OPTIMUM DESIGN OF GRID FLOOR USING GENETIC ALGORITHM

A PROJECT REPORT

Submitted by

B.UMADEVI

Reg. No: 71206413018

in partial fulfillment for the award of the degree

of

MASTER OF ENGINEERING

in

STRUCTURAL ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600 025

JULY 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

I certify that this project report **"OPTIMUM DESIGN OF GRID FLOOR USING GENETIC ALGORITHM"** is the bonafide work of **"MISS.B.UMADEVI"** who carried out the project work under my supervision.

26/6/2008

Dr. S. L. NARASIMHAN Ph.D.

Head Of Department

Department of Civil Engineering

Kumaraguru College Of Technology,

Coimbatore.

Dr. J . PREMALATHA Ph.D.

Professor

Department of Civil Engineering

Kumaraguru College Of Technology,

Coimbatore.

The candidate with University Register No. 71206413018 was examined by us in the project viva- voce examination held on 03.07.2008

3/7/08

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I submit our humble gratitude to feet of the divine spirit for having made this project a tremendous success. I proudly thank our affectionate, and friendly internal project guide Prof. **Dr. J. PREMALATHA Ph.D.,** Department of civil engineering for her invaluable guidance and suggestions. I thank our HOD **Dr. S. L. NARASIMHAN Ph.D.,** Department of civil Engineering for his encouragement and support.

I am blessed to be the student of Kumaraguru College of Technology. I am thankful to Principal **Dr. JOSEPH V THANIKAL Ph.D.,** from his able leadership I was able to acquire good knowledge and experience from this great institution. I also thank our faculty members and non- teaching staffs for their Cooperation and great help. Special thanks to all those who helped me to complete this project successfully.

Last but not the least, I thank my parents for their blessings and their guidance and support they gave me for making this project a successful one.

ABSTRACT

The most common form of the reinforced construction of the private and the public building is the grid floor. Even though the traditional method of design gives logical and economical results We can further improve the results if one chooses the dimensions optimally and includes the effects of the various factors like cost of steel , concrete and formwork .The objective function of this project is to reduce the total cost involved in the grid floor by considering cost of concrete, steel and formwork.

The Genetic Algorithm, a search technique based on natural evolution, is best suited for handling problems of discrete nature .Thus this paper presents an approach for the cost optimum design of grid floors using Genetic Algorithm (GA). A computer program is developed to formulate the optimization problem and few examples are solved and compared with the results obtained from the present model. It is concluded that the formulation presented in this paper leads to minimum cost design of grid floors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 General:

The design optimization of reinforced concrete structures is more challenging than the steel structures because of the complexity associated with the design because in the optimization of the steel structures only one material is considered .But in the concrete structures three different cost components due to steel concrete and formwork are to be considered and slight variations in quantity of one item may affect the total cost to a greater extent. Hence problem becomes the selection of combination of variables in appropriate quantities so that total cost is kept minimum.

Even though conventional methods gives good results mathematical programming techniques in conjunction with the high digital computers have changed the formulation of the design problem it self and they are found to give better results.

Optimization problems are solved by techniques called operation research. Optimization problem cannot be solved by a single method. Though many methods are available the problems that are discrete in nature are solved only by genetic algorithm process. In this project the optimization problem is solved by using genetic algorithm process.

1.2 Steps to solve optimization problem :

1. Choose the design variables
2. To formulate the constraint
3. Formulation of objective function
4. To choose optimization algorithm
5. To obtain the solution

Statement of optimization problem:

An optimization or mathematical programming problem can be stated as follows:

$$X = \begin{pmatrix} x1 \\ x2 \\ \vdots \\ \vdots \\ xn \end{pmatrix}$$

which minimizes function f(x) subject to the constraint

gi(x) < o, j = 1,2,3,......m

li(x) = o, j = 1,2,........p

where x is an n-dimensional vector called the design vector,

f(x) termed as objective function

gi(x), li(x) = inequality and equality constraints.

The above problem stated above is called as the constrained optimization problem. some optimization problem do not involve constraints they are.

$$X = \begin{pmatrix} x1 \\ x2 \\ \vdots \\ \vdots \\ xn \end{pmatrix}$$

Such problems are called as unconstrained optimization problem.

1.3 Objectives of present Investigation :

In this project attempt is made to optimize the grid floors using the genetic algorithm process. The total cost involved in the construction is reduced by considering the cost of concrete, steel and formwork.

CHAPTER – 2

LITERATURE REVIEW

Many researchers have investigated the cost – optimum design of reinforced concrete structures like beam column .But very few literature is available for the cost optimization of the grid floor .Mathematical programming technique called as SUMT technique was used by S.R.Adidam[1] ,N.G.R.Iyengar[1] , and G.V.Narayanan[1] for the cost optimum design of the grid floor .

In their paper a computer program was developed to find the cost optimum design of the grid floor and the comparison of cost is made for floor of different spans. Cost is optimized by the variation of the design variables .Iyengar[2] gives the optimization of the different structures like shells, plates and the grid floors.

Inspired by the Darwin's theory of the survival of the fittest, the genetic Algorithm(GA)is a global search procedure for improving the solution in the succeeding populations using the genetic operators such as reproduction, crossover and mutation .(Goldberg[3] 1989 , S. Rajasekaran[4] and G.A Vijayalakhmi Pai[4] 2003). Much work has been carried for the design optimization of the steel structures by Rajeev [5] and Krishnamoorthy[5] .GA technique was used to the optimization of the steel structure because of the feasible, practical and optimal results.

Reinforced concrete rectangular rectangular column was optimized considering the costs of the concrete, steel and the formwork. The cost optimal and time efficient design was made by incorporating the codal provisions and practical considerations using the GA technique (V.Govindaraj[6] and J.V.Ramasamy[6] 2006) Reinforced beam was optimized using the GA techniqe.In this paper cost optimized design is carried out by reducing the size of the members in addition to it reinforcement templates were created to model the reinforcement and the best bar diameter combinations are obtained from it. and the total cost reduction is obtained.(V.Govindraj

Optimum design of RC Plane Frames was carried using the simple GA where the reinforcement detailing is modeled by constructing the sets of reinforcement bars for both columns and beams and the total cost reduction of the frame is done .(V.Govindraj[7] and J.V.Ramasamy in 2007). Optimization of the plane frames was carried out and the optimum design is done according to the ACI codes. Minimization of the material and the construction costs were made subjecting to serviceability and strength requirements. (Charks. V. Champ[8], Shahram Pzeshk [8]and Hakan Hansson[8 2] 2003)

Shear capacity of the slender beams are optimized and the optimum results were presented without considering the stirrups in the beams.(M. Nehdi [9]and T. Greeenough[9] 2007). Slab formwork design is optimized and the results were produced by A.P. Alex[10] and R. Janes[10] 1978.

Though many projects are available for different structures using GA, grid floor optimization using GA is not available .In this project attempt is made to optimize the grid floor using the Genetic Algorithm by varying the quantities of the concrete , steel and formwork.

CHAPTER – 3

ANALYSIS OF GRID FLOOR

3.1 INTRODUCTION

Grid floor systems consist of becomes spaced at regular intervals in perpendicular directions. Monolithic with a slab are generally employed for architectural reasons fro large rooms such as auditoriums. Vestibules, theatre halls, show rooms of shops where column free space is often main requirement. The size of the beams running in perpendicular directions is generally kept the same. The different types of grid floor used are.

- Square grid
- Rectangular grid
- Diagrids.

Among these rectangular and square grids are used commonly. In this project square grid is used. the analysis method adopted for this project is plate theory.

3.2 BASIC ASSUMPTIONS

The orthogonal plate theory is based on following assumptions.

1. Plate is freely supported along all four edges.
2. Plate is subjected to udl only .

3.3 ANALYSIS OF GRID FLOOR BY PLATE THEORY

A reinforced concrete grid floor with ribs at close intervals in two mutually perpendicular directions connected by slab in between the ribs can be considered as an orthotropic plate freely supported on four sides. Timoshenko's analysis may be used to evaluate the moments and shear of the grid which depend upon deflection surface.

of the grid is expressed as,

$$a = \frac{16}{\Pi^6}\left(\frac{\sin\left(\frac{\Pi^x}{ax}\right)\sin\left(\frac{\Pi^y}{by}\right)}{\frac{Dx}{ax4} + \frac{2H}{ax2\ by2} + \frac{Dy}{by4}}\right)$$

where, q = total uniformly distributed load per unit area

ax, by = length of plate in x and y directions respectively

Dx ,Dy = flexural rigidity per unit length of plate along x and y direction

Cx, Cy = Torsional rigidity per unit length of plate along x and y direction.

a1 b1 = the spacings of the ribs in x and y directions respectively

Dx = (EI1/b1) Cx = (C1/b1)

Dy = (EI2/a1) Cy = (c2/a1)

where E1, E2, C1 and C2 are the flexural and the torsional rigidities of the effective section in x and y directions. The moments and shears are computed using following expressions.

$Mx = -Dx(d^2a/dx^2)$

$My = -Dy(d^2a/dy^2)$

$Txy = -(C1/b1)(d^2a/dxdy)$

$Tyx = -(C2/a1)(d^2a/dxdy)$

$Qx = -d/dx(\ Dx(d^2a/dx^2) + (C2/a1)(d^2a/dxdy))$

$Qy = -d/dy(\ Dy(d^2a/dy^2) + (C1/b1)(d^2a/dxdy))$

Where Mx1 My = The moments at the point along x and y direction

Txy1 Tyx = The torsional forces at the point on the grid along x and y directions.

3.4 DESIGN OF T- BEAMS

3.4.1Introduction

The there is a reinforced slab over a reinforced concrete beam, the slab and beam can designed and constructed in such a way that they act together. The concrete in the slabs which is on the compression side of the beam can be made to resist the compression forcer and the tension carried by the steel in the cension side of the beam. These combined beam and slab unit are called flanged beams.

3.4.2Basis of design for T – Beams

The basic assumptions used for design of rectangular beams can be used for design of T-Beams also. The assumption that plane section remains plane after bending and that failure takes place when the concrete strain reaches 0.035 holds good for T-beam also.

Three different cases (IS456 : Annexure G)

with respect to the position of the neutral axis with which T-Beams are designed are,

Case 1: Neutral axis is within the flange. In this case the beam can be treated as a normal rectangular beam of width bf and depth d.

Case 2 : the neutral axis is below the flange, and the thickness of flange is small enough so that the stress block 0.45 fck. Assumimg that Fe415 steel yields at a strain of 0.004, the following equation can be obtained

$$\frac{0.004 + 0.0035}{d} = \frac{0.0035 - 0.002}{Df}$$

$$\frac{Df}{d} = 0.2$$

Case 3 : In this case the neutral axis is below the flange but the strain in the bottom of the slab is less than 0.002 this occurs when Df/d >0.2, so that the stress in the flange is also non – linear.

3.5 Mode of Design for T – Beam:

The procedure for the design of a flanged beam consists in determining the value of x/d and the value of AS. The various cases that arises are shown in.

Case 1: Neutral axis within the flange

This is the most common case met with the design of the buildings. The formula

$$Mu = 0.36Fck\, x/d\, (1-0.416x/d)\, bfd2$$

from this (x/d) and (ast) are derived as,

$$x/d = 1.2 + \sqrt{(1.2)2 - \frac{6.68Mu}{fck\ bf\ d2}}$$

$$Ast = \frac{x.0.36fckbf}{0.87fy}$$

If (x/d) < (Df/d), than neutral axis, lies inside the flange (x/d) must be restricted to the limiting value in the code.

where Mu	=	Moment in the slab in Nmm
fck	=	compressive strength of concrete in N/mm2
d	=	depth of the beam in mm
x	=	depth of the neutral axis
bf	=	breadth of the flange
Ast	=	Area of steel
Df	=	Depth of the flange

Case 2 : Neutral Axis below the flange and Df/d <0.2

When (x/d >(df/d) the neutral axis lies outside the flange. when the value of Df/d <0.2 the stress in the slab can be assumed to be uniform and equal to 0.446fck. As given in Annex G of IS456, the moment of forces about tension steel is

$$Mu = \left[(0.36fck\ bw\ \ x\ (d-0.416x) + 0.446fck\ (bf-bw)Df(d-Df/2) \right]$$

Solving for (x/d) and Ast, from the above formulae we get,

$$x/d = 1.2 - \sqrt{1.44 - k}$$

$$k = 6.68Mu \left[bf \right] - 1.5 \left[bf\ -1 \right] \left[2-Df \right] \left[Df \right]$$

$$Ast = \frac{0.36fck\ bwx + 0.45fck\ (bf - bw)Df}{0.87fy}$$

Case 3: Neutral Axis below the flange and Df/d>0.2

When the flange thickness is greater than 0.2d and the neutral axis is below the flange, one cannot assume flange is uniformly stressed. Hence Df should be replaced by yf,

$$yf = (0.15x + 0.65Df)$$

Substituting for Df as in case 2 and taking bf as the breadth of the flange, the value of k1, is

$$k1 = \frac{6.68Mu}{fck\ bf\ d2}\left(\frac{bf}{bw}\right)^{-1.5}\left(\frac{bf}{bw} - 1\right)\left(\frac{2-Yf}{d}\right)\left(\frac{Yf}{d}\right)$$

$$x/d = 1.2 - \sqrt{1.44 - k1}$$

The area of steel is found to be

$$Ast = \frac{0.36fck\ bw\ (x) + 0.446fck\ (bf-bw)\ yf}{0.87fy}$$

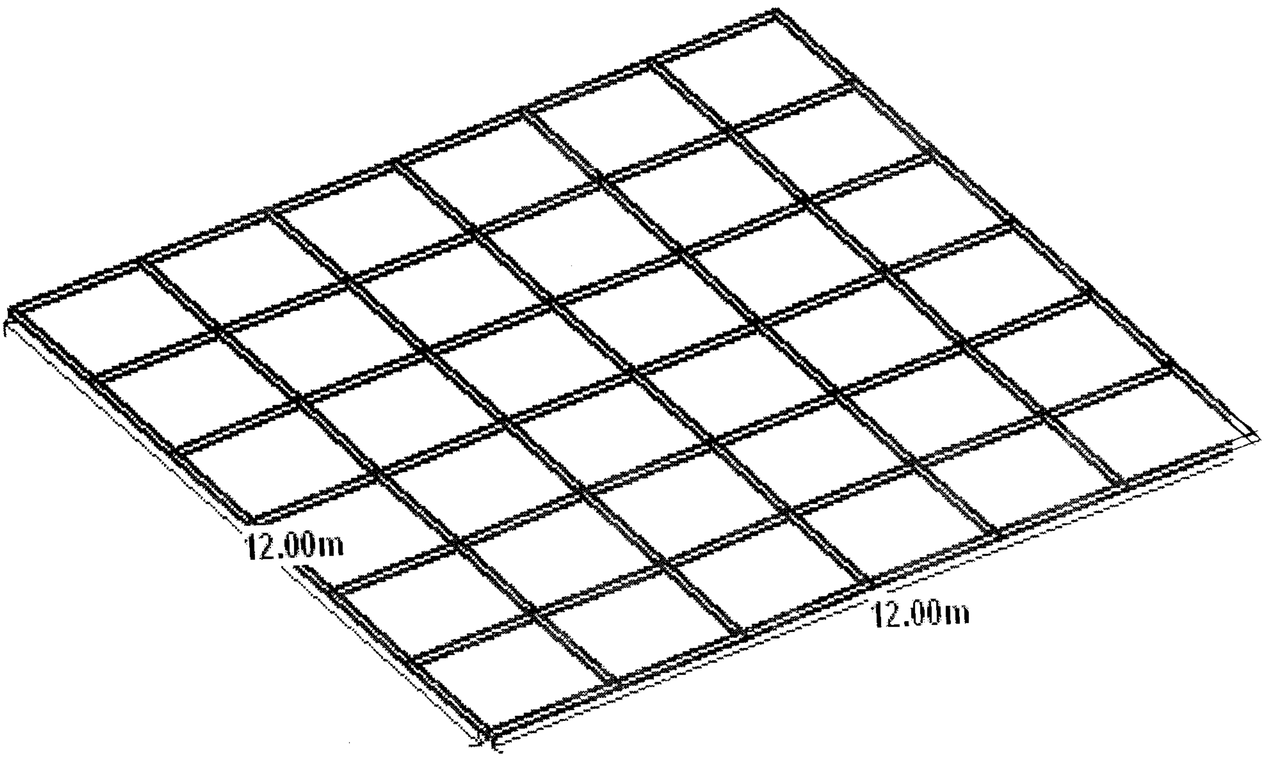
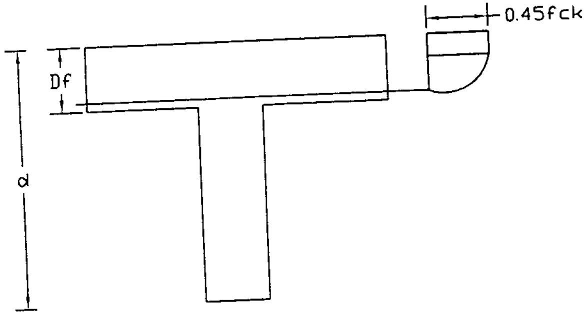
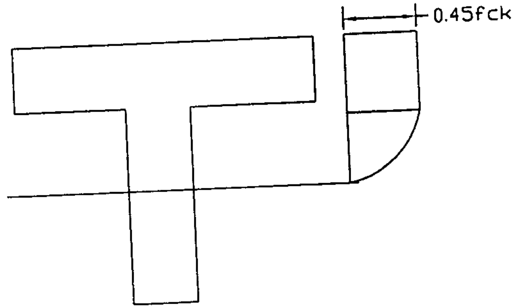


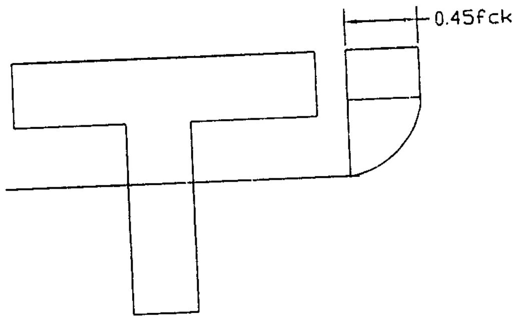
Fig.1 MODEL OF GRID FLOOR



CASE 1 : Neutral axis within flange



CASE 2: Neutral axis outside flange
and Df/d < 0.2



CASE 1 : Neutral axis outside flange
and Df/d > 0.2

CHAPTER – 4

GENETIC ALGORITHM

4.1 INTRODUCTION

What is genetic algorithm?

Problems solved by an evolutionary process resulting in a best (fittest) solution (survivor). Genetic Algorithms (GAs) are computer programs, which create an environment where populations of data can compete and only the fittest survive.

4.2 HISTORY

In 1960s I. Rechenberg first proposed in his work **"Evolution strategies"**. Later on Genetic Algorithm (GAs) were given shape by John Holland and his students and colleagues. This lead to Holland's book named **"Adaption in Natural and Artificial Systems"** published in 1975. In 1989 Goldberg introduced a modified GAs based on natural genetics. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method **"Genetic programming" (GP).**

4.3 COMPARISON OF NATURAL & GA:

Natural	Genetic Algorithm
Chromosome	string (such as "01 1 1")
Gene	feature, character, or detector (one of the bits)
Allele	feature value (such as: is it a 1 or a 0?)
Locus	string position
Genotype	structure
Phenotype	parameter type, alternative solution, a decoded solution

4.4 WORKING PRINCIPLE OF GENETIC ALGORITHM

1. Formation of the objective function.
2. Encoding consisting of bit strings
3. Fitness function for transferring minimized problem into maximized problem.
4. Applying genetic operators like reproduction, crossover ,mutation etc.
5. Applying convergence criteria of genetic algorithm

4.5 GENETIC ALGORITHM IN ENGINEERING

1. Randomness
2. Population
3. Genetic Operators

Randomness:

First, it relies in part on random sampling , which makes it a non-deterministic method, which may yield somewhat different solutions on different runs even if we haven't changed the model.

Population:

It is the no of points chosen for the solution of the problem.Only one of the best ,but the other members of the population are"sample points" in other regions of the search space,where a better solution may later be found.The use of a population of solutions help the evolutionary algorithm avoid becoming "trapped" at a local optimum, when an even better optimum may be found outside the vicinity of the current solution. Population points are chosen randomly.

ENCODING

Binary encoding

...duced this type of encoding with a limiting capacity ,such that things

Chromosome A	101101100011
Chromosome B	010011001100

In order to use GA to solve the maximization or minimization problem, unknown variables X, are first coded in some string structures. Binary coded string having 1s and 0s are mostly used. The length of the string is usually determined according to the desired solution accuracy. To convert any integer to a binary string, go on dividing the integer by 2 . We get equivalent integer for the binary code decoding.

4.6 FITNESS FUNCTION

Genetic Algorithm's mimic the Darwinian theory of survival of the fittest and the principle of nature to make a search process. Therefore, GAs are usually suitable for solving maximization problems. Minimization problems are usually transformed into maximization problems by some suitable transformation.

Fitness function F(x) is derived from the objective function and used in successive genetic operations.

$F(x) = f(x)$ for maximization problem.

$F(x) = 1/f(x)$ for minimization problem, if $f(x)\ 0$

$F(x)=1/(1+f(x))$, if $f(x)=0$;

The fitness function value of the string is known as string fitness.

4.7 GENETIC OPERATOR

Reproduction:

Reproduction is a process in which individual strings (chromosomes) are copied according to their fitness. Copying strings can be done according to their fitness or goodness, strings with a higher value having a higher probability of contributing one or more offspring in the next generation. This operator is an artificial version of natural selection, a Darwinian survival of the fittest among string creatures. ... but does not create new ones. According to

Darwin's evolution theory of survival of the fittest, the best ones should survive and create new offspring. There exist many methods for selecting chromosomes for parents to cross over namely

1. Roulette-wheel selection-it is a proportionate reproductive operator where a string is selected from the matting pool with a probability proportional to the fitness.
2. Boltzmann selection- it is a simulated annealing method of functional minimization or maximization.
3. Tournament selection - the best individual from the tournament selection strategy provides selective pressure by holding a tournament competition among individuals.
4. Rank selection - it first ranks the population according to the fitness value. The worst will be give fitness 1,the next 2,... and the best N.

CROSSOVER

After the reproduction phase is over, the population is enriched with better individuals. Crossover operator is applied to matting pool with a hope that it would create a better string. The aim of cross over operator is to search the parameter space. In addition search is to be made in a way that the information stored in the present string is maximally preserved because these parent strings are instances of good strings selected during reproduction. Cross over is undergoes by three steps, first the reproduction operator selects at random a pair of two individual strings for mating, then a cross site is selected at random along the string length and the position values are swapped between two strings following the cross site. Typically for a population size of 30 to 200, cross over rates are ranged from 0.5 to 1.

TYPES OF CROSS COVER

SINGLE POINT CROSS OVER

Chromosome 1	1 0 1 0 0 1 0 1 0 1

Chromosome 2	0 1 1 1 0 1 1 0 1 0
Offspring 1	1 0 1 **1 0 1 1 0 1 0**
Offspring 2	01 1 **0 0 1 0 1 0 1**

TWO POINT CROSS OVER

Chromosome 1	1 0 1 0 0 1 0 1 0 1
Chromosome 2	0 1 1 1 0 1 1 0 1 0
Offspring 1	1 0 1 **1 0 1** 0 1 0 1
Offspring 2	01 1 **0 0 1** 1 0 1 0

MUTATION

It involves flipping each bit by changing 0 to 1 and vice versa with a small mutation probability. A number between 0 and 1 are chosen at random. If the random is smaller than probability then the outcome of flipping is true, otherwise the outcome is false. If at any bit, the outcome is true the bit is altered, otherwise the bit is kept unchanged. Mutation acts as secondary operator with the role of restoring lost genetic materials. It is also used to maintain diversity in the population. The simple genetic algorithm uses the population size of 30 to 200 with the mutation rates varying from 0.001 to 0.5.

Original offspring 1	1101111000011110

Original offspring 2	1101100100110110
Mutation offspring 1	1100111000011110
Mutation offspring 2	1101101100110110

CONVERGENCE OF GENETIC ALGORITHM

Genetic Algorithm as preceded with more generations, there may not be much improvement in the population fitness and the best individual may not change for subsequent populations. As generation progresses, the population gets filled with more fit individuals with only slight deviation from the fitness of the best individuals so far found, and the average fitness comes very close to the fitness of the best individuals. Thus some fixed number of generations after getting the optimum point to confirm that is no change in the optimum in the subsequent generations.

4.8 BENEFITS OF GA:

The concept of genetic algorithm is

1. Easy to understand.
2. Modular, separate from application.
3. Supports multi-objective optimization.
4. Good for noisy environment.
5. We always get an answer and the answer gets better with time,
6. Inherently parallel and easily distributed.
7. Many methods are available to speed up and improve a GA's basic applications, as knowledge about the problem domain is general.
8. Easy to exploit for previous or alternate solutions.
9. Flexible in forming building blocks for hybrid applications.
10. It has substantial history and range of use.

4.9DIFFERENCES AND SIMILARITIES BETWEEN GA AND TRADITIONAL METHODS

DIFFERENCES

1. GA's are radically different from most of the traditional optimization methods. GA works with a string coding of variables that discretizes the search space even though the function may be continuous.
2. GA requires only function values at discrete points, a discrete or discontinuous function can be handled with no extra care.
3. GA operators exploit the similarities in string structure to make an effective search.
4. GA works with a population of points instead of a single point.
5. GA previously found good information is emphasized using reproduction operator and propagated adaptively through cross over and mutation operators.
5. GA is a population based search algorithm and multiple optimal solutions can be possible.

SIMILARITIES

1. In traditional search methods, where a search direction is used to find a new point, at least two points are either implicitly or explicitly used to define the search direction.
2. In the cross over operator, two points are used to create new points. Thus, cross over operator is similar to a directional search method with an exception that the search direction is not fixed for all points in the population and that ho effort is made to find the optimal point in any particular direction.

3. Since two points used in cross over operator are chosen at random, many search directions are possible. Among them, some may lead to global basin and some may not.
4. The reproduction operator has an indirect effect of filtering the good search direction and helps to guide the search. The search in the mutation operator is similar to a local search method such as exploratory search used in Hooke-Jeeves method.

CHAPTER – 5

OPTIMIZATION USING GENETIC ALGORITHM

5.1 FORMULATION OF OPTIMIZATION PROBLEM

Formulation of optimum design problem consists of identification of design variables, statement of objective function and constraints to be satisfied. In general, the structural optimization problem may be stated mathematically as,

Minimize F (x)

Subject to $gi(x) < 0; i = 1, 2, \ldots, p$

And $h_j(x) = 0; j = 1,2, \ldots, m$

When $x^l < x < x^u$

where

F(x) = objective function,

gi(x) = set of inequality constraints,

(x) = set of equality constraints,

$x = \{X_k\}$, k = 1,2,. . . ,n is the vector of design variables,

$x^l = \{X^{kl}\}$, k = 1,2,. . . ,n is the lower bounds of design variables, .

$x^u = \{X^{ku}\}$, k = 1,2...........n is the upper bounds of design variables.

5.2 OBJECTIVE FUNCTION

The objective function is the total cost consisting of individual cost components due to concrete, steel and formwork. The cost of any component is inclusive of material, fabrication, and labour. The objective function is expressed mathematically as

$$F = VcCc + W\,sCs + A_f C_f$$

where Cc, Cs and Cf are the unit cost of concrete, steel and formwork respectively. Vc, Ws and Af are the volume of concrete, weight of longitudinal plus

DESIGN VARIABLES

The cross-sectional dimensions of the beam are considered as design variables namely,

1. Breadth of section along X-direction (Bx)
2. Breadth of section along Ydirection (By)
3. Depth of section along X-direction(Dx)
4. Depth of section along Y-direction(Dy)
5. Thickness of the slab (t)

5.4 CONSTRAINTS

Constraints are taken based on strength, serviceability, ductility and other side constraints. The constraints regarding bar spacing and other bar detailing requirements are considered in the optimum detailing stage itself. All constraints are represented in the normalized form.

1.The ratio of depth to width of beam section in any span should not be greater than the maximum allowed (D/B) value as desired by the designer

g1=(dx/bx)-1 , If g1 >= 0, P1 = 0, otherwise P1 = abs(g1)

g2=(dy/by)-1 , If g2 >= 0, P2 = 0, otherwise P2 = abs(g1)

2.In order to avoid difficulties in placing and compacting concrete in formwork, the percentages reinforcement is limited to 4% as per IS code.

g3 = 1 - (Pt/4) ,g3=1-(ast/(bdx0.04))

if g3 >= 0, P3= 0, otherwise P3 = abs(g2)

g4= 1 - (Pc/4),g4=1-(ast/(bdx0.04)) , if g4>= 0, P4= 0, otherwise P4= abs(g3)

where Pt, Pc are percentages of tension and compression reinforcement respectively.

3. The serviceability requirements for deflections are imposed in many codes of practice in the form of effective span to effective depth ratios.The actual ratio of effective span to effective depth for each span should be less than the

g5 = 1 - [(Lc/d)act/(Lc/d)max] ,if g5>= 0, P5 = 0, otherwise P5 = abs(g5)

4. The reinforcement in the slab is limited to 0.12 % of the cross sectional area it is imposed by using the following constraints.

g6 = (Asslabx/(1.2*t))-1 ,if g6>= 0, P6= 0, otherwise P6 = abs(g6)

g7= (Asslaby/(1.2*t))-1 , if g7>= 0, P7= 0, otherwise P7= abs(g7)

5. The thickness of the slab is limited to about 120mm by using the following constraints

g8= ((120/t)-1) , If g8>=0, P8=0 otherwise P8=abs (g8)

5.5 PENALIZED OBJECTIVE FUNCTION

The search strategy adopted in GA considers the fitness of a solution and is unaffected by any violation of problem constraints. In order to introduce feasibility into fitness of a solution, exterior penalty functions are used to account for violated constraints. As GA is best suited for unconstrained optimization problem, penalty functions are used to transform constrained optimization problem into an unconstrained optimization problem.

Hence, the modified objective function is

$$W = F(1+C)^2 \qquad (6.10)$$

where C = absolute sum of normalized violated constraints

C= Pi=abs (gi) (6.11)

Pi = abs=(gi) (6.12)

m = total number of normalized constraints, Pi = penalty Coefficient of i^{th} constraint.

5.6 GENETIC ALGORITHM

Genetic Algorithms belong to the class of evolutionary algorithms that use the Darwinian principles of natural selection, or "Survival of the fittest". In SGA three

Binary coding system where design variables are represented using 0s and 1s were used because of its simplicity in carrying out genetic operations. Each design variable represents a potential solution known as a string (or) chromosome. Each string is made up of a series of sub string representing each discrete design variables.

REPRODUCTION

The reproduction operator were used to fill up the mating pool consisting of relatively better chromosomes that will further undergo crossover and mutation process. The reproduction operator chooses the best individuals (or) chromosomes with high fitness values. Thus by reproduction operator the information stored in strings with high fitness values were stored. Tournament selection scheme were adopted, the modified objective function were re-scaled as

$$F = \begin{cases} W_{avg} - W & \text{where } W < W_{avg} \\ 0 & \text{where } W >= W_{avg} \end{cases}$$

where W and W_{avg} is the modified fitness of each individual and average fitness of all strings in the current generation respectively. In tournament selection, two members of the population were chosen randomly at a time and the member having the higher fitness (F_o) were inserted into the mating pool. This process is repeated until the mating pool for generating new offspring is filled. Hence the mating pool comprises of the tournament winners having a higher average fitness than the average population fitness.

CROSSOVER

Amongst many crossover schemes available, two point crossover were implemented. Two parent chromosomes were selected randomly from the mating pool. One point cross-sites (bit position) randomly selected along the length of the parent strings. The binary strings contained between the cross-sites of the parents were exchanged and thus resulted in two new offspring

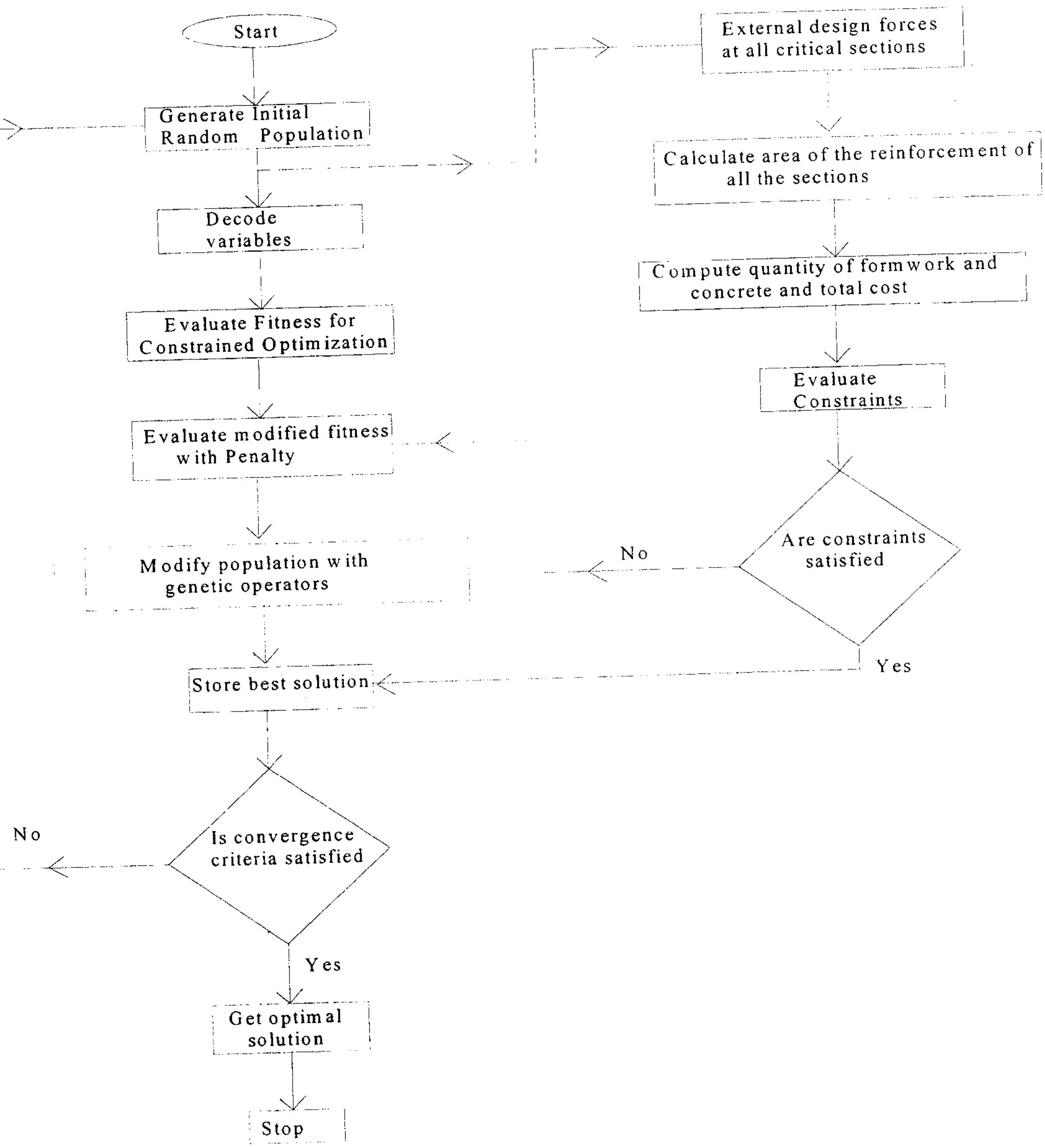
MUTATION

This operation is carried out with a view to search unexplored areas and to avoid premature convergence at local optimum solution. At the same time, the higher frequency of applying this operator may also destroy the important information contained in the offspring. Hence, the probability of mutation is kept low (usually 0.001-0.005). This operation is carried out by randomly selecting a binary bit from the entire population and flipping the values from 0 to 1 or vice-versa.

5.7 CONVERGENCE CRITERIA

The convergence is assumed to be attained by satisfying any one of the following conditions:

1. Average fitness of the last six generations remains unchanged.
2. Maximum number of generations reached.



.3 FLOW CHART OF COMPUTER PROGRAM FOR GA BASED OPTIMIZATION OF GRID FLOORS

CHAPTER – 6

EXAMPLE PROBLEM

6.1 FORMULATION OF THE PROBLEM

The design variable for these problems are breadth, depth and thickness of the grid floor. These three dates are given as input along with the lower and the upper bound values of these three variables.

The objective of this design is to mimimize the cost pex unit length of the grid floor. In this present stredy the cost of concrete, steel and formwork are considered, (the cost of shear reinforcements are negated). The cost of concrete, reinforcing steel and formwork is taken as Rs. 3500, Rs. 45 and Rs. 300 respectively. The unit weight of concrete and steel is taken as 25KNm-3 and 78.5KNM-3 respectively. Table shows the lower and upper bonds, binary bits required to represent the typical candidate solution.

6.2 VARIABLE BOUNDS

Design Variables	Bx	By	Dx	Dy	T
Lower Bound(mm)	200	200	500	500	100
Upper Bound(mm)	350	350	750	750	120

Table.1

6.3GENETIC PARAMETERS

The following genetic parameters are used namely the

String length = 25

Population size = 30,

Cross over rate = 90%

Mutation rate = 0.003

Maximum number of the generation used = 50

6.4 USER DEFINED CONSTRAINTS

The following constraints are adopted by the user in this study,

Minimum width = 200mm

Minimum Thickness=100mm

Maximum Thickness = 120mm

Minimum D/b ratio = 3

6.5 GRID FLOOR PROBLEM

Grid floor of 12m,14m,16m are considered with the live load of 4 KN/m2 and floor finish of 0.6KN/m2 are used. Concrete and steel grades used are for M20 concrete and Fe415 steel..The breadth along the x, y direction, depth along the x, y direction and the the thickness are considered as the variables. By the variation of these parameters the cost is reduced. The cost is found for the floors of span 12m, 14m, 16m and the cost comparison is shown

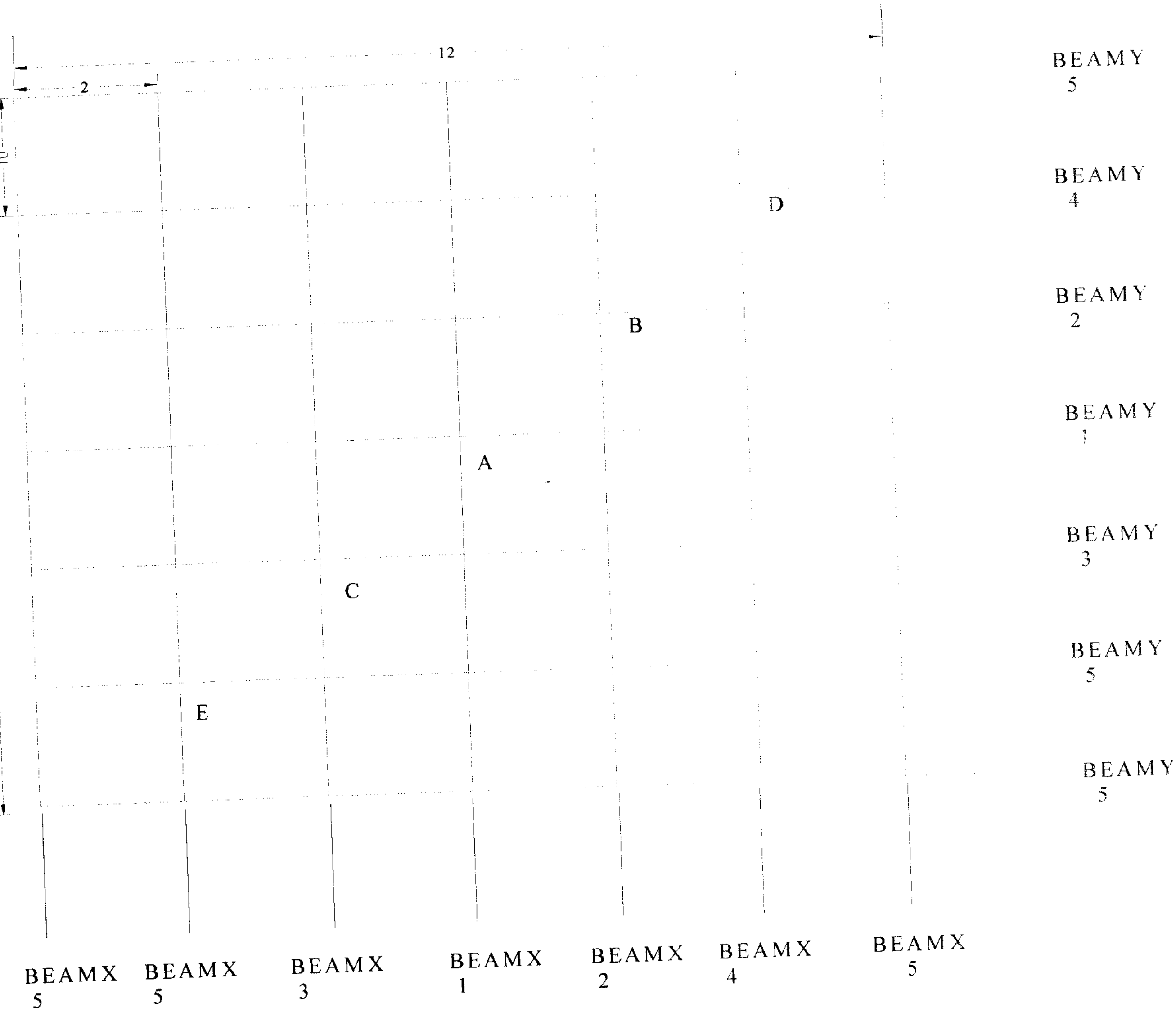


Fig-4 BEAMS FOR GRID FLOOR OF SPAN 12m

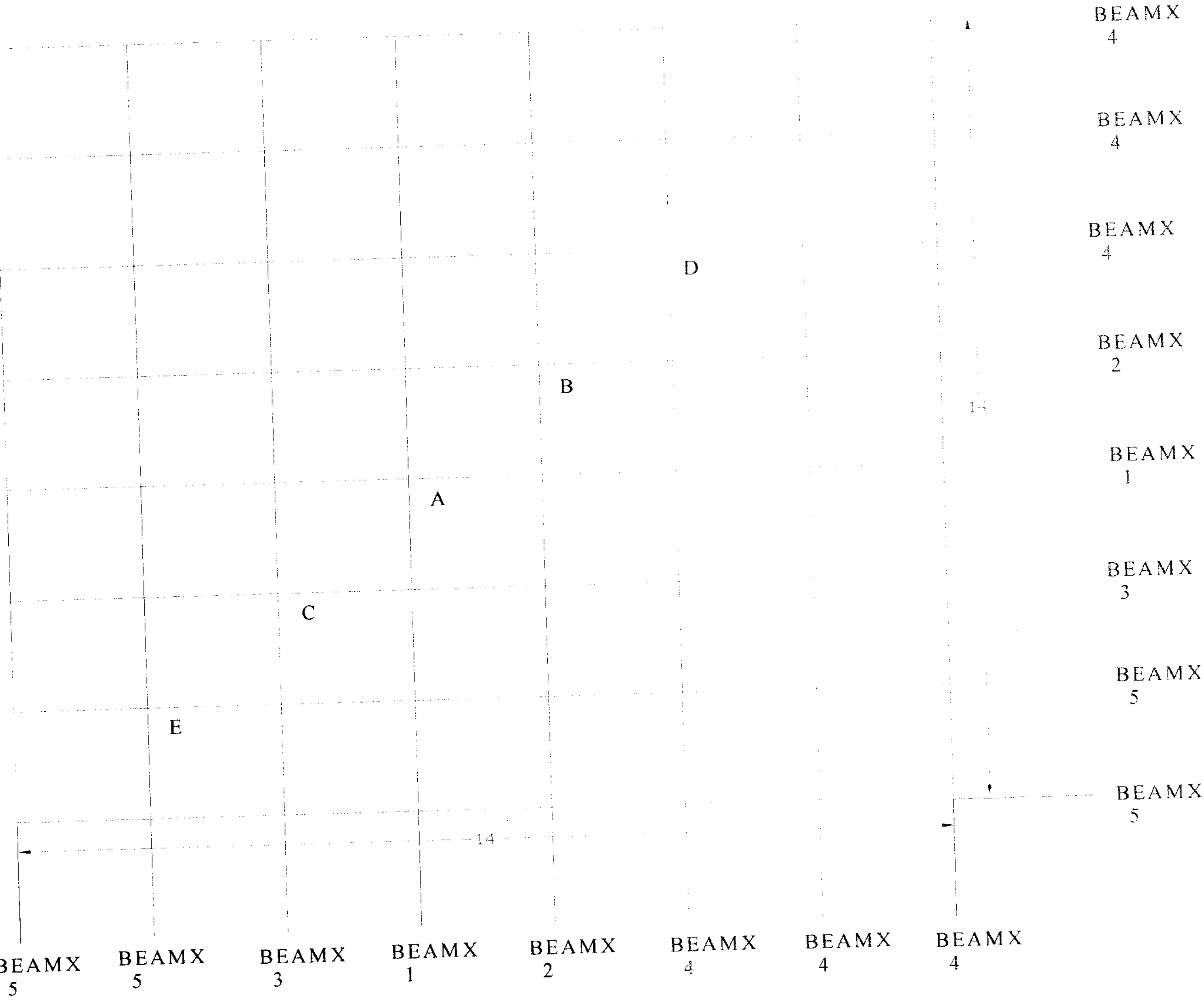


Fig-5 BEAMS FOR GRID FLOOR OF SPAN 14m

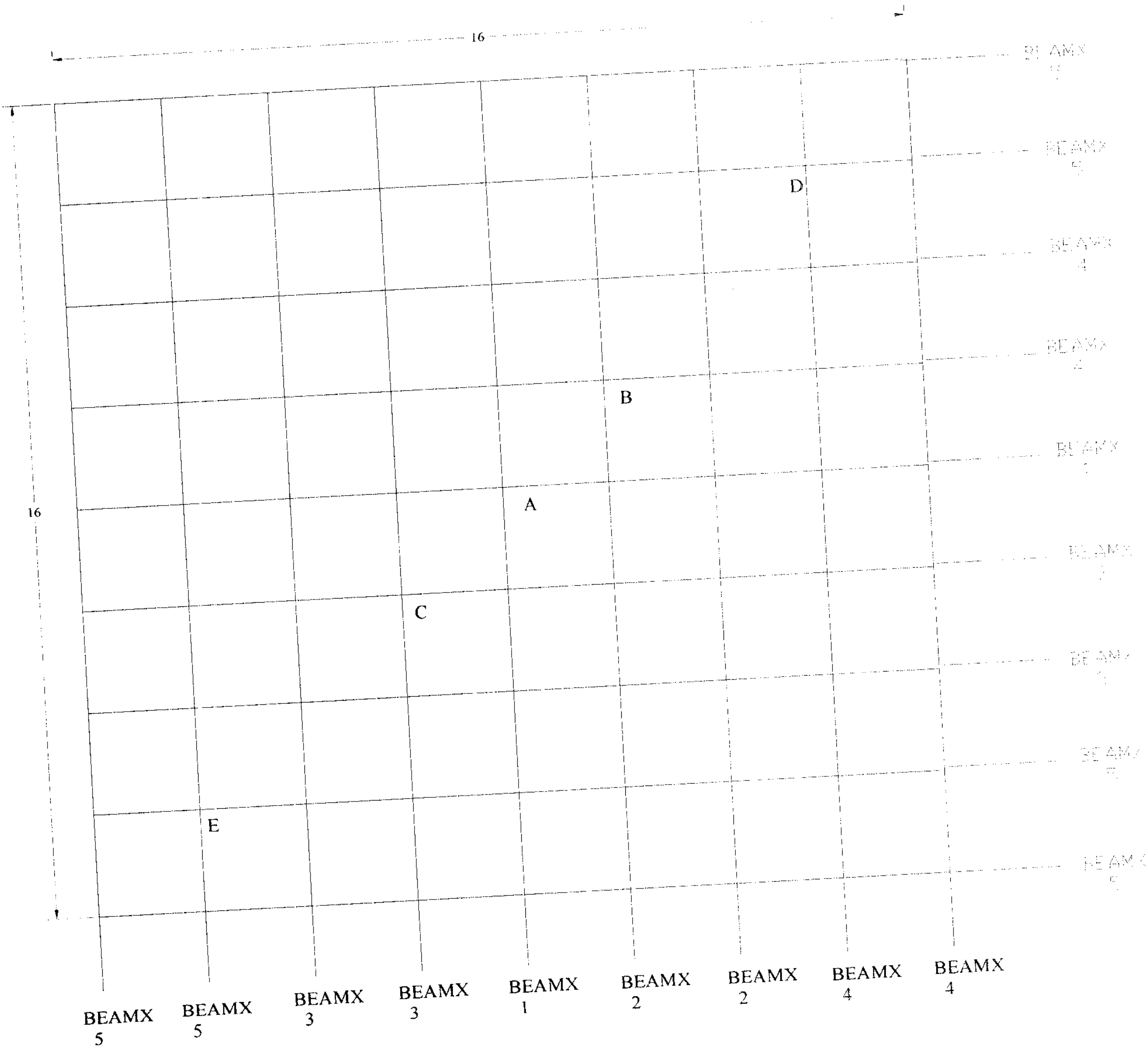


Fig-6 BEAMS FOR GRID FLOOR OF SPAN 16m

6.6 RESULTS

FOR GRID FLOOR OF SPAN 16M

========== Generation No. : 0 ======================
No. Binary|Real x Constr. violation Fitness Parents Cross-site
===

1. 306.45161 229.03226 661.29032 516.12903 118.70968| 1.15789 1.25352 0.00000 0.21951 0.00000 3.08104 4.28351 3.79981 5.20411 3.79981 5.20411 19.00000 19.00000 19.00000 19.00000 0.00000 697428.47181 (0 0) 0
String = 01101-01100-00101-01000-10111
2. 306.45161 248.38710 612.90323 629.03226 118.70968| 1.00000 1.53247 0.00000 0.15789 0.17949 3.73471 3.63008 4.56565 4.43962 4.56565 4.43962 19.00000 19.00000 19.00000 19.00000 0.00000 716485.98635 (0 0) 0
String = 01101-01010-01110-00001-10111
3. 296.77419 219.35484 709.67742 733.87097 111.61290| 1.39130 2.34559 0.00000 0.27273 0.29670 4.59668 4.43459 5.57502 5.38057 5.57502 5.38057 19.00000 19.00000 19.00000 19.00000 0.00000 748981.93086 (0 0) 0
String = 00101-00100-01011-10111-01001
4. 200.00000 340.32258 637.09677 709.67742 119.35484| 2.18548 1.08531 0.00000 0.18987 0.27273 4.78079 4.13616 5.78510 5.03803 5.78510 5.03803 19.00000 19.00000 19.00000 19.00000 0.00000 736469.99636 (0 0) 0
String = 00000-10111-10001-01011-01111
5. 282.25806 272.58065 556.45161 604.83871 105.80645| 0.97143 1.21893 0.00000 0.07246 0.14667 3.55602 3.18722 4.35486 3.92217 4.35486 3.92217 19.00000 19.00000 19.00000 19.00000 0.00000 678014.66080 (0 0) 0
String = 10001-11110-11100-10110-10010
6. 301.61290 204.83871 629.03226 516.12903 112.25806| 1.08556 1.51969 0.00000 0.17949 0.00000 2.82836 3.72027 3.50348 4.54274 3.50348 4.54274 19.00000 19.00000 19.00000 19.00000 0.00000 672251.91691 (0 0) 0
String = 10101-10000-00001-01000-11001
7. 243.54839 296.77419 717.74194 548.38710 102.58065| 1.94702 0.84783 0.00000 0.28090 0.05882 3.32365 4.66622 4.08030 5.65650 4.08030 5.65650 19.00000 19.00000 19.00000 19.00000 0.00000 697890.37745 (0 0) 0
String = 10010-00101-11011-01100-00100
8. 306.45161 301.61290 500.00000 661.29032 118.06452| 0.63158 1.19251 0.00000 -0.03226 0.21951 4.66839 3.26244 5.65961 4.01205 5.65961 4.01205 19.00000 19.00000 19.00000 19.00000 0.03226 704342.15062 (0 0) 0
String = 01101-10101-00000-00101-00111
9. 316.12903 306.45161 516.12903 629.03226 118.06452| 0.63265 1.05263 0.00000 0.00000 0.17949 4.29478 3.32865 5.22244 4.08990 5.22244 4.08990 19.00000 19.00000 19.00000 19.00000 0.00000 706186.89675 (0 0) 0
String = 00011-01101-01000-00001-00111
10. 340.32258 253.22581 540.32258 500.00000 118.06452| 0.58768 0.97452 0.00000 0.04478 -0.03226 2.87458 3.22571 3.56000 3.96611 3.56000 3.96611 19.00000 19.00000 19.00000 19.00000 0.03226 673762.36916 (0 0) 0
String = 10111-11010-10100-00000-00111

11. 335.48387 200.00000 637.09677 725.80645 111.61290| 0.89904 2.62903
0.00000 0.18987 0.28889 4.49782 3.85726 5.46147 4.70298 5.46147 4.70298
19.00000 19.00000 19.00000 19.00000 0.00000 731785.09554 (0 0) 0
String = 00111-00000-10001-00111-01001
12. 248.38710 204.83871 556.45161 540.32258 116.77419| 1.24026 1.63780
0.00000 0.07246 0.04478 2.23899 2.35282 2.80977 2.94052 2.80977 2.94052
19.00000 19.00000 19.00000 19.00000 0.00000 643882.21585 (0 0) 0
String = 01010-10000-11100-10100-01011
13. 325.80645 325.80645 508.06452 532.25806 114.83871| 0.55941 0.63366
0.00000 -0.01587 0.03030 3.45791 3.25027 4.24251 3.99925 4.24251 3.99925
19.00000 19.00000 19.00000 19.00000 0.01587 682504.52045 (0 0) 0
String = 01011-01011-10000-00100-11101
14. 330.64516 243.54839 677.41935 548.38710 110.96774| 1.04878 1.25166
0.00000 0.23810 0.05882 3.65655 4.80721 4.47553 5.81859 4.47553 5.81859
19.00000 19.00000 19.00000 19.00000 0.00000 715607.08195 (0 0) 0
String = 11011-10010-01101-01100-10001
15. 214.51613 316.12903 645.16129 629.03226 104.51613| 2.00752 0.98980
0.00000 0.20000 0.17949 3.75304 3.84288 4.58171 4.69297 4.58171 4.69297
19.00000 19.00000 19.00000 19.00000 0.00000 697109.45963 (0 0) 0
String = 11000-00011-01001-00001-11100
16. 340.32258 224.19355 685.48387 524.19355 111.61290| 1.01422 1.33813
0.00000 0.24706 0.01538 3.60411

# Generation Number	Best Fitness	Average Fitness	Worst Fitness
0	643882.215851	707312.910687	784670.044435
1	636026.534499	693089.171530	734572.228736
2	636026.534499	678310.976017	732030.889593
3	615742.275063	667729.346182	710911.995066
4	612698.870405	654998.885055	724445.611904
5	614873.881794	644064.128665	679572.110557
6	602351.764995	634199.784587	671519.779432
7	600170.667732	624756.911858	654295.313637
8	599446.414551	618479.719225	641538.047333
9	595568.278207	614091.353235	640032.769037
10	596301.952899	609299.029810	628211.850826
11	594077.575418	603840.654645	634645.377346
12	597036.722080	602384.967275	631855.694288
13	596301.952899	600214.090390	613501.584749
14	596301.952899	599670.621603	612211.162086
15	596301.952899	598930.611244	602351.764995
16	596301.952899	598748.262767	608927.641565
17	596301.952899	597751.314565	601768.479223
18	596301.952899	599083.007802	638823.011967
19	594823.316322	596870.542308	606345.040915
20	592401.458906	596547.799833	602581.839137
21	594823.316322	596750.896900	608138.981610
22	594823.316322	596154.127482	602217.646446
23	593965.309739	595583.322277	596301.952899
24	593965.309739	596948.402000	619314.373353
25	592526.647064	595184.381914	600065.873482
26	592526.647064	595068.590747	607255.982093
27	592526.647064	594459.400323	597036.722080
28	592526.647064	594181.295294	598429.057917
29	592526.647064	594073.859664	602817.040779
30	588696.545343	593626.807820	607764.700305
31	592526.647064	593696.319172	604709.867259
32	592526.647064	594096.053243	622719.190259
33	592526.647064	592551.195028	593263.086003
34	592526.647064	592526.647064	592526.647064
35	592526.647064	592526.647064	592526.647064
36	592526.647064	593333.182789	607764.700305
37	592526.647064	592627.937257	594415.562612
38	592526.647064	592918.362453	601625.877597
39	592526.647064	592747.942057	598429.057917
40	592526.647064	592932.754404	604709.867259

# Generation Number	Best Fitness	Average Fitness	Worst Fitness
0	644379.429824	717792.816102	778058.759539
1	644379.429824	702924.156812	749501.347101
2	645817.678808	688600.671897	757533.613160
3	645817.678808	675695.790109	717674.616105

5 628998.173105 657086.285400 699365.664811
6 617679.200798 648469.406264 703130.893969
7 615969.924370 640105.934446 673686.741196
8 604138.032846 635868.422498 663705.263928
9 604138.032846 628207.498679 651208.891296
10 604138.032846 621979.237985 648281.588743
11 599402.355554 616199.406455 638202.656992

12 602820.285915 611062.796434 622032.861121
13 601202.239023 607218.526976 614552.119631
14 599734.369161 605238.273654 612250.683004
15 599734.369161 606314.633508 633726.599748
16 598497.983956 603420.757847 631753.533915
17 598497.983956 601863.056294 609212.071422
18 598497.983956 601119.799680 607763.868663
19 593470.175994 600061.290516 602820.285915
20 596708.000498 599619.638608 601365.472732
21 598497.983956 600082.736364 612437.829705
22 598497.983956 599148.916513 604425.723393
23 591987.280604 598991.016046 618785.065601
24 595542.275666 598706.716300 604974.591886
25 595542.275666 599778.117050 625714.223252
26 597768.410383 598643.108502 607958.735350
27 595542.275666 598271.989335 601461.816570
28 594805.665280 597985.800761 598497.983956
29 597768.410383 598234.268043 601461.816570
30 597768.410383 597792.729502 598497.983956
31 597768.410383 601416.960646 649247.165607
32 597768.410383 600226.203628 627143.470503
33 590098.629867 597503.320153 603684.739728
34 597768.410383 600404.068009 636043.051102
35 597768.410383 598611.765805 623069.073042
36 597768.410383 597768.410383 597768.410383
37 597768.410383 598184.800080 610260.101292
38 597768.410383 598251.391956 610660.079411
39 590098.629867 599149.159959 628197.557385
40 597768.410383 599282.346833 623069.073042

Generation Number Best Fitness Average Fitness Worst Fitness
0 608421.267779 707064.435838 782604.084124
1 651756.690513 697715.724822 747021.696171
2 632313.950641 684694.193274 774380.997413
3 629876.557772 669921.098093 731146.583587
4 629876.557772 651263.889448 699944.792053
5 617218.481999 641963.039986 673697.811074
[illegible] 638301.219073 699490.531820

```
8 611797.217396 623418.178061 638559.193214
9 610136.752495 619870.949559 635352.045570
10 610136.752495 616646.346862 627872.919807
11 605157.817821 615112.234903 643955.362657
12 603679.069127 612295.258554 624160.223586
13 602936.987449 610225.603069 616747.752105
14 602936.987449 608268.574674 618927.753033
15 601452.838749 606795.481809 615785.735429
16 601452.838749 605113.328496 612588.220267
17 601452.838749 604155.278930 608121.658959
18 601452.838749 603882.446277 615071.451503
19 601452.838749 603077.612877 612570.309026
20 601452.838749 603105.857535 625427.343086
21 601452.838749 602473.790326 625897.431382

22 598461.451439 602203.564607 625427.343086
23 598461.451439 602084.986681 617093.936878
24 598461.451439 601302.412030 607425.915916
25 598461.451439 601961.673290 622588.638837


26 598461.451439 602359.552330 625427.343086
27 598461.451439 600356.889145 620934.030544
28 598461.451439 600008.767128 628501.665808
29 598461.451439 600124.248954 628501.665808
30 594636.597424 598533.382126 601452.838749
31 598461.451439 599263.271556 622516.054928
32 598461.451439 598499.602868 599605.994293
33 598461.451439 599101.759428 617670.691093
34 598461.451439 599510.487597 628501.665808
35 594636.597424 598528.087382 604285.383726
36 598461.451439 599481.739168 617670.691093
37 598461.451439 599177.849541 610528.112043
38 598461.451439 598823.323199 607886.733884
39 598461.451439 601084.110582 628501.665808
40 594636.597424 599316.846767 617670.691093

# Generation Number Best Fitness Average Fitness Worst Fitness
0 658326.525952 709292.720029 785109.165747
1 658326.525952 691846.997826 735926.369949
2 649667.827668 682619.930724 735926.369949
3 624822.323758 672757.578598 711614.234386
4 625541.935747 664465.475449 690706.645176
5 620958.349269 656601.927658 693231.817058
6 620958.349269 649098.305064 687456.987834
```

INITIAL REPORT

Variable Boundaries :
Population size : 30
Total no. of generations : 40
Cross over probability : 0.9000
Mutation probability (binary): 0.0030
Total String length : 25
Number of binary-coded variables: 5
Total Runs to be performed : 10
Lower and Upper bounds :
200.0000 <= x_bin[1] <= 350.0000, string length = 5
200.0000 <= x_bin[2] <= 350.0000, string length = 5
500.0000 <= x_bin[3] <= 750.0000, string length = 5
500.0000 <= x_bin[4] <= 750.0000, string length = 5
100.0000 <= x_bin[5] <= 120.0000, string length = 5

Run No. 1

Max = 604709.86726 Min = 592526.64706 Avg = 592932.75440
Mutations (real)= 0 ; Mutations (binary) = 94 ; Crossovers = 536
Best ever fitness: 592526.647064 (from generation : 25)
Variable vector: Binary | Real -> 200.000000 200.000000 516.129032 516.129032 100.000000|
Best_ever String = 00000-00000-01000-01000-00000
Constraint value: 1.580645 1.580645 0.000000 0.000000 0.000000 1.748480 1.748480 2.232167 2.232167 2.232167 2.232167 18.172058 18.172058 18.172058 18.172058|
Overall penalty: 0.000000

Run No. 2

Max = 623069.07304 Min = 597768.41038 Avg = 599282.34683
Mutations (real)= 0 ; Mutations (binary) = 98 ; Crossovers = 534
Best ever fitness: 597768.410383 (from generation : 23)
Variable vector: Binary | Real -> 200.000000 200.000000 532.258065 524.193548 100.000000|
Best_ever String = 00000-00000-00100-11000-00000
Constraint value: 1.661290 1.620968 2.168519502405807480000000000000000000e+67 0.030303 0.015385 1.828344 1.872893 2.325736 2.377929 2.325736 2.377929 18.717653 19.000000 18.717653 19.000000| Overall penalty: 0.000000

Run No. 3

===

Max = 617670.69109 Min = 594636.59742 Avg = 599316.84677
Mutations (real)= 0 ; Mutations (binary) = 82 ; Crossovers = 535
Best ever fitness: 598461.451439 (from generation : 22)
Variable vector: Binary | Real -> 200.000000 200.000000 516.129032 516.129032 105.161290|

Best_ever String = 00000-00000-01000-01000-00010
Constraint value: 1.580645 1.580645 265726216710699473000000000000000000000.0 0.000000 0.000000 1.718965 1.718965 2.197587 2.197587 2.197587 2.197587 17.970430 17.970430 17.970430 17.970430| Overall penalty: 0.000000

===

Run No. 4

===

Max = 592526.64706 Min = 592526.64706 Avg = 592526.64706
Mutations (real)= 0 ; Mutations (binary) = 87 ; Crossovers = 534
Best ever fitness: 592526.647064 (from generation : 25)
Variable vector: Binary | Real -> 200.000000 200.000000 516.129032 516.129032 100.000000|
Best_ever String = 00000-00000-01000-01000-00000
Constraint value: 1.580645 1.580645 0.000000 0.000000 0.000000 1.748480 1.748480 2.232167 2.232167 2.232167 2.232167 18.172058 18.172058 18.172058 18.172058|
Overall penalty: 0.000000

===

===

FOR 14m SPAN GRID FLOOR:

INITIAL REPORT

Variable Boundaries :
Population size : 30
Total no. of generations : 50
Cross over probability : 0.9000
Mutation probability (binary): 0.0030
Total String length : 25
Number of binary-coded variables: 5
Total Runs to be performed : 9
Lower and Upper bounds :
200.0000 <= x_bin[1] <= 350.0000, string length = 5
200.0000 <= x_bin[2] <= 350.0000, string length = 5
500.0000 <= x_bin[3] <= 750.0000, string length = 5
500.0000 <= x_bin[4] <= 750.0000, string length = 5
100.0000 <= x_bin[5] <= 120.0000, string length = 5

Run No. 1

Max = 523370.71682 Min = 500215.71630 Avg = 522598.88346
Mutations (real)= 0 ; Mutations (binary) = 108 ; Crossovers = 668
Best ever fitness: 523370.716815 (from generation : 24)
Variable vector: Binary | Real -> 200.000000 200.000000 709.677419 709.677419 100.000000|
Best_ever String = 00000-00000-01011-01011-00000
Constraint value: 2.548387 2.548387 0.000000 0.363636 0.363636 5.205918 5.205918 5.198653 5.198653 8.681313 8.681313 8.525821 8.525821 19.000000 19.000000 0.013825 0.013825| Overall penalty: 0.000000

Run No. 2

Max = 521947.54510 Min = 520475.81084 Avg = 521653.19825
Mutations (real)= 0 ; Mutations (binary) = 128 ; Crossovers = 671
Best ever fitness: 520475.810841 (from generation : 47)
Variable vector: Binary | Real -> 200.000000 200.000000 701.612903 701.612903 100.000000|
Best_ever String = 00000-00000-10011-10011-00000
Constraint value: 2.508065 2.508065 6.979997774245528880000000000000000000e+64 0.356322 0.356322 5.097363 5.097363 5.090224 5.090224 8.512665 8.512665 8.359863 8.359863 19.000000 19.000000 0.002304 0.002304| Overall penalty: 0.000000

Run No. 3

==

Max = 523370.71682 Min = 520533.49868 Avg = 523276.14288
Mutations (real)= 0 ; Mutations (binary) = 111 ; Crossovers = 677
Best ever fitness: 523370.716815 (from generation : 30)
Variable vector: Binary | Real -> 200.000000 200.000000 709.677419 709.677419 100.000000|

Best_ever String = 00000-00000-01011-01011-00000
Constraint value: 2.548387 2.548387 924634782063354531000000000.000000 0.363636 0.363636 5.205918 5.205918 5.198653 5.198653 8.681313 8.681313 8.525821 8.525821 19.000000 19.000000 0.013825 0.013825| Overall penalty: 0.000000

==

Run No. 4

==

Max = 627596.45670 Min = 559623.08240 Avg = 594982.73573
Mutations (real)= 0 ; Mutations (binary) = 115 ; Crossovers = 676No feasible solution found!

==

Run No. 5

==

Max = 520475.81084 Min = 509061.26529 Avg = 520095.32599
Mutations (real)= 0 ; Mutations (binary) = 115 ; Crossovers = 673
Best ever fitness: 520475.810841 (from generation : 27)

==

FOR 12m SPAN GRID FLOOR:

```
================================================================
        INITIAL REPORT
================================================================
Variable Boundaries :
Population size           : 30
Total no. of generations  : 50
Cross over probability    : 0.9000
Mutation probability (binary): 0.0030
Total String length           : 25
Number of binary-coded variables: 5
Total Runs to be performed : 9
Lower and Upper bounds    :
  200.0000  <=  x_bin[1]  <= 350.0000, string length = 5
  200.0000  <=  x_bin[2]  <= 350.0000, string length = 5
  500.0000  <=  x_bin[3]  <= 750.0000, string length = 5
  500.0000  <=  x_bin[4]  <= 750.0000, string length = 5
  100.0000  <=  x_bin[5]  <= 120.0000, string length = 5
======================================================================

Run No. 1
======================================================================
Max = 361275.10059  Min = 343572.08438   Avg = 344162.18492
Mutations (real)= 0 ; Mutations (binary) = 108 ; Crossovers = 668
Best ever fitness: 343572.084379 (from generation : 24)
Variable vector: Binary | Real ->  200.000000 200.000000 612.903226 612.903226
100.000000|
Best_ever String = 00000-00000-01110-01110-00000
Constraint value: 2.064516 2.064516 0.000000 0.368421 0.368421 5.357027 5.357027
7.505695 7.505695 7.505695 7.505695 19.000000 19.000000 19.000000 19.000000
0.021505 0.021505| Overall penalty: 0.000000
======================================================================


Run No. 2
======================================================================
Max = 341287.09120  Min = 341287.09120   Avg = 341287.09120
Mutations (real)= 0 ; Mutations (binary) = 128 ; Crossovers = 671
Best ever fitness: 341287.091197 (from generation : 23)
Variable vector: Binary | Real ->  200.000000 200.000000 604.838710 604.838710
100.000000|
Best_ever String = 00000-00000-10110-10110-00000
Constraint value: 2.024194 2.024194 0.000000 0.360000 0.360000 5.228119 5.228119
7.333808 7.333808 7.333808 7.333808 19.000000 19.000000 19.000000 19.000000
0.008065 0.008065| Overall penalty: 0.000000
======================================================================
```

Run No. 3

==

Max = 420533.53792 Min = 372594.99041 Avg = 393441.55078
Mutations (real)= 0 ; Mutations (binary) = 111 ; Crossovers = 677No feasible solution found!

==

Run No. 4

==

Max = 425079.70054 Min = 368263.94747 Avg = 391306.48949
Mutations (real)= 0 ; Mutations (binary) = 115 ; Crossovers = 676No feasible solution found!

==

Run No. 5

==

Max = 353444.35559 Min = 344609.80133 Avg = 344904.28647
Mutations (real)= 0 ; Mutations (binary) = 115 ; Crossovers = 673
Best ever fitness: 344609.801326 (from generation : 16)
Variable vector: Binary | Real -> 200.000000 200.000000 604.838710 629.032258 100.000000|
Best_ever String = 00000-00000-10110-00001-00000
Constraint value: 2.024194 2.145161 5.682388768885906780000000000000000000000e+281 0.360000 0.384615 5.559170 5.303594 7.775235 7.434447 7.775235 7.434447 19.000000 19.000000 19.000000 19.000000 0.008065 0.048387| Overall penalty: 0.000000

==

Run No. 6

==

Max = 404340.39591 Min = 340522.73808 Avg = 376182.93962
Best ever fitness: 417199.534434 (from generation : 16)
Variable vector: Binary | Real -> 296.774194 267.741935 685.483871 733.870968 113.548387|
Best_ever String = 00101-01110-11101-10111-10101
Constraint value: 1.309783 1.740964 6.571083598808159630000000000000000000000e+89 0.435294 0.472527 9.483366 8.796204 13.021919 12.101422 13.021919 12.101422 19.000000 19.000000 19.000000 19.000000 0.142473 0.223118| Overall penalty: 0.000000

6.7 COST COMPARISION

Table2.Cost comparison of grid floors with different spans

	12m(Run 1)	12m(Run 7)	14m(Run 7)	16m(Run1)	16m(Run 2)
m	200	200	200	200	200
m	200	200	200	200	200
m	612	604	720	516	532
m	612	612	720	516	532
m	100	100	100	100	100
	12	.12	14	16	16
x1 sqmm	1198	1191	1298	1844	1818
y1 sqmm	1198	1211	1298	1844	1818
bx2 sqmm	1003	998	1299	1632	1610
by2 sqmm	1003	1031	1299	1632	1610
bx3 sqmm	671	668	985	1632	1610
by3 sqmm	671	671	985	1632	1610
sbx4 sqmm	1198	668	994	633	639
sby4 sqmm	1198	671	994	633	639
sbx5 sqmm	671	1191	714	633	639
sby5 sqmm	671	1211	714	633	639
otal Cost Rs	343572	342395	497351	592526	597768

6.8 CONVERGENCE HISTORY FOR THE PROBLEM

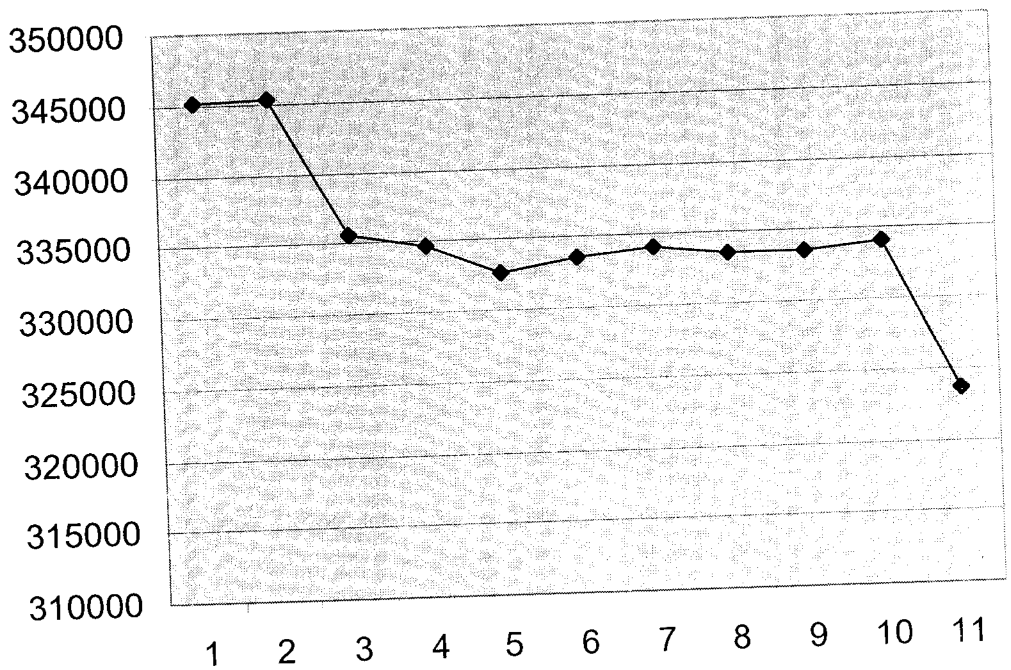


Fig.7 Cost Vs Generation for span 12m-run 7

Cost Vs Generation for span 12m-run 7

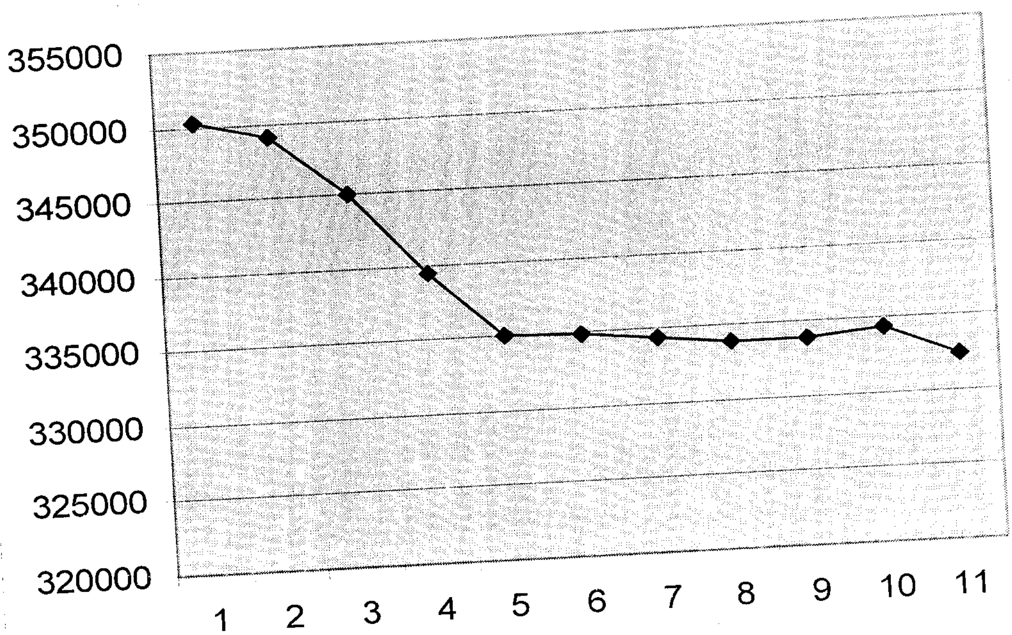


Fig.8

Cost Vs Generation for span 14M

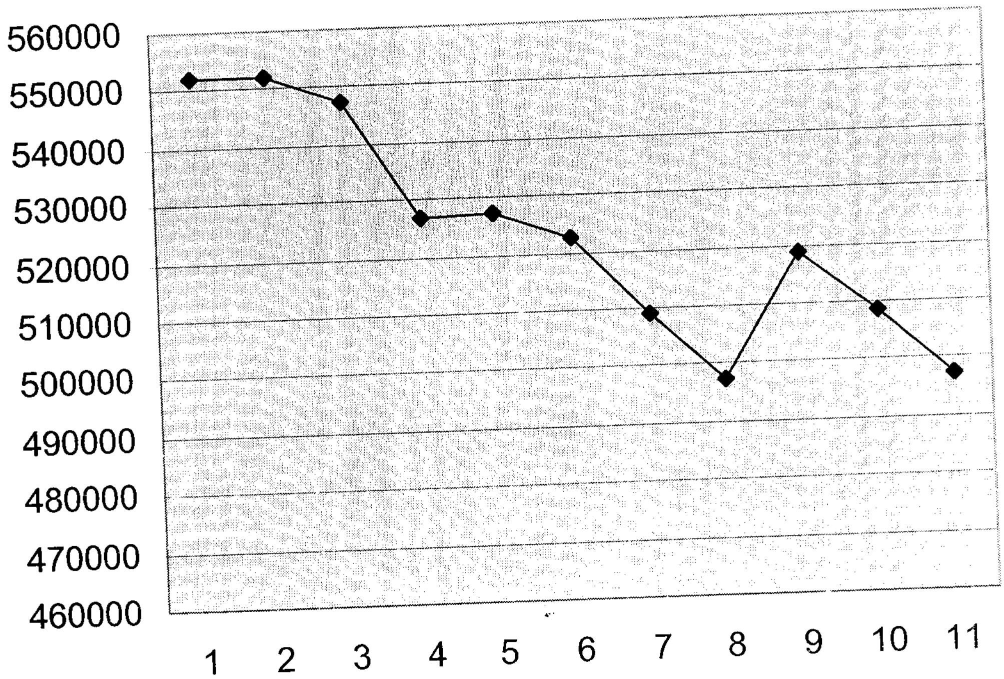


Fig.9

Cost Vs Generation for span 16m-run1

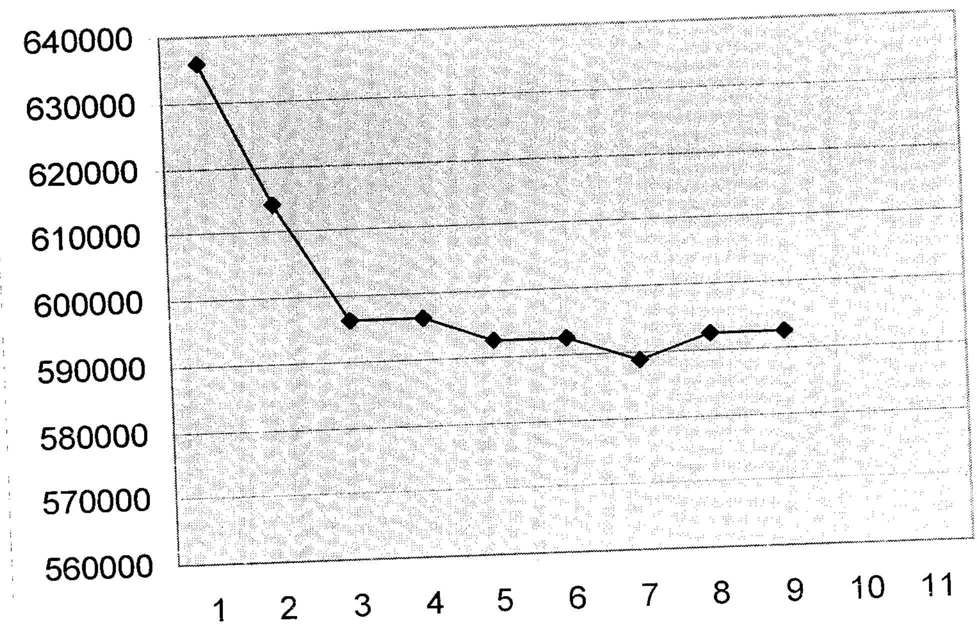


Fig.10

Cost Vs Generation for span 16m-run2

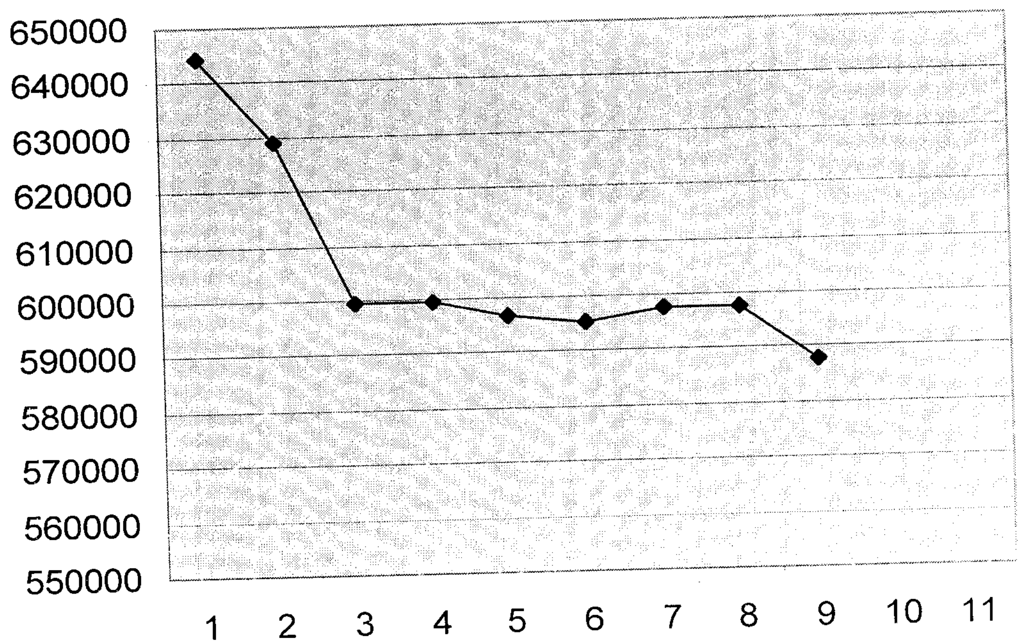


Fig.11

CHAPTER – 8

CONCLUSION

following conclusions are drawn from the present study.

1. Optimization using genetic algorithm is been worked out using 'c' programming and the result obtained over generations are given. The cost variation is found for spans of different length.
2. The cost reduction over successive generations was found
3. The designer have control over the result by specifying the design variable, constraints and requirements
4. The method proposed is less mathematically complex and easier than traditional optimization techniques.

REFERENCES

1. S.R. Adidam, N.G.R. Iyengar and G.V. Narayanan, "Optimum design ofT-Beam and grid floors", Volume 6, October 1978, Pgno 113 – 124
2. N.G.R. Iyengar , "Optimization in Structure design"
3. Goldberg, D.E, "Genetic Algorithm in search, optimization and machine leaning", Addison Wesley, Newyork, 1989.
4. S. Rajasekaran and G.A Vijayalakhmi Pai," Neutral Networks, fuzzy Logic and Genetic Algorithm synthesis and applications", 2003
5. S. Rajeev and C.S. Krishnamoorthy," Discrete optimization of structural engineering".
6. V. Govindaraj and J.V. Ramasamy, "Optimum design of reinforced concrete rectangular columns using genetic algorithm," Journal of structure engineering, V0133, No2, June – July 2006
7. V. Govindaraj and J.V. Ramasamy, "Optimum detailed design of reinforced concrete frames using genetic algorithm," Engineering optimization, volume 39, June 2007, PgNo 471 – 494.
8. Charks. V. Champ, Shahram Pzeshk and Hakan Hansson,"Flexural design of reibforced concrete frames using a genetic algorithm", Journal of structural engineering, January 2003.
9. M. Nehdi and T. Greeenough, " Modeling shear capacity of RC Slender beams without stirrups using genetic algorithm", Smart structures and systems, Volums 3, 2007.
10. A.P. Alex and R. Janes,"Slab formwork design using genetic algorithm," construction informatics Digital libraryS.R. Adidam, N.G.R. Iyengar and G.V. Narayanan, "Optimum design of T-Beam and grid floors", Volume 6, October 1978, Pgno 113 – 124

11. Hai gong, Tsc- Yung P.Chang and Guo – Qiangli, "Multi-level optimization for structural Design of Tall Buildings,"6th world congresses of structural and multi ciplinary optimization, Rio de Janeiro, June 2005.
12. "Code of Practice for plain and reinforced Concrete IS:456 2000, IS New Delhi.
13. S.R. Karve and V.L. shah," Limit state theory and design of reinforced concrete,"1989.
14. Krishnaraju," Advanced reinforced concrete design",.
15. P.C. Varghese," Limit State design of reinforced concrete," Prentice Hall of India, 2004, New Delhi M.

Mahendhirapuri, Mallasamudram (W) Vadugapalayam (P.O.),
Tiruchengode (Tk.), Namakkal (Dt.) - 637 503

An ISO 9001:2000 Certified Institution

Department of Civil Engineering

NATIONAL CONFERENCE ON RECENT TRENDS IN CIVIL ENGINEERING

NCRTC '08

CERTIFICATE

is to certify that Mr./ Ms. B. UMADEVI of
MARAGURU COLLEGE OF TECHNOLOGY presented paper entitled GRID FLOOR OPTIMIZA-
N USING GENETIC ALGORITHM in **NCRTC '08** held at Mahendra Engineering College
24 March 2008.

Chairman

Convenor