10. OPTIMIZATION VIA DISTRIBUTED GENETIC ALGORITHM

10.1 SEARCH FOR OPTIMUM VIA GA

The recent developments in the application of biological principles in to computational algorithms have produced a different class of numerical optimization methods. Among these Genetic Algorithms [124] are powerful search techniques that are used successfully to solve problems in many different disciplines like business, science and engineering. Genetic Algorithm (GA) may be defined as a search algorithm based on the mechanics of natural selection and natural genetics. It is combination of the process of Darvin's law of survival of the fittest and structured yet randomized information exchange with some flair of intuitive search.

For optimization of large structural system Genetic Algorithms have been successfully used since long, due to their robustness and efficiency in obtaining a good optimal solution. In structural engineering problems, optimization often aims at arriving at optimum topology, configuration and cross sectional parameters of members such that the cost or weight of the structure is minimum. Optimization of large structures such as high-rise buildings, space structures etc. subjected to constraints of design codes requires lot of time and that is why it becomes difficult to apply this technology to large-scale complex real life problems.

To speed up the computations in conventional optimization algorithms like those based on optimality criteria method, due to interdependencies in different design iterations, parallelism can be obtained at subdomain or substructure level, which may require larger communications. On the other hand biologically inspired genetic algorithms require higher number of structural analysis, but as all analysis can be done concurrently and independently, inter-processor communication will be very less. Due to this reason, Genetic algorithms have been successfully implemented over loosely coupled coarse grain system.

This chapter reports preliminary study carried out for implementation of distributed genetic algorithm over a network of personal computers using

WebDedip environment running on Windows operating system. In a GA-based structural optimization algorithm 90 – 95% of total time is spent in objective function evaluation. So, objective function evaluation is distributed over number of slave computers and population is stored on master computer. This approach is illustrated here by a small example of optimization of truss but it can be applied to any large size and complex structural system.

10.2 GENETIC ALGORITHMS - FUNDAMENTALS

Genetic algorithms are developed by applying the principle of survival of the fittest into a numerical search method. They are used as function optimizers particularly when the variables have discrete values. An admissible design is represented as an individual and a set of such admissible designs termed as the population. Populations of these designs or individuals are allowed to evolve over a number of generations under control of genetic algorithm. In every generation a new set of individuals is created using bits and pieces of the fittest of the old generation, additionally an occasional new part is tried for good measure. The randomized genetic algorithm efficiently exploits the historic information to speculate on new search points with expected improved performance. A genetic algorithm is different from traditional optimization approach in number of ways, which may be enumerated as follows:

- A GA works with a coding of the parameter set, not the parameter themselves.
- A GA searches from a population of points, not a single point.
- A GA uses payoff (objective function) information, not derivatives or other auxiliary knowledge.
- A GA uses probabilistic transition rules, not deterministic rules.

Simplicity of operation and power of effect are two of the main attractions of genetic algorithm approach. The mechanics of simple genetic algorithm are surprisingly simple in concept and in computer implementation. A simple genetic algorithm that yields good results in many practical problems is composed of three operators: reproduction, crossover and mutation.

Reproduction is a process in which individual strings are copied according to their fitness value. Intuitively the fitness function may be thought of as some measure of the function to be maximized. This means that strings having higher fitness have a probability of contributing one or more offspring in next generation. After reproduction, strings of newly reproduced members are selected and crossed over at random. Crossover is a process of creating new string by swapping characters between two arbitrary positions of two selected strings from mating pool. The action of crossover with previous reproduction speculates on the new ideas constructed from the high performance population of past trial. Mutation is required to recover genetic material lost over the generation by changing 1 to 0 and vice versa in binary coded string, and may also be interpreted as a way to climb out of a local optima.

10.3 GA BASED MINIMUM WEIGHT DESIGN OF TRUSS

The objective function in optimization of truss system [125] is to minimize the weight, it can be written as,

$$f(x) = \sum_{i=1}^{nm} \rho A_i L_i \qquad \ldots (10.1)$$

Where A_i = the cross sectional area of ith member; L_i = the length of ith member; ρ = the weight density of the material. The constraint $g_i(x)$ can be written as,

$$\left.\begin{array}{l} \sigma_i \le \sigma_a, \text{ for } i = 1 \text{ to number of members.} \\ u_j \le u_a, \text{ for } j = 1 \text{ to number of joints.} \end{array}\right] \qquad \ldots (10.2)$$

Where σ_i = stress in member i; σ_a = allowable stress; u_j = horizontal and vertical displacements at joint j; u_a = allowable displacement.

As genetic algorithms are ideally suited for unconstrained optimization problems it is necessary to transform constrained optimization problem into unconstrained problem to solve using GAs. This transformation is achieved by formulation based on the violations of normalized constrains. The constraints are expressed in normalized form as,

$$\frac{\sigma_i}{\sigma_a} - 1 \le 0; \frac{u_j}{u_a} - 1 \le 0 \qquad \ldots (10.3)$$

A violation coefficient C is computed as:

$$\left.\begin{array}{l} \text{if } g_i(x) > 0, \text{ then } c_i = g_i(x); \text{ or if } g_i(x) \le 0, \text{ then } c_i = 0. \\ C = \sum_{i=1}^{m} c_i \quad \text{where m = number of constraints.} \end{array}\right] \qquad \ldots (10.4)$$

Now the unconstrained or modified objective function $\phi(x)$, incorporating constraint violations, is written as

$$\phi(x) = f(x)(1 + KC) \qquad \ldots (10.5)$$

where parameter K has to be judiciously selected depending on required influence of a violated individual in the next generation. The value of K is taken as 10. The expression for fitness of individual is considered as:

$$F_k = [\,\phi(x)_{max} + \phi(x)_{min}] - \phi(x)_k \qquad \ldots (10.6)$$

where F_k is the fitness of kth individual in population, $\phi(x)_{max}$ and $\phi(x)_{min}$ are maximum and minimum value of modified objective functions respectively and $\phi(x)_k$ is value of modified objective functions of kth individual.

The steps of the optimum design process using genetic algorithm are as follows:

(1) Input the geometrical properties of the structure, external loading and number of design variables.

(2) Input the data required for the generation of initial population such as population size, length of substring and a seed value.

(3) Construct the initial population randomly.

(4) Decode each individual and obtain the sequence number for each group in the standard angle section table.

(5) Analyse the structure for different load cases such as dead, imposed and wind and determine its response under these loading cases.

(6) Using relationship (10.3), (10.4) and (10.5) obtain the constraint violation under most unfavorable loading condition and calculate the value of unconstrained or modified objective function $\phi(x)$ for each individual. Determine the maximum and minimum values of this function for the population.

(7) Using relationship (10.6), calculate the fitness value of each individual. By means of average fitness (F_{av}), find the fitness factor of each individual as F_k/F_{av}, which is also known as expected count of individual in mating pool.

(8) Count of individual is converted into an actual count by appropriately rounding off and copy them into mating pool proportional to their fitness.

(9) Couple the individuals randomly and apply crossover operator and produce new off - springs.

(10) Apply mutation operator, if mutation probability adopted is satisfied.

(11) Replace the old generation with the new one and repeat the steps from (4) to (10) until a specified maximum number of generations is reached.

In each generation, the individual which satisfies all the constraints and the least weight is stored and compared with the one obtained in the next generation. At the final generation, the one which is the lightest of all is the optimum result.

The above steps for optimization of truss using genetic algorithm is presented in form of flowchart in Fig. 10.1.

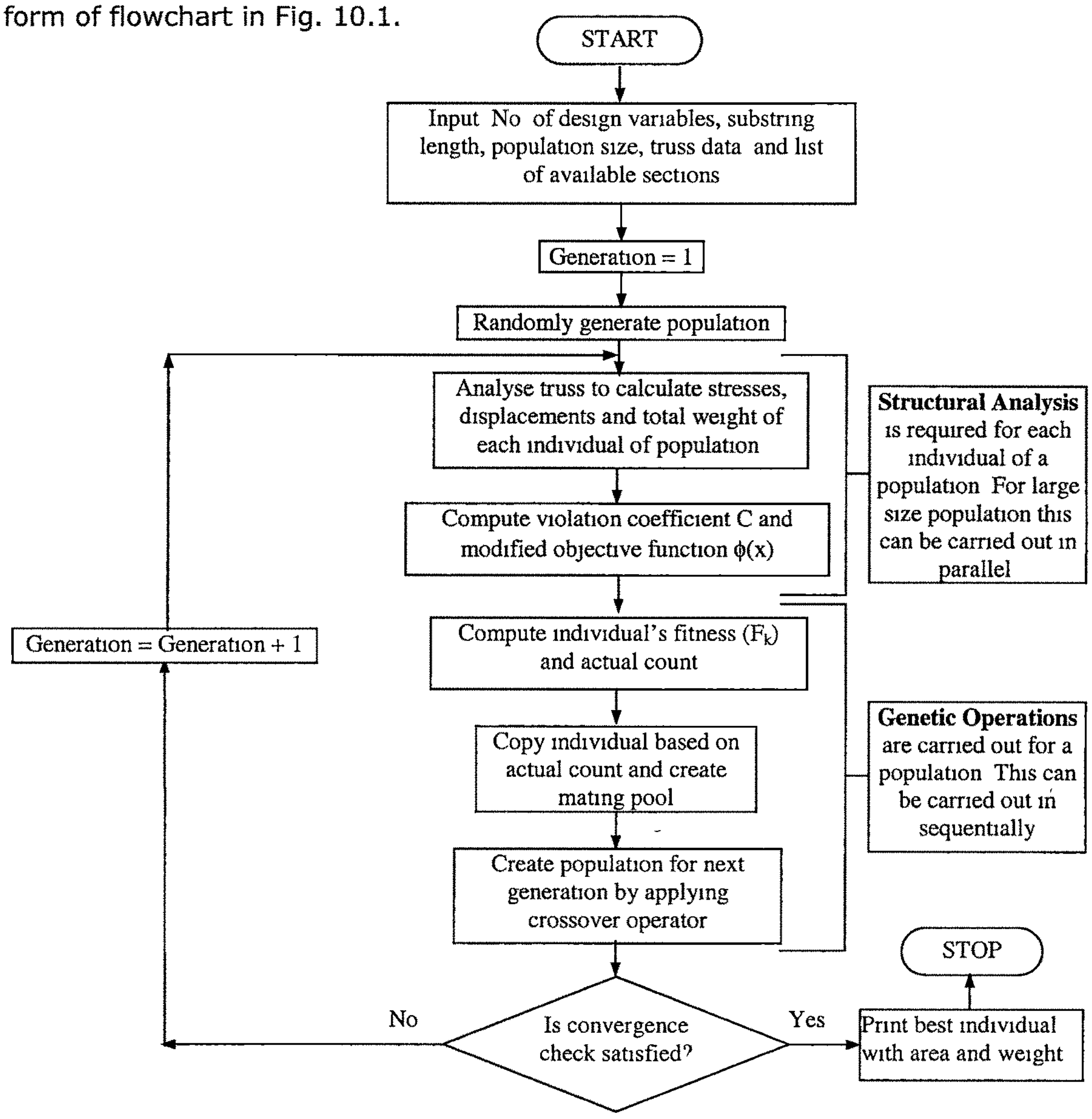

START
Input No of design variables, substring length, population size, truss data and list of available sections
Generation = 1
Randomly generate population
Analyse truss to calculate stresses, displacements and total weight of each individual of population
Compute violation coefficient C and modified objective function ϕ(x)
Compute individual's fitness (F_k) and actual count
Copy individual based on actual count and create mating pool
Create population for next generation by applying crossover operator
Is convergence check satisfied?
No
Yes
Generation = Generation + 1
Print best individual with area and weight
STOP
Structural Analysis is required for each individual of a population For large size population this can be carried out in parallel
Genetic Operations are carried out for a population This can be carried out in sequentially

FIG. 10.1 FLOWCHART OF GA FOR TRUSS OPTIMIZATION

10.4 DISTRIBUTED IMPLEMENTATION OF GA

GA can be implemented on parallel or distributed processing in following ways [126]:

- Evaluation of fitness functions in parallel either a number of individuals at a time or by parallelization of the assessment of the constraint.
- Migration of best individual from one processor to another, which may require inter processor communications.
- Population distribution over processors to run sequential simulation in parallel.

The choice from above methods depends on time required for various tasks of the entire process. The computation time required for each design iteration consists of two main parts: the unconstrained or modified objective function evaluation and genetic operations like reproduction, cross over and mutation. Since structural analysis is required to obtain value of modified objective function of each string, there will be number of analysis equal to number of strings in the population. If time required for objective function evaluation is t sec and there are q strings, then time required for objective function evaluation is qt sec. If g sec is the time required for genetic operations for each iteration then total computation time required for s iterations of design optimization is expressed as (qt + g) sec.

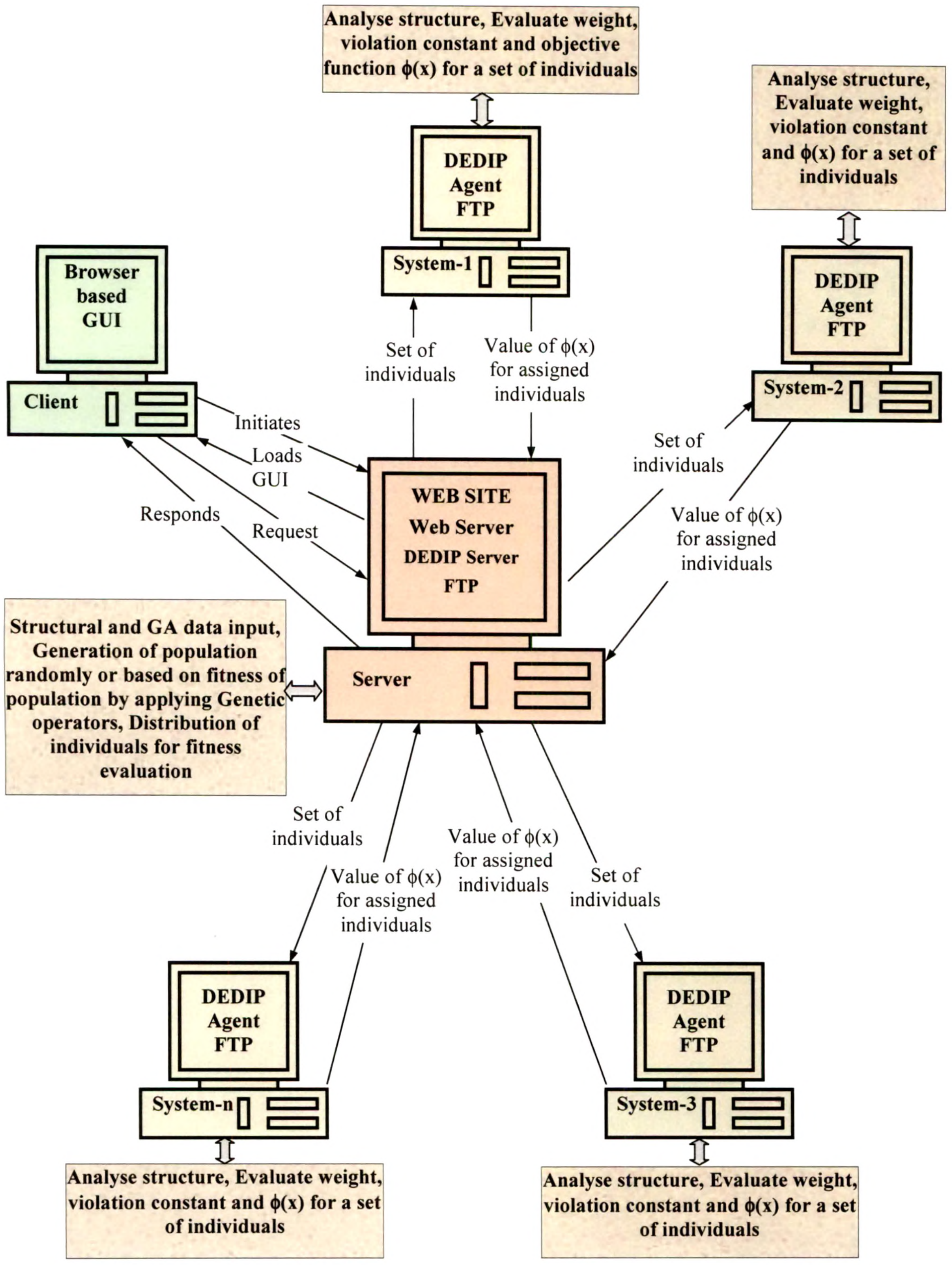
Actual value of q and g varies with the size of the problem, but generally ratio of g/qt is very small. So to speed up computations, attention should be focused on reducing time for objective function evaluation, which depends upon number of string in population and time required for each objective function evaluation.

For balancing load among various processors static load balancing or dynamic load balancing technique can be used. In static load balancing depending on performance of available computers on network, specified numbers of strings are allotted in the beginning of the process. In dynamic load balancing the load of computers i.e. number of strings allotted for objective function evaluation will change during the process also. Dynamic load balancing is better alternative to use computing resources efficiently as often machines are of different speed and computers load in multiuser multitasking environment is varying.

In present work, unconstrained / modified objective function evaluation is done in parallel by distributing strings of population to various slave processors or computers. The genetic operations like reproduction, crossover and generation of new population is done on master computer. For balancing load among various computers, static approach is used, because all machines available during this study are of same configuration. The communication between master and slave computer is done twice for each iteration. In first communication master sends a set of individuals of population for objective function evaluation to number of slave computers. After evaluation the slave computers sends back the value of objective functions for a set of individuals to master computer. Master computer collects information of objective function for all individuals from different slave computers. From this data fitness of each individual is evaluated and by applying reproduction, cross over and mutation operators new population is generated as shown in Fig. 10.2. The individuals of new generation are again distributed amongst the slave computers and this process iterates till result converges.

As the communication of data between master and slave computer is done using FTP, the communication process consists of two parts. First part is authorizing the client or setting up for communication and second part is actual transfer of data. The setup time, also known as latency, is almost constant irrespective of size of data communicated. The transfer time, also known as bandwidth, depend on communication speed of network and size of data. The speedup, the ratio of time required to execute a task sequentially on one computer to time required to execute a task for number of computers, mainly depends on processing time and communication time. For small size problem time required to complete the task is smaller compared to communication time and so speed-up observed is small. On the other hand, for large size problem computation time predominates the communication time and higher speed-up can be observed.

In this work generation of new population is done on master computer only; the quality of search is not affected, as master has global knowledge about the search process. A few alternative distributed GAs, suggested independent processes to handle small subpopulation with periodic communication between processors. This approach improves the degree of parallelism but degrade the quality of search as processes may get struck in local minima.



Analyse structure, Evaluate weight, violation constant and objective function ϕ(x) for a set of individuals
DEDIP Agent FTP
System-1
Analyse structure, Evaluate weight, violation constant and ϕ(x) for a set of individuals
DEDIP Agent FTP
System-2
Browser based GUI
Client
Set of individuals
Value of ϕ(x) for assigned individuals
Initiates
Loads GUI
Responds
Request
Set of individuals
Value of ϕ(x) for assigned individuals
WEB SITE
Web Server
DEDIP Server
FTP
Server
Structural and GA data input, Generation of population randomly or based on fitness of population by applying Genetic operators, Distribution of individuals for fitness evaluation
Set of individuals
Value of ϕ(x) for assigned individuals
Value of ϕ(x) for assigned individuals
Set of individuals
DEDIP Agent FTP
System-n
DEDIP Agent FTP
System-3
Analyse structure, Evaluate weight, violation constant and ϕ(x) for a set of individuals
Analyse structure, Evaluate weight, violation constant and ϕ(x) for a set of individuals

FIG. 10.2 DISTRIBUTED GENETIC ALGORITHM MODEL ON WEBDEDIP

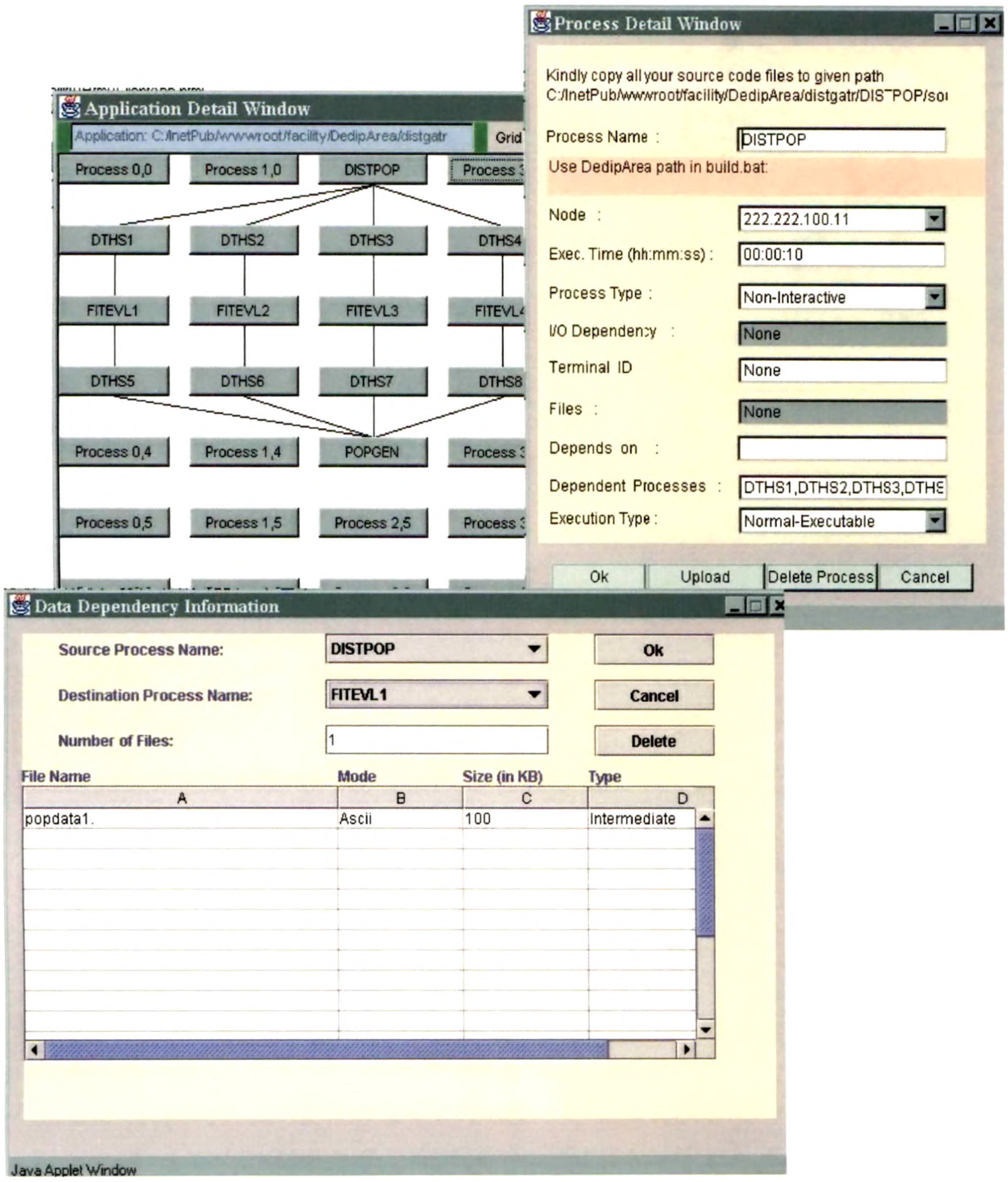
In this study the entire process of optimization of plane truss using GA is divided into following three tasks:

The first task, DISTPOP, in first iteration reads parameters related to GA, i.e. population size, number of sub strings in each population and possible design variables, and generates a set of random population. After first generation, it distributes populations to number of slave computers using static load balancing concept. As in this implementation all computers of LAN are having identical configurations, all computers are assigned equal number of individuals from population for evaluation of modified / unconstrained objective function. In subsequent iteration, this task distributes the new population generated by POPGEN task.

The second task FITEVL, evaluates the objective function based on analysis of the structures considering decoded cross sectional area of populations submitted by first task. As this task is most time consuming, it runs in parallel on different number of computers. It calculates stresses in each member and displacements at each joint and compares with allowable value. It also calculates weight of structure, violation coefficient and modified objective function $\phi(x)$ as discussed in earlier section. This data for all the assigned individuals is sent to master computer.

The third task POPGEN receives the results of FITEVL task and calculates fitness of individuals of population as discussed in section 10.3. Applying genetic operators like reproduction, crossover and mutation, it generates a new population, which in second iteration is distributed to slave computers by first task. The interface among different tasks is carried out using intermediate files. All the tasks are then inter linked using DEDIP GUI to configure the application for parallel processing.

Figure 10.3 shows screen shots for configuring application on four computers. Along with screen shot depicting the required interdependency, a sample screen shot for providing information about a process, DISTPOP, and that for providing information about DTHS (Data Transfer from Host to Slave) is shown. Flow of data through intermediate files between various processes is shown in Fig. 10.4.



Process Detail Window

Kindly copy all your source code files to given path C:/InetPub/wwwroot/facility/DedipArea/distgatr/DISTPOP/so

Process Name :	DISTPOP
Use DedipArea path in build.bat:	
Node :	222.222.100.11
Exec. Time (hh:mm:ss) :	00:00:10
Process Type :	Non-Interactive
I/O Dependency :	None
Terminal ID	None
Files :	None
Depends on :	
Dependent Processes :	DTHS1,DTHS2,DTHS3,DTHS
Execution Type :	Normal-Executable

Ok | Upload | Delete Process | Cancel

Data Dependency Information

Source Process Name:	DISTPOP
Destination Process Name:	FITEVL1
Number of Files:	1

Ok | Cancel | Delete

File Name	Mode	Size (in KB)	Type
A	B	C	D
popdata1.	Ascii	100	Intermediate

FIG 10.3 CONFIGURING THE DISTRIBUTED GA ON FOUR COMPUTERS

The number of individuals in population (or size of population) and number of generations can be varied to get the optimum solution. It is observed that with larger population size number of generations required can be reduced. For smaller size problems, with increase in size of population there is not much increase in computation time but with increase in number of generations communication time increases. As the reduction in number of generations requires less number of communications, larger size population may improve the computational efficiency.

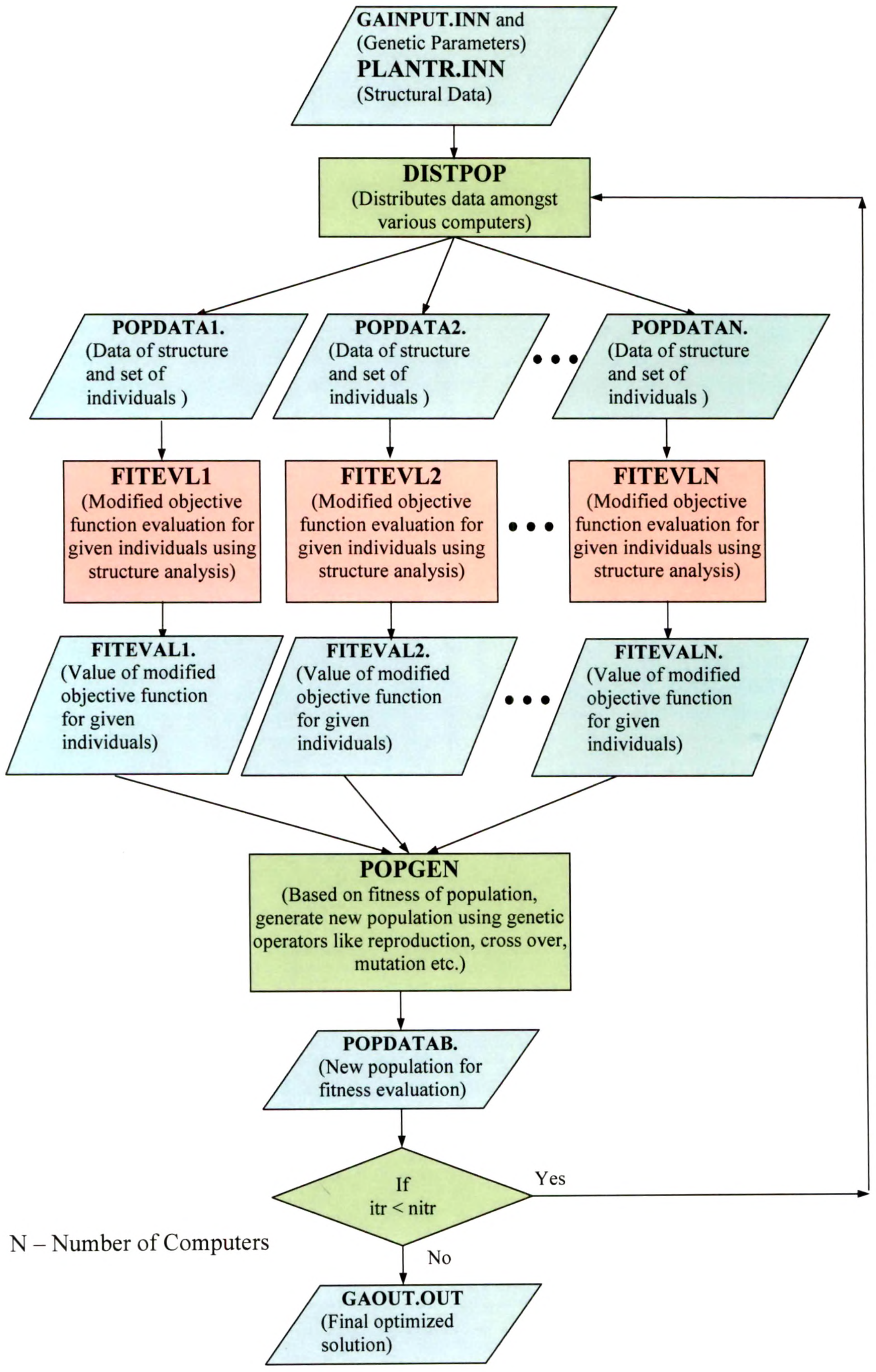

GAINPUT.INN and
(Genetic Parameters)
PLANTR.INN
(Structural Data)
DISTPOP
(Distributes data amongst various computers)
POPDATA1.
(Data of structure and set of individuals)
POPDATA2.
(Data of structure and set of individuals)
POPDATAN.
(Data of structure and set of individuals)
FITEVL1
(Modified objective function evaluation for given individuals using structure analysis)
FITEVL2
(Modified objective function evaluation for given individuals using structure analysis)
FITEVLN
(Modified objective function evaluation for given individuals using structure analysis)
FITEVAL1.
(Value of modified objective function for given individuals)
FITEVAL2.
(Value of modified objective function for given individuals)
FITEVALN.
(Value of modified objective function for given individuals)
POPGEN
(Based on fitness of population, generate new population using genetic operators like reproduction, cross over, mutation etc.)
POPDATAB.
(New population for fitness evaluation)
If itr < nitr
Yes
No
N – Number of Computers
GAOUT.OUT
(Final optimized solution)

FIG. 10.4 FLOW OF DATA IN DISTRIBUTED GA FOR TRUSS OPTIMIZATION

10.5 EXAMPLE OF A PLANE TRUSS OPTIMIZATION

Figure 10.5 shows a 10 bar truss [125] with dimensions, loading and other required parameters. The assumed data are: E = 10^4 ksi (6.89 × 10^4 MPa), ρ = 0.10 lb/in^3 (2.770 kg/m^3) and vertical downward loads of 100 kips (445.374 kN) at joints 2 and 4. The objective of the problem is to minimize the weight of structure. Constraints are imposed on member stresses and displacements. The allowable displacement is limited to 2 in. (50.8 mm) and stresses to ± 25 ksi (172.25 MPa).

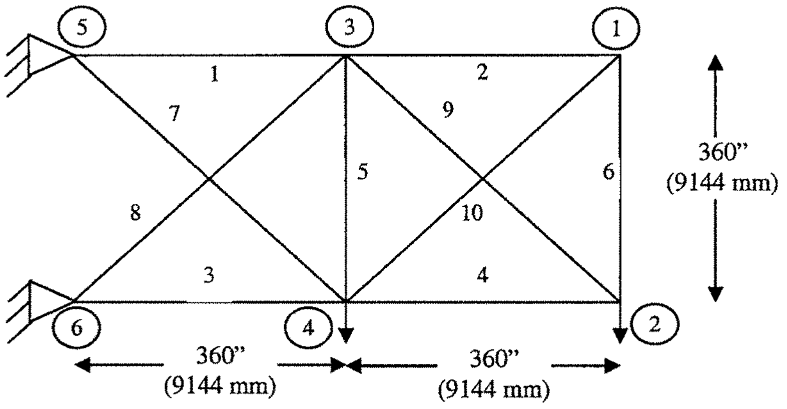


FIG. 10.5 GEOMETRIC DATA OF 10 BAR TRUSS

The generation history for population size 40 is shown in Fig. 10.6. The comparison of results with that available in literature [125] is shown in Table 10.1.

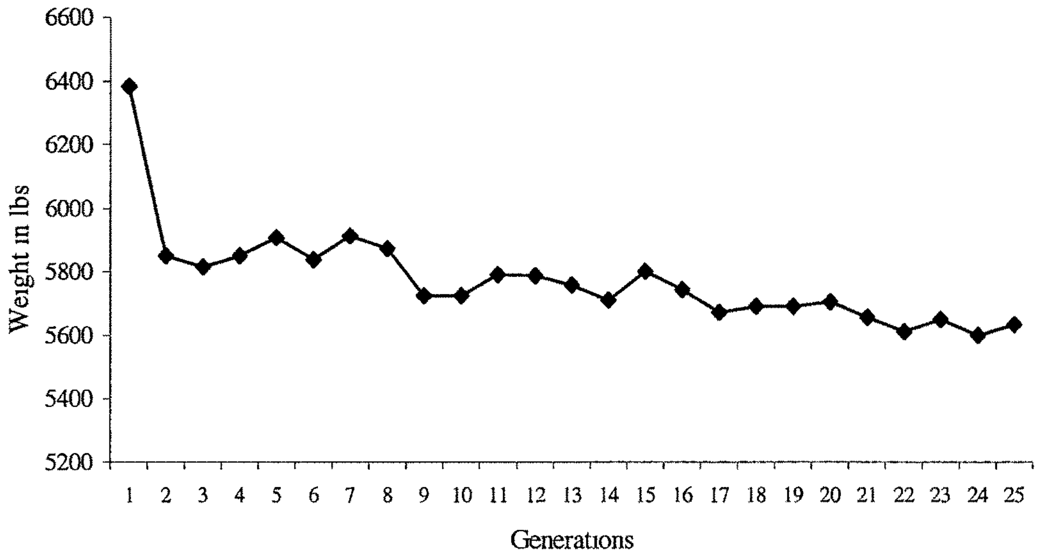


FIG. 10.6 GENERATION HISTORY OF 10 BAR TRUSS

TABLE 10.1 COMPARISON OF RESULTS – PLANE TRUSS PROBLEM

Member	1	2	3	4	5	6	7	8	9	10
Cross section area in in^2 (Rajeev and Krishnamoothy) [125] Wt. = 5613.84 lbs (2548.7 kg)	33.5	1.62	22.0	15.5	1.62	1.62	14.2	19.9	19.9	2.62
Cross section area in in^2 (Present study) Wt. = 5598.78 lbs (2541.9 kg)	30.0	2.13	26.5	15.5	1.99	1.62	7.97	22.0	22.9	2.13

10.6 EXAMPLE OF A SPACE TRUSS OPTIMIZATION

The configuration, dimensions and loading of 25 – bar truss are shown in Fig. 10.7. The grouping of member is shown in Table 10.2.

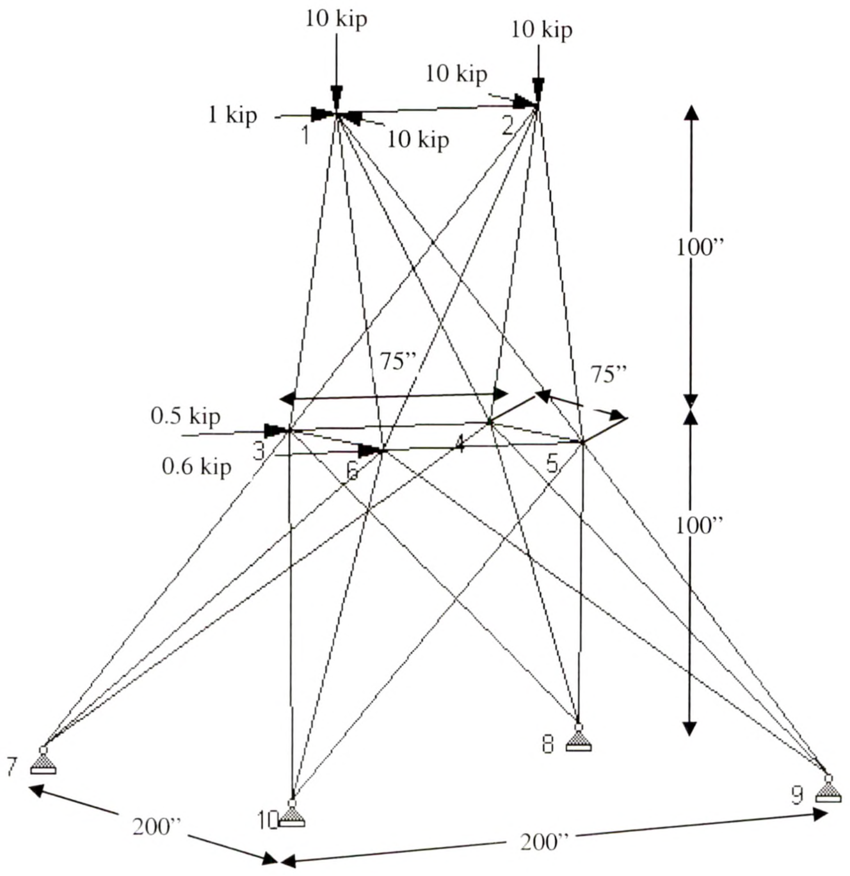


FIG. 10.7 STRUCTURAL DATA FOR A 25 BAR TRUSS

The data for analysis is taken as E = 10^4 ksi (6.89 × 10^4 MPa); ρ = 0.10 lb/in^3 (2770 kg/m^3). The displacements at joint 1 and 2 in the X and Y are restricted to be less than ± 0.35 in (8.89 mm). The stresses are limited to ± 40 ksi (275.6 MPa).

TABLE 10.2 DETAILS OF MEMBER GROUPING

Group No.	Members connected to joints
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

The set of available sections used for this problem given is S = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6. 2.8, 3.0, 3.2, 3.4] (sq. in).

Assuming that each variable will take value from any of 16 values, size of substring is taken as 4. As there are 8 groups of members, total length of each string will be 32. The comparison of results obtained by considering population size 20, 30, 40 with that given by Rajeev and Krishnamoorthy [125] is shown in Table 10.3. Generation history is shown in Fig. 10.8.

In the distributed implementation, the task of objective function evaluation is carried out using different number of computers on the network. WebDedip has facility, which gives the summary of application indicating the status of various processes i.e. node number (IP address), start time and end time. The time consumed for the computation and communication can be obtained from this summary when process is distributed over number of computers. When the problem of space truss is implemented over distributed processing the time spent for computation and communication is shown in Table 10.4.

TABLE 10.3 COMPARISON OF RESULTS – 25 BAR TRUSS PROBLEM

Method	Weight	A1 in²	A2 in²	A3 in²	A4 in²	A5 in²	A6 in²	A7 in²	A8 in²
Rajeev & Krishnamoorthy Population size 20	546.76 lbs (248.23 kg)	0.20	1.80	2.30	0.20	0.10	0.80	1.80	3.00
Rajeev & Krishnamoorthy Population size 30	546.01 lbs (247.89 kg)	0.10	1.80	2.30	0.20	0.10	0.80	1.80	3.00
Rajeev & Krishnamoorthy Population size 40	546.01 lbs (247.89 kg)	0.10	1.80	2.30	0.20	0.10	0.80	1.80	3.00
Present Study Population size 20	**538.91 lbs (244.67 kg)**	**0.30**	**1.60**	**2.60**	**0.20**	**1.30**	**1.40**	**0.80**	**3.00**
Present Study Population size 30	**514.78 lbs (233.71 kg)**	**0.20**	**0.80**	**3.20**	**0.30**	**1.20**	**0.90**	**1.20**	**3.00**
Present Study Population size 40	**528.68 lbs (240.02 kg)**	**0.30**	**1.20**	**2.80**	**0.10**	**0.60**	**1.10**	**1.0**	**3.40**

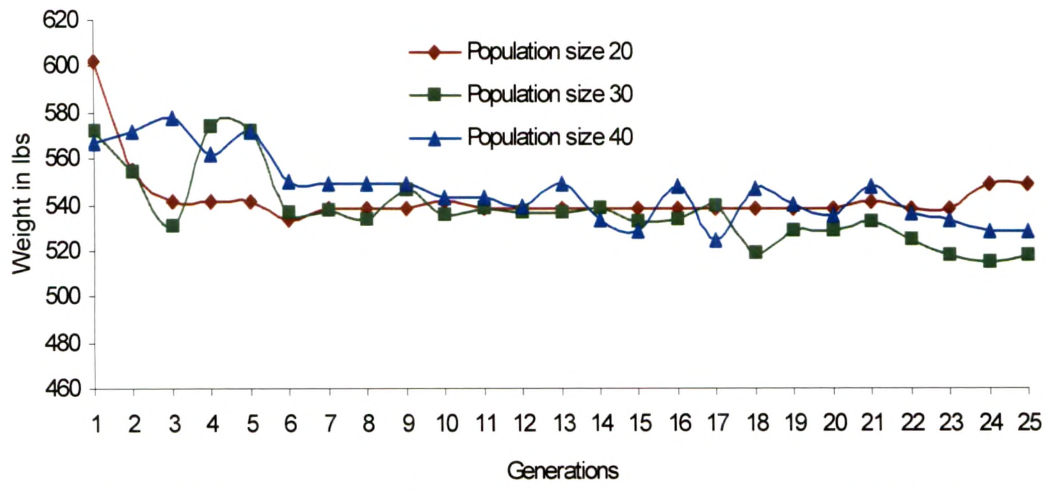


FIG 10.8 GENERATION HISTORY – 25 BAR TRUSS PROBLEM

TABLE 10.4 COMPARISON OF TIME

No. of Computers	Computation Time (Sec)	Communication Time (Sec)	Total Time (Sec)	Speedup
1	102	-	102	1
2	61	19	80	1.28
3	53	22	75	1.36
4	47	25	72	1.42
5	43	29	72	1.42

From the observation of time the speed-up is calculated. Due to time involved in communication of data between the computers, observed speed-up is lower than ideal speed-up. The speed-up observed is shown in Figure 10.9. As the size of problem implemented is small, the advantage of distributed processing is not very high. However, the generalized programs prepared for this study can be utilized for any size of pin jointed plane / space problem without any modifications. For larger size problem better computational efficiency is expected.

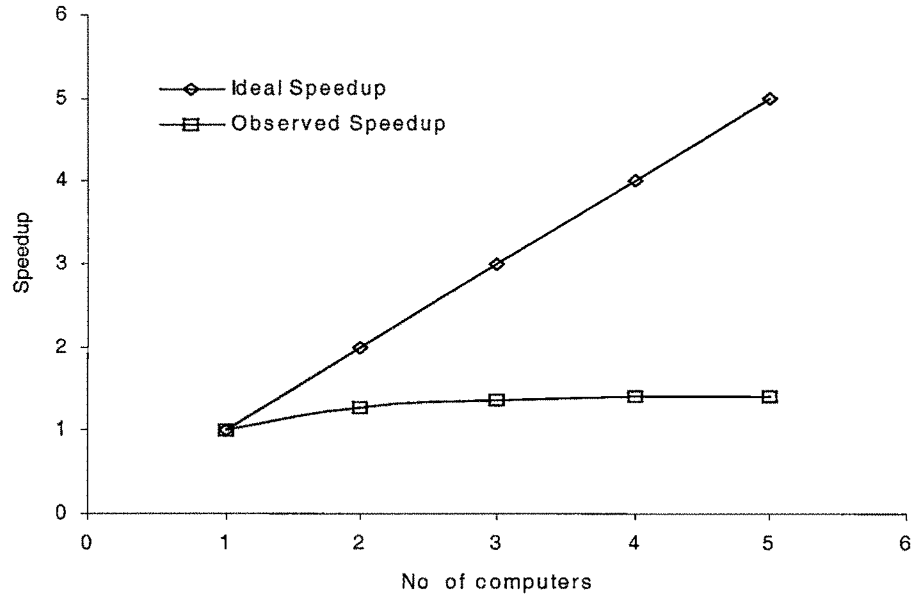


FIG 10.9 COMPARISON OF SPEED-UP

10.7 DISCUSSION OF RESULTS

The use of Genetic Algorithms for optimization of structural system is increasing due to its robustness and efficiency. As the process is computationally intensive for large size problems, it can be made faster by implementing on distributed processing. In the present study, network of personal computers was utilized as distributed computing resource with WebDedip environment. For the small size problem, taken up here to investigate the possibility of implementing distributed GA using WebDedip environment, an efficiency of order 40-50% was observed. But for large size problem higher efficiency can be anticipated. Also to enhance speed up number of iterations can be reduced by increasing size of population, as more number of iterations requires larger communication time.