CHAPTER 4

MATRIX STORAGE OPTIMIZATION IN FINITE ELEMENT ANALYSIS BY NODE RENUMBERING

The cracking localization analysis method proposed in this study involves finding a crack pattern which has the minimum total potential energy. As a result, many system stiffness matrices for different crack patterns are formed and computed in order to obtain the energy of their systems. Moreover, for accurate results, the finite element mesh employed should be rather fine in order to allow as many cracks as possible to occur. For this reason, a solver has to form massive system stiffness matrix equations which consumes high resource of computer and takes long computational time. Therefore, minimization of computational time and storage of stiffness matrices is inevitable.

4.1 Minimization of Matrix Storage

The system stiffness matrix is rather sparsely or weakly populated in most structural analysis problems. This is because each row or equilibrium equation for a particular degree of freedom is only influenced by degrees of freedom associated with the often small number of elements connecting to that degree of freedom. All other degrees of freedom for remaining unattached elements have no effect on this equilibrium equation and hence have zero or void stiffness coefficients in that row. Therefore, the sparseness or profile matrix is usually used to store the system stiffness matrix in the effective solvers. It is indicated by clustering of the nonzero stiffness coefficients about the main diagonal (see Fig. 4.1). The opportunity to gain the efficiency in solving can be realized if we note all elements outside the sparseness that always retain zero value. Hence, the performance of these schemes can be improved by modifications that avoid the storage and manipulation of the useless zeros outside the clustering.

The element stiffness matrices are assembled to form a system stiffness matrix according to the degrees of freedom of the elements which are commonly assigned by the node numbers. Actually, the numbering sequence of the nodes has no influence on the result, but it influences the computational time and the requirement of storage space of the system stiffness matrix. If the nodes are numbered in an appropriate sequence, the coefficients in the system stiffness matrix are arranged close to the

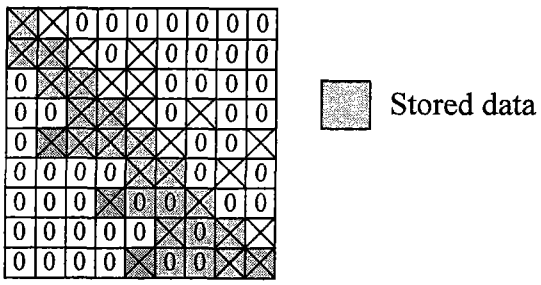
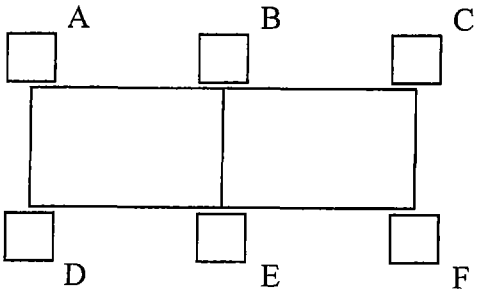
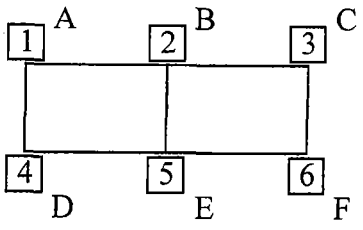
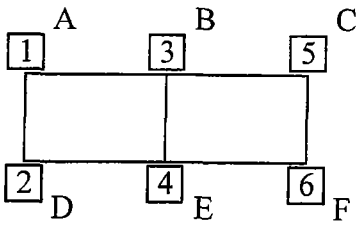


Fig. 4.1 Storage of profile symmetric matrix



(a) Finite element mesh





E1

A	B	C	D	E	F
1	2	3	4	5	6

E2

A	B	C	D	E	F
1	3	5	2	4	6

(B) Example of node numbering

Fig. 4.2 Illustration of node renumbering

diagonal of the matrix. Thereafter, the profile matrix can store fewer components and certainly use lower computational time. In summary, the objective of the matrix storage optimization is to find a node numbering sequence that makes most of the coefficients of the system stiffness matrix close to the diagonal of the matrix.

One of the famous optimization problems is the traveling salesman problem. The goal of this problem is to find the shortest route passes all prescribed cities that a salesman has to visit, in other words, the best sequence of cities to visit. However, there is a condition that the salesman can visit each city only once. In our optimization of the system matrix storage, we consider renumbering of node numbers. Since each node number cannot be repeated in a finite element mesh, different node numbering can be considered as different sequences of numbers. If each node number is thought of as a city name, then the problem can be considered as the traveling salesman problem. In order to illustrate this idea, consider blank boxes shown in Fig. 4.2a. Assume that the name of the first city that the salesman visits is placed in the A box and the second city is in the B box and so on. The different node numberings E1 and E2 indicate the different order of cities to visit (see Fig. 4.2b). In E1, the sequence of the cities is 1,2,3,4,5, and 6 while, in E2, the sequence is 1,3,5,2,4, and 6. These two sequences will lead to a different storage of the system matrix.

From the above similarity, the minimization of the matrix storage in this study will be based on the genetic algorithm for the traveling salesman problem. The algorithm will follow the ordinary genetic algorithms mentioned in chapter 3 except for the details of coding, crossover and mutation.

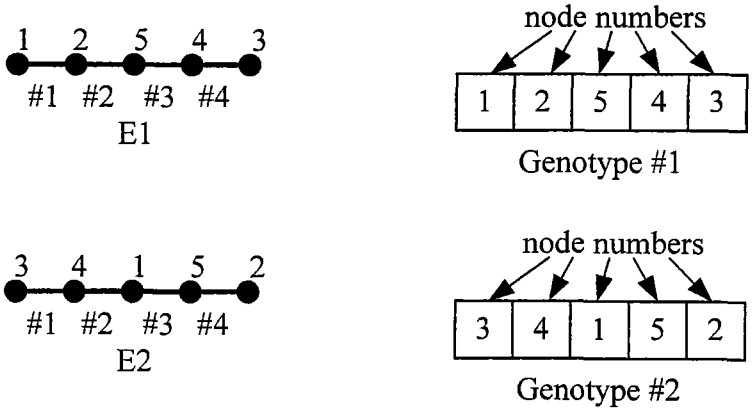


Fig. 4.3 Coding of different node numberings

4.1.1 Coding

In usual coding for genetic algorithms, binary strings are often used. Nevertheless, for the traveling salesman problem, strings of city names can be directly used. For example, consider two meshes for a bar shown in Fig. 4.3. The genotypes that represent these two different numberings are the string of the node numbers. Coding the sequence of node numbers this way is to make certain that it will be possible to find appropriate crossover and mutation operators that will not result in repeated node numbers in the mesh.

4.1.2 Reproduction operator

The function of the reproduction is to reproduce or to duplicate the high fitness genotypes and discard the low-fitness genotypes from the population. In this study, genotypes with high fitness are those representing node numberings that require low matrix storages. For the reproduction operator in this study, a traditional procedure of the reproduction operator for a simple genetic algorithm is employed (see section 3.4.3).

4.1.3 Crossover operator

The function of crossover operator is to create different genotypes (children strings) by exchanging the data (chromosomes and bits) between the existing genotypes (parent strings). The traditional crossover algorithm, which is a crossover of binary-coded string as described in chapter 3, cannot be applied to integer-coded strings used in this problem because it might cause repetition or loss of some node numbers. Therefore, a different type of crossover algorithm must be used to avoid such problems.

Usually, the crossover operator needs two parent strings and returns two children strings as the outputs. The crossover operator used here also needs two parent strings and gives two children strings. Nevertheless, this algorithm is divided into two parts, i.e., the first part for the first children string and the second one for the second children string. It begins with random selection of the crossing site (see Fig. 4.4a). In the figure, P1, bits on the left side of the crossing site are kept. The node numbers in

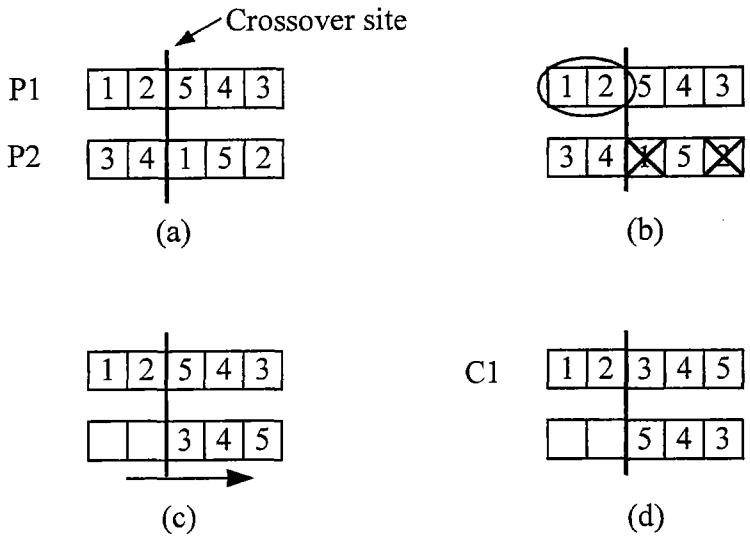


Fig. 4.4 Schematic diagram of crossover of node numberings

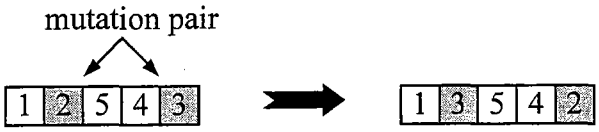


Fig. 4.5 Schematic diagram of mutation of node numberings

P2 that appear on the left side of P1 are removed from P2 (see Fig. 4.4b). Then, the bits in P2 are moved to the right by keeping the order of the node numbers (see Fig. 4.4c). Finally, all bits on the right side of the crossing site between P1 and P2 are swapped (see Fig. 4.4d) and the children string C1 is obtained. In order to create the second children string C2, the same procedure can be done. The difference is only that at the beginning bits on the left side of P2 instead of P1 are kept unchanged.

4.1.4 Mutation operator

The mutation operator is another operator to create new genotypes. The difference is that the crossover operator needs another genotype to exchange the data while the mutation operator manipulates the data within itself. Certainly, the traditional mutation operator in section 3.4.3 cannot be used. The new algorithm begins by selecting the two exchange bits at random then switches the node numbers between the two bits (see Fig. 4.5).

4.2 Example Problems

In order to illustrate the advantage of the proposed method in the optimization of matrix storage, the following problems are considered, i.e.,

- Plate using eight-noded quadrilateral elements
- Space truss using two-noded line elements
- Meshes using four-noded quadrilateral elements

4.2.1 Plate using eight-noded quadrilateral elements

Here, a finite element mesh consists of 8 eight-noded elements, 37 nodes and the search space is 1.376×10^{43} patterns (example 3 in Gajewski and Lompies 1996) is considered. It is named mesh A. The finite element mesh and its original numbering are shown in Fig. 4.6. The system stiffness matrix is shown Fig. 4.7. It is obvious that it is a symmetric matrix. Therefore, only half of its components will be stored in the lower triangular and along its diagonal shown as shaded area. The original storage is 449 coefficients and the maximum storage is 703 coefficients.

In this study, the genetic algorithm's parameters were taken as: maximum generation= 5,000, population size=100, number of chromosome=37, probability of crossover p_c=0.8 and probability of mutation p_m=0.05. During the process of genetic algorithm, the average of fitness value of genotypes in a generation slightly increases on the longer generation and converges to 0.0018 (Fig. 4.9). For the storage of matrix, it is obvious that its average reduces and converges in the same generation as obtained in average of fitness (Fig. 4.8). This is a natural behavior of the minimization problem because the less objective value is the higher fitness value. Up to this point, after the result converges, the process can be terminated and the optimum solution can be obtained. For example, the process can be terminated at the 2000^{th} generation because the optimum solution may not be changed with process on the further generations. Note that the optimum solution was obtained during the process, not in the last generation.

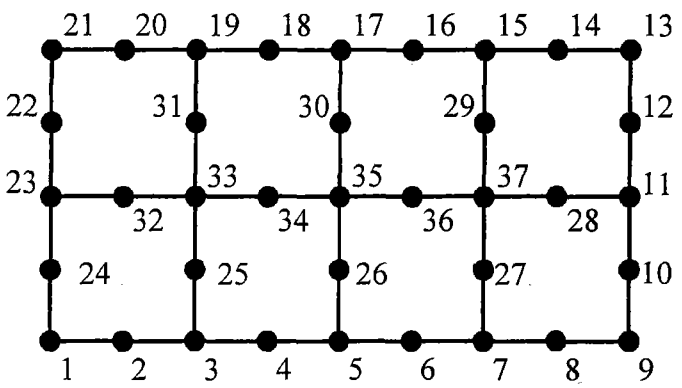


Fig. 4.6 Finite element mesh of plate using eight-noded quadrilateral elements

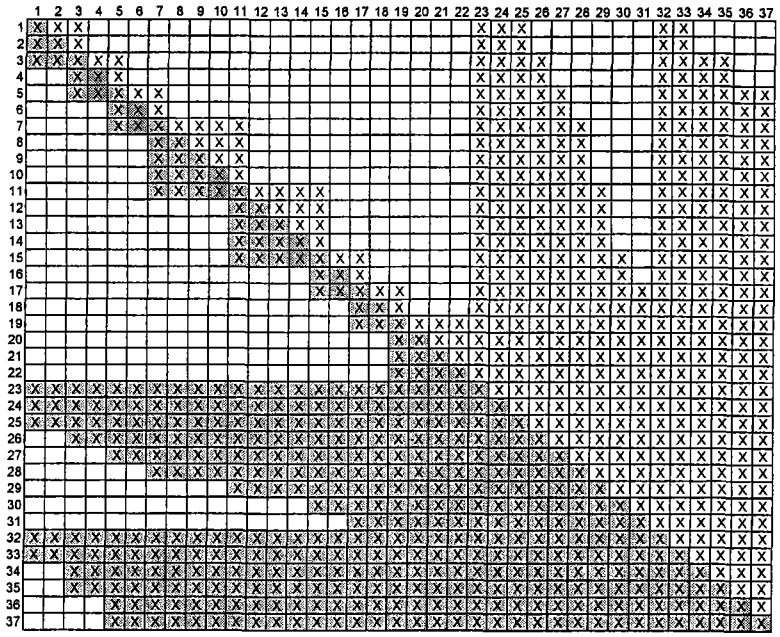


Fig. 4.7 Original system stiffness matrix of mesh A

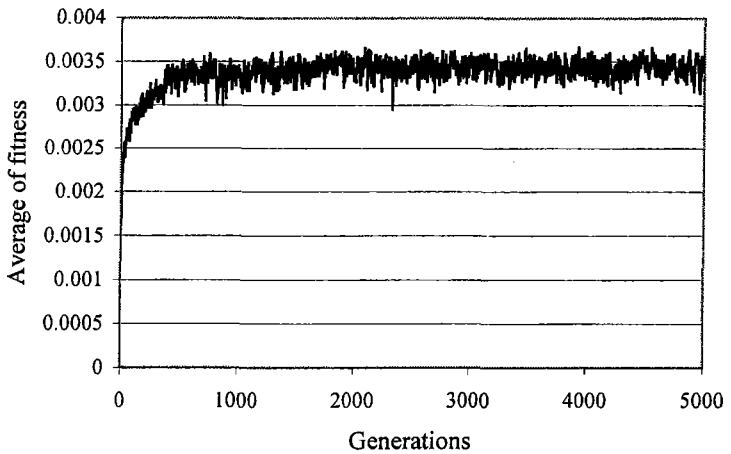


Fig. 4.9 Relationship between average of fitness values and number of generations

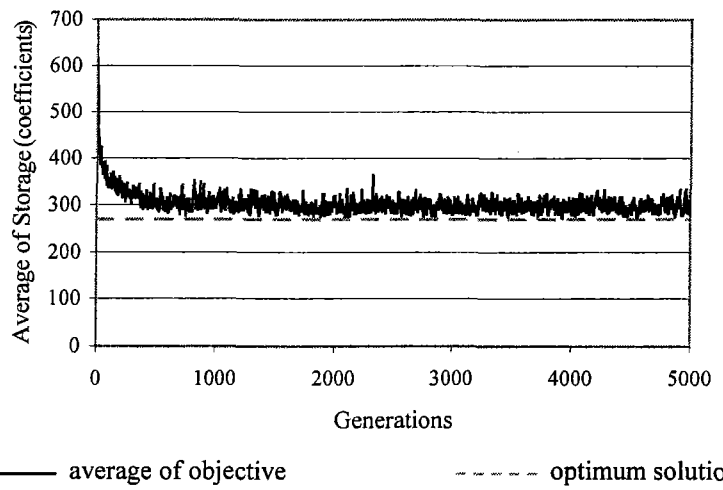
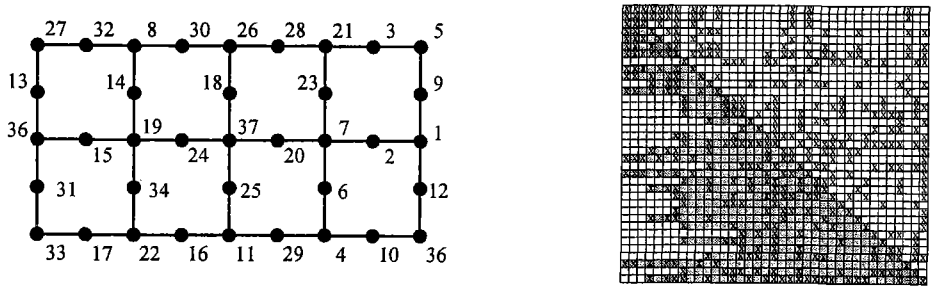
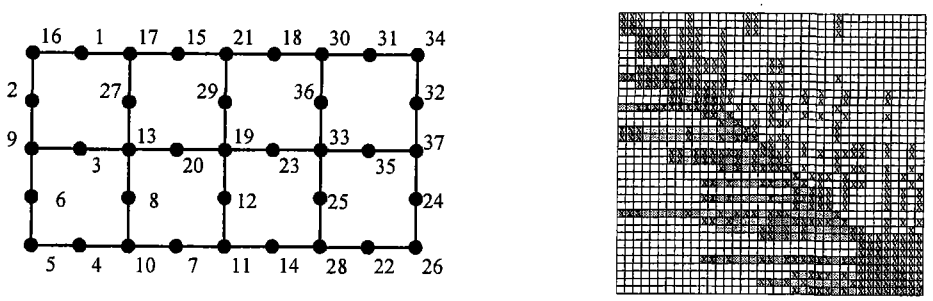


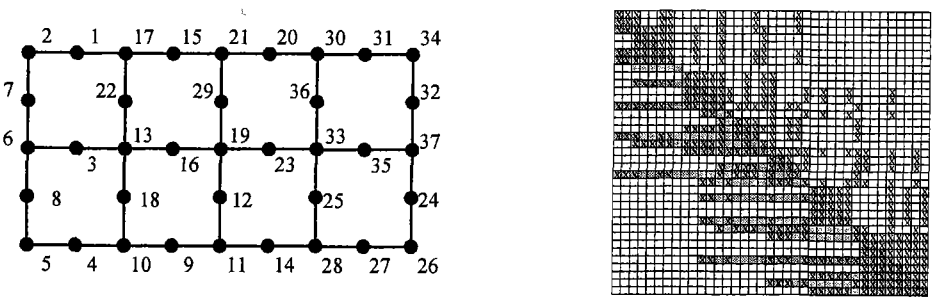
Fig. 4.8 Relationship between optimum and average of storages and generations



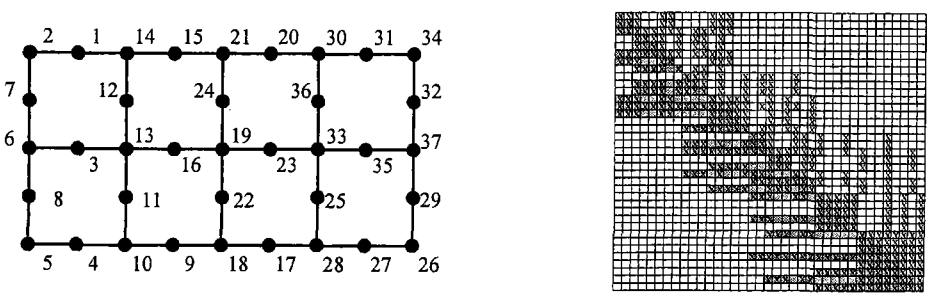
(a) 0^{th} generation, 528 coefficients



(b) 100^{th} generation, 334 coefficients

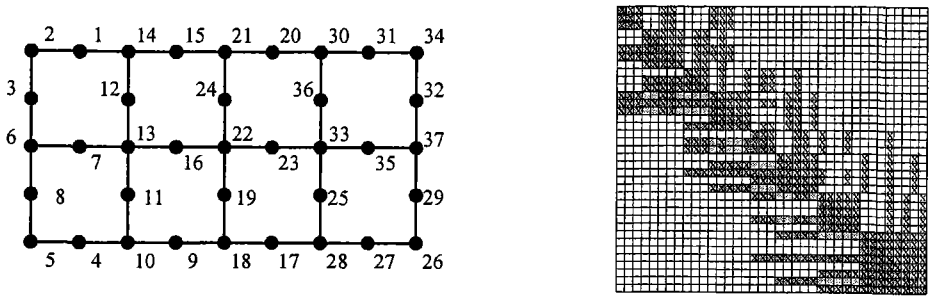


200^{th} generation, 331 coefficients

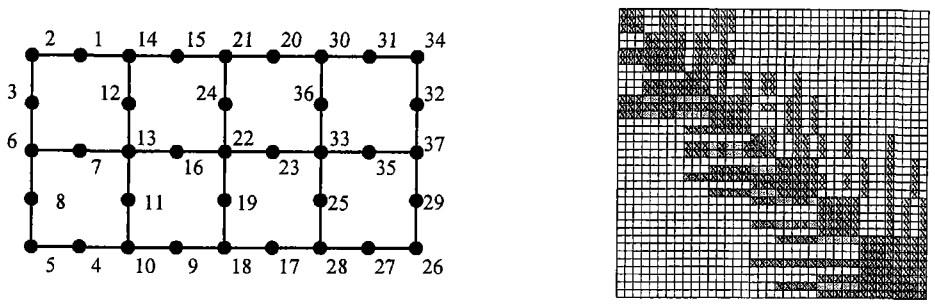


(d) 500^{th} generation, 279 coefficients

Fig. 4.10 The optimum solution in various generations



(e) 1500th generation, 271 coefficients



(f) 3000th generation, 271 coefficients

Fig. 4.10 The optimum solution in various generations (continued)

According to the relationship between average of storage and number of generation, the average of storage declines dramatically at the beginning and slightly decreases in the later generations. The optimum solution can be obtained in much later generations by following this manner.

Fig. 4.10 shows the node numbering and its storage on the optimal system stiffness matrix up to the recent generations. Every generation, the current optimum solution of matrix storage will be replaced by the best recent optimum solution. According to Fig. 4.10, the optimum storage pattern of system stiffness matrix in the first generation (Fig. 4.10a) developed dramatically to a less matrix storage within 500 generations (Fig. 4.10b and c). Since there are plenty of voids in the matrix in the beginning of generations, the stiffness coefficients can be condensed tremendously. However, as each generation processes, matrix is denser, resulting in the difficulty of optimizing the matrix storage. Thereafter, the average of storage slightly decreases and converges to a certain value in later generations (Fig. 4.10d-f). It is observed that the solution is found on the 1500th generation but there is a chance to obtain a better solution in the further generations. Therefore, the process is extended for a while to guarantee that there will be no better solution. Finally, after the process is terminated in 5000th generation, the obtained answer is same as in the 1500th and 3000th generation (Fig. 4.10f). Consequently, it is observed that the stiffness coefficients move closer to the diagonal of the matrix and eliminate most of the voids inside. The resources of computer will be utilized in the more efficient way as well as the faster computational time.

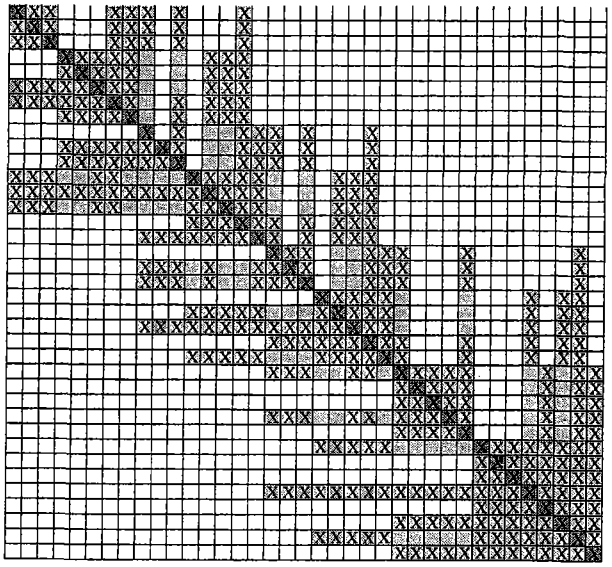


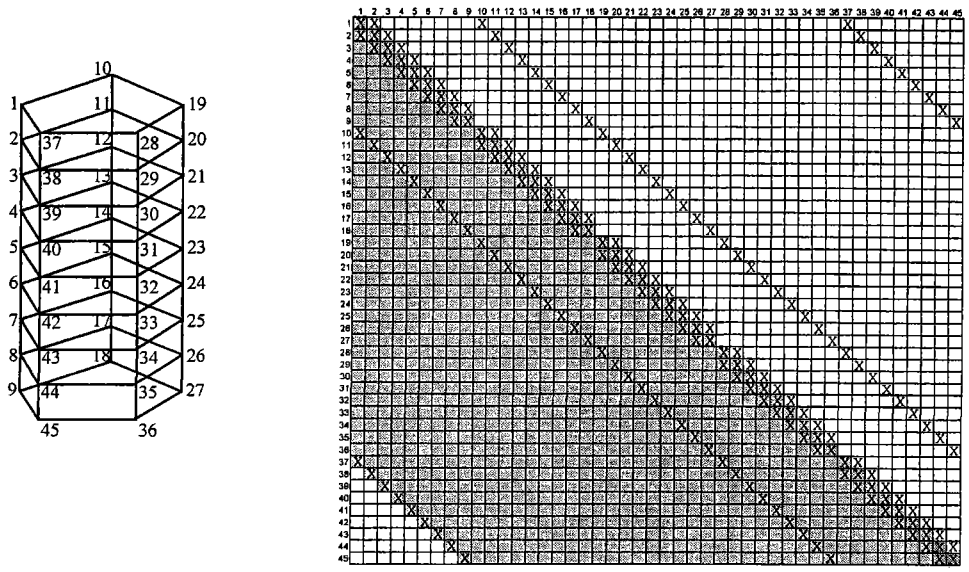
Fig. 4.11 Final system stiffness matrix comparing the genetic algorithm with graph theory

Fig. 4.11 illustrates the system stiffness matrix of obtained node numbering (lower triangular) comparing with the result from the profiles reduction algorithm based on the graph theory (Gajewski and Lompies 1996) (upper triangular). Their labeling scheme comprises two distinct steps: The selection of pseudo-peripheral nodes which make a good starting point for band width as well as for profile and wavefront reduction algorithms and then the algorithm which labels the nodes. The basic idea behind the algorithms is that during each stage of the labeling process nodes with small current degrees and long distances from the end node are labeled first. Their answer has storage of 270 stiffness coefficients but the obtained solution has 271 stiffness coefficients. Indeed, the genetic algorithms are based on the probabilistic methods. Many processes involve with the randomization, e.g., reproduction, crossover and mutation. The optimum solution of the large and complicated problems may not be obtained. Although, the answer from the wavefront reduction algorithm is better, the obtained answer from genetic algorithm is also satisfied.

4.2.2 Space truss using two-noded line elements

The second problem is a mesh of space truss. This mesh has been taken from problem 2 in Collins (1973) and involves two-noded line elements only. It consists of 85 two-noded line elements, 45 nodes, the search space is 1.196×10^{56} node numbering patterns, and named mesh B. The mesh topology and the original numbering are shown in Fig. 4.12a and the structural stiffness matrix is shown in Fig. 4.12b. Its original storage is 620 stiffness coefficients and the maximum storage is 703 stiffness coefficients. The genetic algorithm's parameters are same as a previous problem in which maximum generation=5,000, population size=100, probability of crossover p_c=0.8 and probability of mutation p_m=0.05. For the number of chromosome, since there are 45 nodes in this problem, the number of chromosome must be 45.

Fig. 4.13 illustrates the relationship between the solutions and the average of storages of system stiffness matrix and the generations. As in previous problem, both



(a) Mesh B (b) Original stiffness matrix

Fig. 4.12 Mesh topology and original system stiffness matrix of mesh B

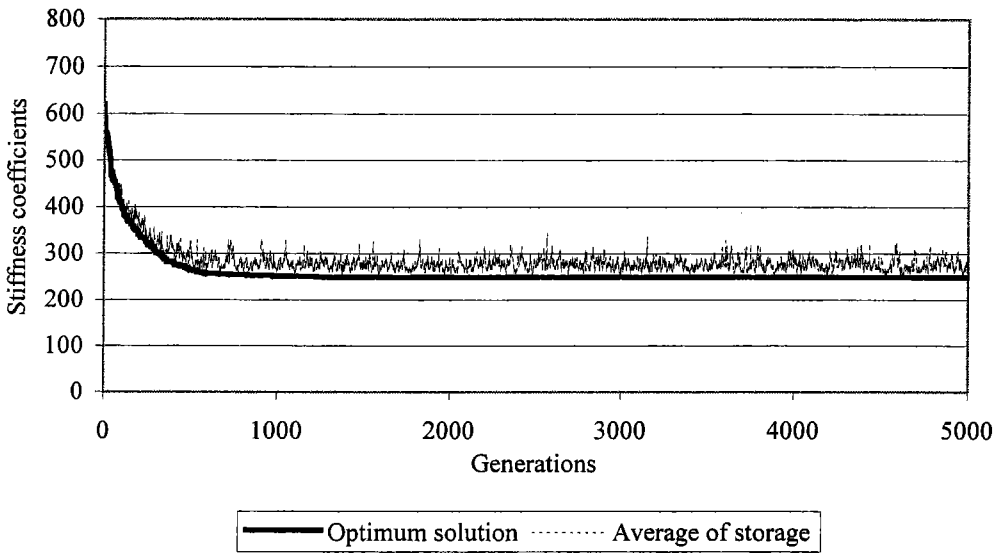


Fig. 4.13 Relationship between optimum and average of storages and generations

the solutions and the averages decline tremendously at the beginning after that they seem to converge within the 1500th generation. The final solution is obtained in 1226th generation in which has 249 stiffness coefficients.

Fig. 4.14 illustrates the system stiffness matrix of obtained node numbering pattern (lower triangular) comparing with the result from the bandwidth reduction (Collins 1973) (upper triangular). His labeling algorithm reduces the bandwidth of matrix by exchanging the rows and columns of matrix. The origin of new numbering system is located at each node in turn so that the number of different renumbering schemes generated (or attempted) is equal to the number of nodes. The new

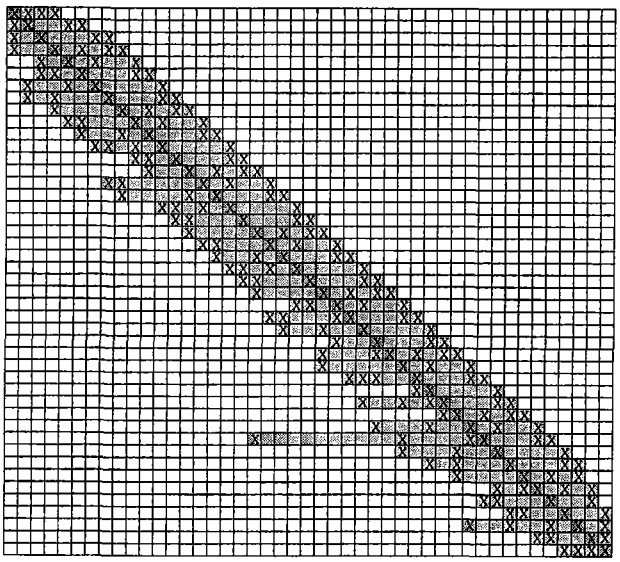


Fig. 4.14 Final system stiffness matrix comparing the genetic algorithm with band width reduction

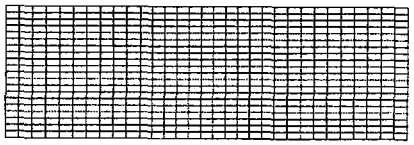
numbering is examined for the bandwidth. If it is grater than the bandwidth produced by a previous numbering scheme, the current scheme is abandoned and new one is started. In this problem, his answer was obtained as 249 stiffness coefficients, which is equal to the answer from our node renumbering algorithm. However, the storage patterns of both answers are difference. Our obtained answer is more scatter. Finally, both numbering patterns can be considered as the optimum answer of this problem.

4.2.3 Meshes using four-noded quadrilateral elements

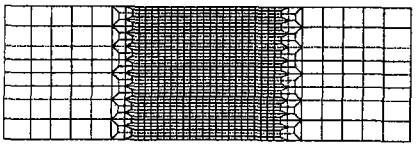
This example is the finite element meshes used in an analysis of cracking localization of the quasi-brittle materials in four-point bending test as shown in Fig. 4.15a-d. These meshes are varying the degree of difficulty of the optimization with 672, 1140, 2288 and 3826 nodes. Their search spaces are $9.1\text{x}10^{1609}$, $5.5\text{x}10^{2901}$, $7.1\text{x}10^{6694}$ and $1.5\text{x}10^{12048}$ node numbering patterns, respectively. All of them are constructed by using only four-noded quadrilateral elements. With these meshes, the performance of the node renumbering algorithm has been figured out by comparing the original storage of the system stiffness matrix with the optimized ones as well as the computational time of the solver.

For a consistent comparison of the computational time, a solver was executed to solve the displacements in both original and optimized node numberings on the same computer (Pentium 200MMX, RAM 64 MB). The computational times were captured from the solving of displacements only because we want to eliminate the memory allocation time when assemble the element stiffness coefficients to the system stiffness matrix.

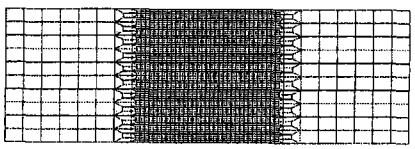
Since the problem has various sizes of the search space, the genetic algorithm's parameters are difference in each mesh as shown in Table 4.1. These parameters are set by the degree of difficulty of the problems accordingly. It is obvious that the largest search space is the most difficult to search the solution. In this case, a population size of such mesh is set to be larger than the other meshes to increase the



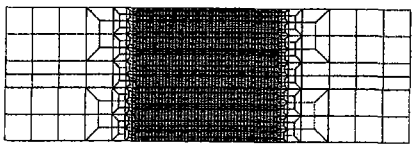
Mesh C1 (672 nodes, 620 elements)
(a)



Mesh C2 (1140 nodes, 1096 elements)
(b)



Mesh C3 (2288 nodes, 2232 elements)
(c)



Mesh C4 (3826 nodes, 3764 elements)
(d)

Fig. 4.15 Finite element meshes for the analysis of cracking localization in four-point bending test

Table 4.1 Genetic algorithm's parameters for the meshes using four-noded quadrilateral elements problem

Item	Value			
	Mesh C1	Mesh C2	Mesh C3	Mesh C4
Maximum number of generation	10,000	10,000	10,000	10,000
Population size	100	100	100	300
Crossover probability	0.85	0.85	0.85	0.85
Mutation probability	0.05	0.05	0.05	0.1

Table 4.2 The answer of optimization of four meshes

Mesh	Original numbering		Optimized numbering		Percent reduction	
	Storage (coefficients)	Computing time (sec.)	Storage (coefficients)	Computing time (sec.)	Storage (%)	Time (%)
C1	24,831	0.187	14,962	0.142	39.7%	24.1%
C2	63,561	0.542	47,020	0.472	26.0%	12.9%
C3	186,878	2.117	144,891	1.885	22.5%	11.0%
C4	370,249	13.449	309,950	11.860	16.3%	11.8%

searching area in the search space or, in other words, to examine more numbering patterns. Moreover, the higher probability of mutation increases the chance of discovering the new node numbering patterns.

Unfortunately, these optimization problems have extremely large search space. The randomization of population in the first generation might not yield the answers that better than the original ones. Therefore, the algorithm needs some modifications. Two of randomized node numbering patterns are replaced by the fixed ones. That is the original node numbering patterns (1 to N) and the inverse of their patterns (N to 1). By following the ordinary genetic algorithms, the solutions are guaranteed that they are always better than the original ones. After performing the node renumbering

of these meshes, the result is shown in a Table 4.2. Once again, the robustness of the proposed algorithm is confirmed. The matrix storage of the systems and the computational times of all meshes are significant reduced.

From the results, the averages of storage in all meshes suddenly decrease to their original storage within a few generations. This is because the storages of fixed node numbering patterns are much less than the randomized patterns. Most of the randomized patterns are eliminated by the reproduction operator. Thereafter, all meshes seem to be converged within $10{,}000^{th}$ generation (see Fig. A.1-4). It means that 10,000 generations are enough for these problems. In summary, the obtained answers are satisfied.

4.3 Conclusions

From the obtained solution of all meshes in above three problems, it can be concluded that the small meshes are easier to optimize than the larger meshes because the search space of the larger meshes are much larger than the smaller meshes. For example, the search space of mesh C4 has 1.45×10^{12048} node numbering patterns while the mesh A has only 1.196×10^{56} patterns. Indeed, with this number, it is almost impossible to obtain the minimum storage of mesh C4. Considering the complexity of the problems, the element type that consists of more nodes causes more complexity in problems because one degree of freedom is related to more other degrees of freedom. With this fact, although the search space of the mesh A is smaller than mesh B, the mesh A obviously cannot be obtained the minimum matrix storage that is easily obtained in mesh B.

The literatures of meshes A and B were the optimization of storage of the matrix obtained from the band width reduction method by using graph theory. This implies that the obtained results are the minimum storage of the meshes. Nevertheless, the results from the node renumbering by using genetic algorithm are close to results from the literatures. Consequently, The advantage of genetic algorithm is that it is quite easy to understand than the graph theory. Moreover, the genetic algorithm is very flexible algorithm. It can be applied to many problems by using same procedure.