X_2<5$

$1<X_3<3$

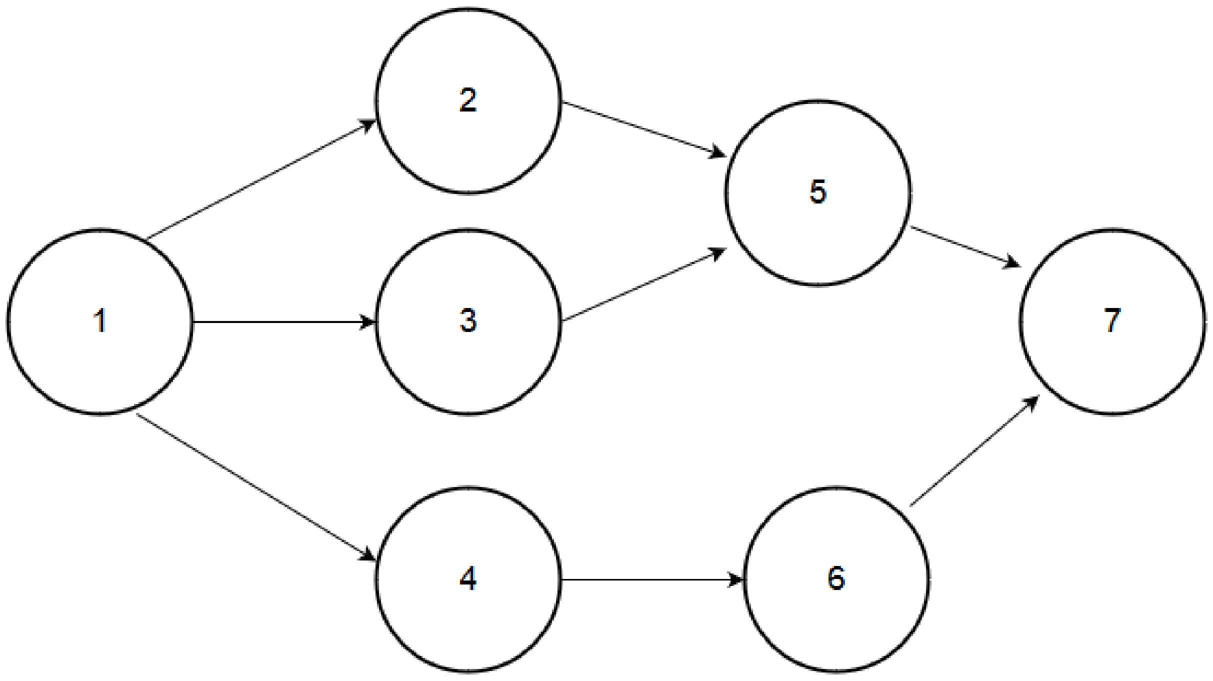
$1<X_4<3$

$1<X_5<4$

$1<X_6<3$

$1<X_7<4$

Also during the construction there is an order in which the events take place. For eg, Excavation (2) cannot be performed before the Survey (1) is over. The precedence diagram for the activities are shown below-



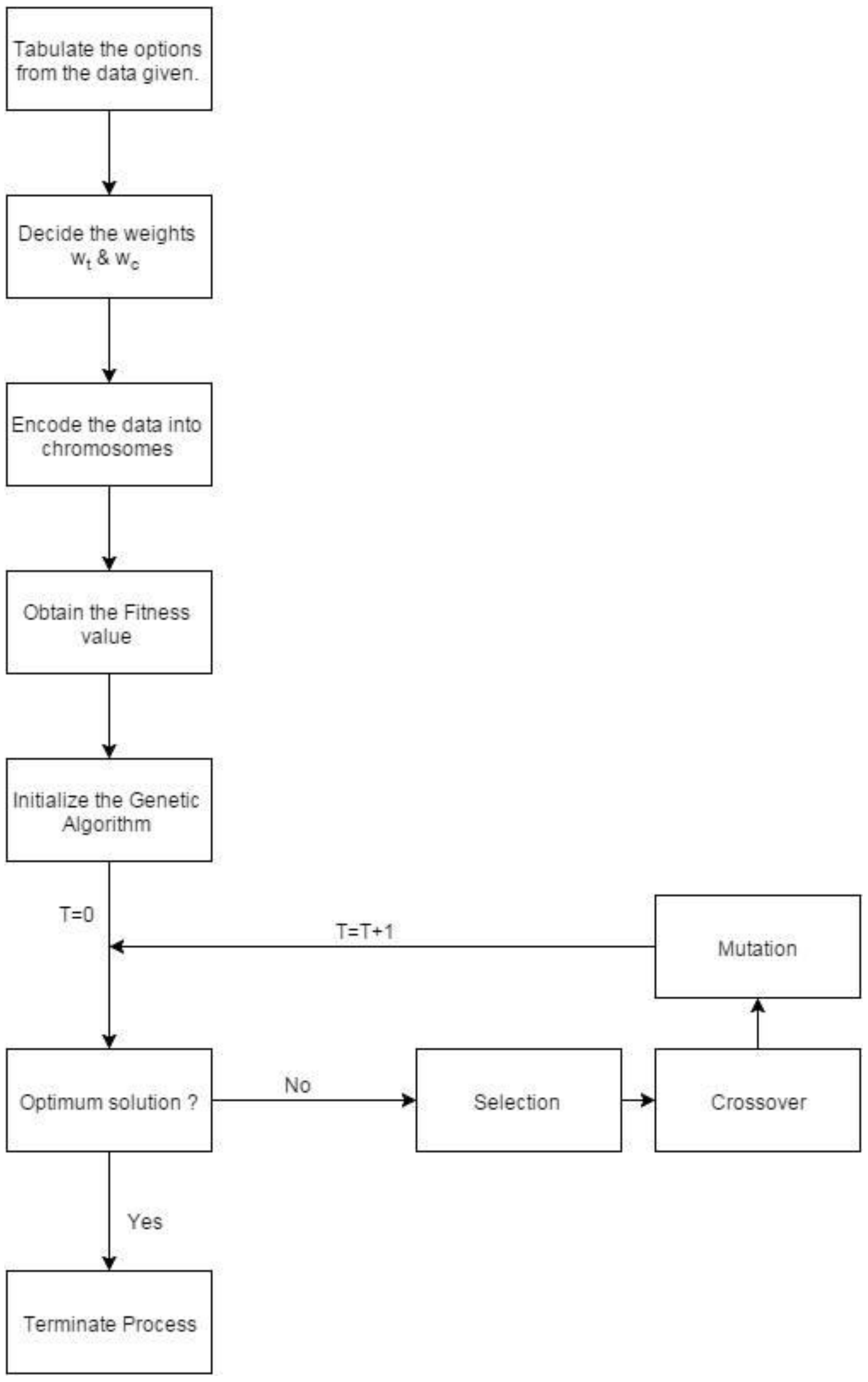
Activity Precendence Diagram

As we can see there are three paths for the construction sequence-

1. 1-2-5-7
2. 1-3-5-7
3. 1-4-6-7

The time duration required for the construction project will not be the sum of time duration of all activities, it would rather be the maximum time required for one of the above three paths (critical path). Hence, there is a constraint to follow the above paths and we need to minimize the time required for the critical path.

Flow chart for the optimization problem:


Tabulate the options from the data given.
Decide the weights w_t & w_c
Encode the data into chromosomes
Obtain the Fitness value
Initialize the Genetic Algorithm
T=0
T=T+1
Mutation
Optimum solution ?
No
Selection
Crossover
Yes
Terminate Process

Optimization technique and statistical tool to be used:

The given optimization problem is a multi-objective optimization problem. The given problem is discontinuous and has discrete values. Classical methods cannot be used and hence we will use Genetic Algorithms for the same. All the calculations will be done in the Matlab software.

Implementation of Genetic Algorithm

For a particular population there are four points namely, c_{max}, t_{max}, c_{min} and t_{min} which form the four extreme points of a quadrilateral. In the subsequent generations, these extreme points are renewed to reach an ideal minimum point. Hence, the constants in the fitness function change with every new population to reduce the size of the quadrilateral formed and reach the optimal point.

The whole data is stored in a Matrix B(5x2x5) in which the duration and time of respective activities are stored. Two functions are created in Matlab.First one (Func(x) to calculate the fitness value of population. The Genetic alogirthm inbuilt function in matlab is used in vectorized case on,which means the whole population of one generation is passed in the fitness function (func(x)) .Here x is a (populationsizeX7) matrix.which contain random value generated by ga function.GA function is used in the vectorized form to calculate the maximum value of decision variables in a set of populations, which is later used to calculate the fitness in fitness function.

To calculate the min and maximum value of decision variable a new function is defined called function calling(x) which is called in the fitness function definition.In this way for every set of populations, fitness function is called and the whole population is passed.The maximum and minimum values of the variables are calculated and fitness values are returned.In our case GA function is called for integer value solution with lower and upper limit of variables as [1,1,1,1,1,1,1] and [3,5,3,3,4,3,4] respectively.

The Matlab code is as shown at the end of the report.

As we can see the optimal point is 1133131.

Sensitivity Analysis of Decision Variables:

The problem contains discrete values for the decision variable. A change in a X_i denotes the change in the time and cost associated with the action. Below is the table showing how the fitness value changes when the decision variables are changed. The first is the optimal solution. In the subsequent rows the variables are changed and their fitness value is as shown.

Variable Changed	X1	X2	X3	X4	X5	X6	X7	Fitness Function Value
-	1	1	3	3	1	3	1	0.001
X1	2	1	3	3	1	3	1	0.9990
X2	1	2	3	3	1	3	1	1.4985
X3	1	1	2	3	1	3	1	0.0020
X4	1	1	3	2	1	3	1	0.0015

X5	1	1	3	3	2	3	1	1.4985
X6	1	1	3	3	1	2	1	0.005
X7	1	1	3	3	1	3	2	1.4980

Sensitivity Analysis of Optimization Technique:

Finding the optimal value of population size-

To find the optimal value of population size the population was varied and iterations were performed. The population size was varied from 1-120. It was observed that the optimal solution varied initially but it started converging became constant for population sizes greater than 105. Hence we can conclude that the optimal population size could be taken as 105.

Function for optimization using genetic algorithm for varying population size

```
function f = main
clear
clc;
answer_using_genetic_algorithm=zeros(10,7);
for K = (300:1:300)
   % options = gaoptimset('OutputFcns',@calling,'PopulationSize',[K]);
FitnessFunction = @(x) func(x);
options =
gaoptimset('Vectorized','on','PopulationSize',[105],'Generation',[]);
 global a1;
%[answer_using_genetic_algorithm,val]= ga(@func, 7, [],
[],[],[],[1,1,1,1,1,1,1],[3,5,3,3,4,3,4],[],[1,2,3,4,5,6,7],options);
answer_using_genetic_algorithm(K,:)= ga(@func, 7, [],
[],[],[],[1,1,1,1,1,1,1],[3,5,3,3,4,3,4],[],[1,2,3,4,5,6,7],options);
val(K,1)=func(answer_using_genetic_algorithm(K,:));
end
[answer_using_genetic_algorithm,val]
end


function a= calling(Population)
global a1;
load data;
for i=(1:1:size(Population,1));

c(i)=B(Population(i,1),1,1)+B(Population(i,2),1,2)+B(Population(i,3),1,3)+B
(Population(i,4),1,4)++B(Population(i,5),1,5)+B(Population(i,6),1,6)+B(Popu
lation(i,7),1,7);

t(1)=B(Population(i,1),2,1)+B(Population(i,2),2,2)+B(Population(i,5),2,5)+B
(Population(i,7),2,7);
t(2)=B(Population(i,1),2,1)+B(Population(i,3),2,3)+B(Population(i,5),2,5)+B
(Population(i,7),2,7);
t(3)=B(Population(i,1),2,1)+B(Population(i,4),2,4)+B(Population(i,6),2,6)+B
(Population(i,7),2,7);
d(i)=max([t(1),t(2),t(3)]);
   end
a1= [max(c),min(c),max(d),min(d)] ;
a=a1;
%a1= [max(i,c),min(c),max(i,d),min(d)] ;
end


function score = func(x)
load data;
x
popsize=size(x,1);
if(popsize~=1)
calling(x);
end

global a1;
for i =(1:1:popsize)
c
=B(x(i,1),1,1)+B(x(i,2),1,2)+B(x(i,3),1,3)+B(x(i,4),1,4)++B(x(i,5),1,5)+B(x
(i,6),1,6)+B(x(i,7),1,7);
t(1)=B(x(i,1),2,1)+B(x(i,2),2,2)+B(x(i,5),2,5)+B(x(i,7),2,7);
```

```
t(2)=B(x(i,1),2,1)+B(x(i,3),2,3)+B(x(i,5),2,5)+B(x(i,7),2,7);
t(3)=B(x(i,1),2,1)+B(x(i,4),2,4)+B(x(i,6),2,6)+B(x(i,7),2,7);
d=max(t);

if(a1(1)==a1(2) && a1(3)==a1(4))
    wc=.5;
    wt=.5;
end
if(a1(1)~=a1(2) && a1(3)~=a1(4))
    vc=a1(1)/(a1(1)-a1(2));
    vt=a1(3)/(a1(3)-a1(4));
    v=vc+vt;
    wc=vc/v;
    wt=vt/v;
end
if(a1(1)~=a1(2) && a1(3)==a1(4))
    wc=0.1;
    wt=0.9;
end
if(a1(3)~=a1(4) && a1(1)==a1(2))
    wc=0.9;
    wt=0.1;
end
score(i,1)=1*(wc*(a1(1)-c)/(a1(1)-a1(2)+1)+wt*(a1(3)-d)/(a1(3)-a1(4)+1));
end
a1
 end
```