ACADEMIC PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Sound and Vibration 262 (2003) 141–160

JOURNAL OF SOUND AND VIBRATION

www.elsevier.com/locate/jsvi

Crack identification using evolutionary algorithms in parallel computing environment

Mun-Bo Shim*, Myung-Won Suh

School of Mechanical Engineering, SungKyunKwan University, Jangan-Gu, Suwon, Kyoungki-Do, 440-746, South Korea

Received 20 November 2001; accepted 17 June 2002

Abstract

It is well known that a crack has an important effect on the dynamic behavior of a structure. This effect depends mainly on the location and depth of the crack. To identify the location and depth of a crack in a structure, classical optimization technique was adopted by previous researchers. That technique overcame the difficulty of finding the intersection point of the superposed contours that correspond to the eigenfrequency caused by the crack presence. However, it is hard to select the trial solution initially for optimization because the defined objective function has heavily local minima. A method is presented in this paper which uses a continuous evolutionary algorithm (CEA), which is suitable for solving inverse problems and implemented on PC clusters to shorten calculation time. With finite element model of the structure to calculate eigenfrequencies, it is possible to formulate the inverse problem in optimization format. CEAs are used to identify the crack location and depth minimizing the difference from the measured frequencies. We have tried this new idea on beam structures and the results are promising with high parallel efficiency over about 91%.

1. Introduction

Techniques to detect cracks and defects hidden in structure and to evaluate their residual life span are very important to assure the structural integrity of operating plants and structures. Many researchers have investigated the potential of system identification to determine the properties of a structure. A state of damage could be detected by a reduction in stiffness. A crack, which occurs in a structural element, causes some local variations in its stiffness which affects the dynamics of the whole structure to a considerable degree. An analysis of the changes is tried to identify the crack.

*Corresponding author. Tel.: +82-31-290-7502; fax: +82-31-290-5276.
E-mail addresses: shimmb@nature.skku.ac.kr (M.-B. Shim), suhmw@yurim.skku.ac.kr (M.-W. Suh).

Most of the studies on crack identification problem have adopted the modal parameter or the dynamic response to identify the global stiffness and mass matrices of a structure.

A crack in a structure introduces a local flexibility, which is a function of the crack depth. This flexibility changes the stiffness and the dynamic behavior of the structure. Chondros and Dimarogonas [1,2] considered the crack as a local elasticity, which effects the elasticity of the whole cracked structure under consideration and related the crack depth with the frequency decrease. Gounaris and Dimarogonas [3] have constructed a special cracked beam finite element and Papadopoulos and Dimarogonas [4] used a 6×6 compliance matrix, including off-diagonal terms, to simulate a cracked shaft and to study its dynamic behavior.

A number of papers deal with the problem of crack location and size identification in order to propose new, efficient and more precise methods. Inagai et al. [5] used a procedure with eigenfrequency measurements to find the crack size and location. Leung [6] and Anifantis et al. [7] proposed crack identification methods through measurements of the dynamic behavior in bending. Dimarogonas and Massouros [8] investigated the dynamic behavior of a circumferentially cracked shaft in torsion and proposed nomographs for finding the crack depth and location. Nikolakopoulos et al. [9] presented the dependency of the structural eigenfrequencies on crack depth and location in contour graph form. To identify the location and depth of a crack, they determined the intersection points of the superposed contours that correspond to the measured eigenfrequency variations caused by the crack presence. However, the intersecting points of the superposed contours are not only difficult to find but also incorrect to evaluate since the procedure mainly depends on men's eye. Suh et al. [10] adopted classical optimization technique to overcome the difficulty of finding the intersection point of the superposed contours that correspond to the eigenfrequency caused by the crack presence. However, it is also hard to select the trial solution initially for optimization because the defined objective function has heavily local minima.

To identify the location and depth of a crack in a structure with only eigenfrequency information efficiently, a method is presented in this paper which uses a continuous evolutionary algorithm (CEA) [11], which is implemented on PC clusters to shorten calculation time. With finite element model of a structure to calculate eigenfrequencies, it is possible to formulate the inverse problem in optimization format. CEAs, which are efficient in real parameter identification problem and are suitable for solving inverse problems, are used to identify the crack location and depth minimizing the difference from the measured frequencies.

2. Inverse analysis method

The inverse analysis is generally defined as identifying the parameter set $x^* \in X$ when measured, or reference data $y^* \in Y$ and direct mapping $\psi : X \to Y$ are known. Problems with the non-linear direct mapping ψ are termed non-linear inverse problems. In practice, deterministic models describe reality only in an idealized sense, and thus we may express the input–output relation as follows:

$$y = \psi(x) + \varepsilon, \tag{1}$$

where $\varepsilon = \varepsilon_1 + \varepsilon_2$, and ε_1 and ε_2 are errors in the measurement of y and those in the model equations, respectively.

In the analysis of field quantities shown in Fig. 1, the model equations in general take the form:

$$L(k)\phi = q, \tag{2}$$

where L, k, ϕ, q are the differential operator, material property, field quantity and a source term, respectively.

Inverse problems for Eq. (2) can be classified in terms of the parameter set to be identified: (a) domain Ω, (b) governing equations, (c) boundary conditions, (d) force or source q applying in Ω, and (e) material properties k defined in Ω and involved in the governing equations [13]. In these problems, the input and output vectors reside in the continuous space. There are two main strategies for solving inverse problems. One is to solve a set of equations and the other is to directly find the minimum or maximum of a certain function. However, the former is worth noting the following difficulty: the inverse problem can always be defined as an abstract theoretical concept. In general, inverse function is a subset of original input, in fact such a subset could even be empty, so that the usual concept of "function" as a "one-to-one" injection breaks down. Generally, it is reasonable to solve the latter. Out of them, minimizing a least-squares criterion has been most widely used for identification.

In this approach, optimization techniques are used to find the input by adjusting them until the measured, or reference data match the corresponding data computed from parameter set in the least-squares fashion, i.e.,

$$\min f(x) \tag{3a}$$

with the cost functional

$$f(x) = \sum_{i=1}^{m} k_i (y_i^* - \Psi_i(x))^2, \tag{3b}$$

where k_i is a weighting factor. Various calculus-based optimization techniques have been intensively used to solve this optimization problem. These techniques can, however, fail if errors contained in the model equations and in the measurement cause the objective function to be complex. In such cases, the solution may result in a local minimum, unless some regularization method is incorporated. The present study uses evolutionay algorithms, which are significantly promising for complex optimization.

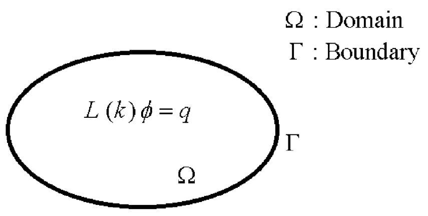

Fig. 1. Problems of field quantities.

3. Structure analysis

In the finite element model of the damaged structure, the effect of the crack on the behavior of the structure can be simulated through the introduction of the transfer matrices which are method for finding the stiffness matrix. A planar frame structure can be modelled using two-dimensional beam elements having 3 d.o.f. $(\delta_x, \delta_y, \theta_z)$ per node, i.e., with extension and bending, in Fig. 2.

The corresponding stiffness and consistent mass local matrices [14] are

$$[K_e] = \frac{EI_{zz}}{L^3}\begin{bmatrix} \beta L^2 & 0 & 0 & -\beta L^2 & 0 & 0 \\ 0 & 12 & 6L & 0 & -12 & 6L \\ 0 & 6L & 4L^2 & 0 & -6L & 2L^2 \\ -\beta L^2 & 0 & 0 & \beta L^2 & 0 & 0 \\ 0 & -12 & -6L & 0 & 12 & -6L \\ 0 & 6L & 2L^2 & 0 & -6L & 4L^2 \end{bmatrix}, \tag{4}$$

$$[M_e] = \frac{\rho AL}{420}\begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22L & 0 & 54 & -13L \\ 0 & 22L & 4L^2 & 0 & 13L & -3L^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13L & 0 & 156 & -22L \\ 0 & -13L & -3L^2 & 0 & -22L & 4L^2 \end{bmatrix}, \tag{5}$$

where $\beta = A/I_{zz}$, L is the length of element e and A is the cross-section area. E and ρ are the modulus of elasticity and mass density, respectively, and I_{zz} is the second moment of inertia about the local z-axis.

From the Euler–Bernoulli theory for the above-mentioned degrees of freedom, the transfer matrix [14] which transfers the state variables (displacement, force) from one node to the other

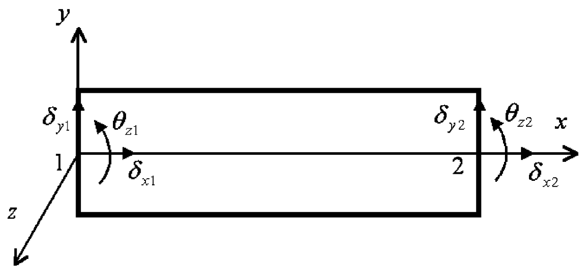

Fig. 2. A beam finite element with extension and bending.

node is

$$[T_e] = \begin{bmatrix} 1 & 0 & 0 & \dfrac{-L}{AE} & 0 & 0 \\ 0 & 1 & L & 0 & \dfrac{L^3}{6EI_{zz}} & -\dfrac{L^2}{6EI_{zz}} \\ 0 & 0 & 1 & 0 & \dfrac{L^2}{2EI_{zz}} & \dfrac{L}{EI_{zz}} \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & L & -1 \end{bmatrix}. \tag{6}$$

A beam finite element of length L_e, containig a crack of depth α at distance L_{1e} from its left end, is depicted in Fig. 3.

The crack introduces a local compliance in the structure. The state vectors at positions $i, C_L, C_R,$ and, j are

$$\{z_i\} = \{\, \delta_{xi} \quad \delta_{yi} \quad \theta_{zi} \quad F_{xi} \quad F_{yi} \quad M_{zi}\,\}^{\mathrm{T}}, \tag{7a}$$

$$\{z_L\} = \{\, \delta_{xL} \quad \delta_{yL} \quad \theta_{zL} \quad F_{xL} \quad F_{yL} \quad M_{zL}\,\}^{\mathrm{T}}, \tag{7b}$$

$$\{z_R\} = \{\, \delta_{xR} \quad \delta_{yR} \quad \theta_{zR} \quad F_{xR} \quad F_{yR} \quad M_{zR}\,\}^{\mathrm{T}}, \tag{7c}$$

$$\{z_j\} = \{\, \delta_{xj} \quad \delta_{yj} \quad \theta_{zj} \quad F_{xj} \quad F_{yj} \quad M_{zj}\,\}^{\mathrm{T}}. \tag{7d}$$

If no force is acting between nodes i and j, then it can be derived from simple beam theory, where the four state vectors are related as follows:

$$\{z_L\} = [T_1]\{z_i\}, \tag{8a}$$

$$\{z_R\} = [T_C]\{z_L\}, \tag{8b}$$

$$\{z_j\} = [T_2]\{z_R\}, \tag{8c}$$

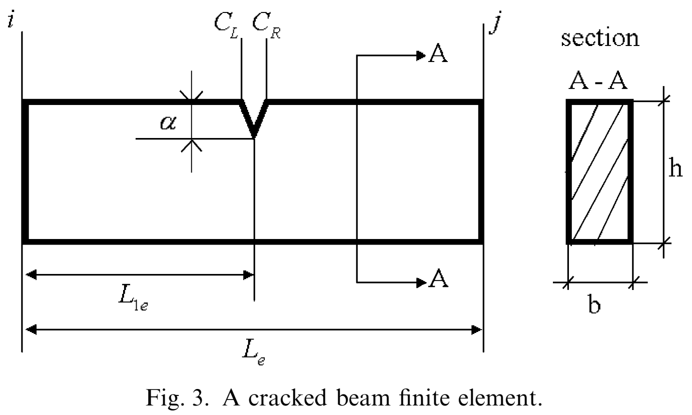

Fig. 3. A cracked beam finite element.

where $[T_1]$ and $[T_2]$ are the transfer matrices of the subelements $C_L - i$ and $C_R - j$, respectively, and $[T_C]$ is the point transfer matrix due to the crack. Matrix $[T_C]$, which relates the state vectors on the left and right of the crack is

$$[T_C] = \begin{bmatrix} 1 & 0 & 0 & c_{11} & 0 & c_{13} \\ 0 & 1 & 0 & 0 & c_{22} & 0 \\ 0 & 0 & 1 & c_{31} & 0 & c_{33} \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{9}$$

where subscripts 1, 2 and 3 correspond to tension, shear and bending, respectively. Terms c_{13} and c_{31}, responsible for the coupling of tension and bending [3], are not considered here, whereas the rest are known as follows [12]:

$$c_{11} = \frac{2\Phi_1(1-v^2)}{Eb}, \quad c_{22} = \frac{2k^2\Phi_3(1-v^2)}{Eb}, \quad c_{33} = \frac{72\Phi_2(1-v^2)}{Ebh^2}, \tag{10a−c}$$

where v is the Poisson ratio, k is a constant which for rectangular cross-sections is known to be 1.5 and Φ_i are functions of the non-dimensional crack depth α/h [13]. These functions, which are presented in Fig. 4, are integrals of the empirical formulas used by Tada [13] for computation of stress intensity factors K_I in single edge notch specimens under pure tension, bending and shear.

From Eq. (8a–c) the following is obtained:

$$\{z_j\} = [T_e^C]\{z_i\}. \tag{11}$$

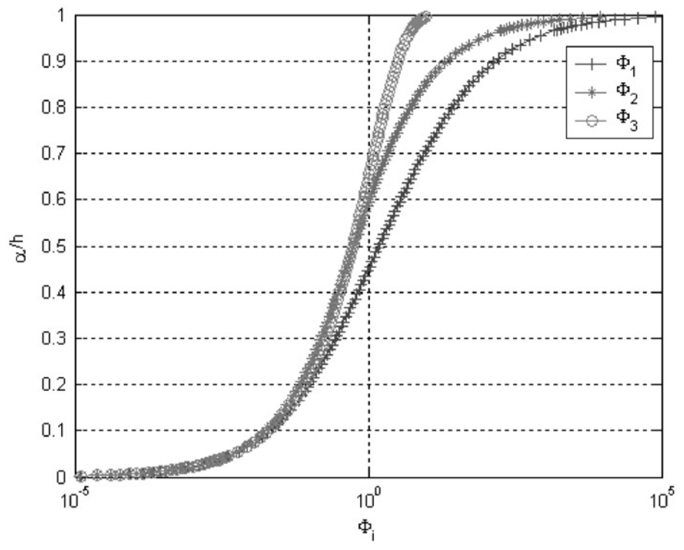

Fig. 4. Φ_i versus α/h for single edge notch specimen under pure tension, bending and shear.

The transfer matrix $[T_e^C]$ of the cracked element is written in the form

$$[T_e^C] = [T_2][T_C][T_1] = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}, \tag{12}$$

where $[A_i]$ are 3×3 submatrices. Eq. (12) leads to the stiffness matrix of the crack element

$$[K_e^C] = \begin{bmatrix} -[A_2]^{-1}[A_1] & [A_2]^{-1} \\ [A_3] - [A_4][A_2]^{-1}[A_1] & [A_4][A_2]^{-1} \end{bmatrix}. \tag{13}$$

The equation of motion in matrix form is known to be

$$(-\omega^2[M] + [K])\{x\} = \{0\}, \tag{14}$$

where ω is eigenfrequency, x is a displacement vector. The above analysis serves to identify the location and depth of a crack in a frame structure, just by measuring the eigenfrequency variations.

4. Parallel evolutionary algorithms (EAs) for crack identification

The cracked structure in this study is discretized into a set of elements and the crack is assumed to be located within one of the elements. For estimating the location and size of a crack CEAs are utilized, which is suitable for solving inverse problems and implemented on PC clusters to shorten calculation time. With finite element model of the structure to calculate eigenfrequencies, it is possible to formulate the inverse problem in optimization format. CEAs are used to identify the crack location and depth minimizing the difference from the measured frequencies.

4.1. *Evolutionary algorithms*

4.1.1. *Fundamental algorithms*

EAs are probabilistic optimization algorithm based on the model of natural evolution and the algorithm has clearly demonstrated its capability to create good approximate solutions in complex optimization problems. The popularity of the algorithms is due to the following characteristics:

(i) Less possibility to converge to a local minimum as the search starts from a number of points.
(ii) Compatibility with the parallel computer.
(iii) Robustness since only objective function information is required.
(iv) Capability to find a solution in broad search space effectively through probabilistic operations.

Fig. 5 shows the fundamental structure of EAs. First, a population of individuals, each represented by a vector, is initially (generation $\boldsymbol{k} = 0$) generated at random, i.e.,

$$P^k = \{u_1^k, \ldots, u_\lambda^k\} \in (I)^\lambda, \tag{15}$$

where I represents the space of individual and λ is the population size of parental individuals. The population then evolves towards better regions of the search space by means of randomized

```
k = 0;
Initialize P(k)
do
        Evaluate P(k);
        Select P(k);
        Recombine P(k);
        Mutate P(k);
        k = k +1;
while terminal condition is not satisfied
```

Fig. 5. Fundamental structure of EAs.

processes of recombination, mutation and selection though either the recombination or mutation is not implemented in some algorithms. In the recombination, parental individuals breed offspring individuals by combining part of the information from the parental individuals. The mutation then forms new individuals by making large alterations with small possibility to the offspring individuals regardless of their inherent information. With the evaluation of fitness for all the individuals, the selection operator favorably selects individuals of higher fitness to produce more often than those of lower fitness. These reproductive operations form one generation of the evolutionary process, which corresponds to one iteration in the algorithm, and the iteration is repeated until a given terminal criterion is satisfied.

4.1.2. Genetic algorithms (GAs)

In GAs [15], the individual is given by a binary string $\boldsymbol{u}_i^k \in \boldsymbol{I} = \{0,1\}^l$ as if it represents a chromosome of genes in genetics. Recombination of the genetics is conducted by the cross-over. In the case of one-point cross-over, two randomly selected individuals are renewed by two offspring individuals:

$$\begin{aligned} u_\alpha^k &= \{u_{\alpha 1}^k, \dots, u_{\alpha m}^k, u_{\beta(m+1)}^k, \dots, u_{\beta l}^k\}, \\ u_\beta^k &= \{u_{\beta 1}^k, \dots, u_{\beta m}^k, u_{\alpha(m+1)}^k, \dots, u_{\alpha l}^k\}. \end{aligned} \tag{16}$$

Each gene is then mutated with small possibility by

$$u_{ij}^k = 1 - u_{ij}^k. \tag{17}$$

Due to the binary representation, conversion of binary information to continuous search space is necessary

$$c : u_i^k \rightarrow x_i^k. \tag{18}$$

The evaluation of the fitness can be conducted with a linear scaling, where the fitness of each individual is calculated as the worst individual of the population subtracted from its objective function value

$$\Phi(\mathbf{x}_i^k) = \max\{f(\mathbf{x}^k)|\mathbf{x}^k \in \boldsymbol{P}^k\} - \boldsymbol{f}(\mathbf{x}^k) \quad \forall i \in \{1, \dots, \lambda\}. \tag{19}$$

$\Phi(\mathbf{x}_i^k) \geqslant 0$ is thus satisfied by this equation. Proportional selection, which is the most popular selection operation, can be directly used. In this selection, the reproduction probabilities of individuals are given by their relative fitness

$$p_s(\mathbf{x}_i^k) = \frac{\Phi(\mathbf{x}_i^k)}{\sum_{j=1}^{\lambda} \Phi(\mathbf{x}_j^k)} \geqslant 0. \tag{20}$$

Optionally, ranking selection can be implemented in this algorithm. These reproductive operations form one generation of the evolutionary process, which corresponds to one iteration in the algorithm, and the iteration is repeated until a given terminal criterion is satisfied.

4.1.3. Continuous evolutionary algorithms (CEAs)

CEAs [11] are one of EAs, which is specifically formulated for the optimization with continuous search space. The reproductive operations of CEAs are intended to be similar to those of GAs such that it can take the advantage of probabilistic features in GAs. The major difference of CEAs from GAs is that a search point itself, i.e., a real continuous vector ($x_i^k \in I = R^n$), gives the representation of the individual. This formulation was made with an assumption that the direct use of the search point may search more efficiently than the representation decoded into a binary string as used in GAs. This representation makes us grasp the concept of the individual not as genetic information but phenomenological information.

The definition of the recombination and mutation becomes the probabilistic distribution of the phenomenological measures accordingly. The recombination operation is therefore defined as [18,19]

$$\begin{aligned} \mathbf{x}_\alpha^{k+1} &= (1-\mu_\alpha^k)\mathbf{x}_\alpha^k + \mu_\beta^k \mathbf{x}_\beta^k, \\ \mathbf{x}_\beta^{k+1} &= \mu_\alpha^k \mathbf{x}_\alpha^k + (1-\mu_\beta^k)\mathbf{x}_\beta^k, \end{aligned} \tag{21}$$

where x_α^k and x_β^k are parental individuals at generation k and parameter $\mu_i^k, \forall i \in \{\alpha, \beta\}$ may be defined by the normal distribution with mean 0 and standard deviation σ

$$\mu_i^k = N(0, \sigma^2). \tag{22}$$

The standard deviation can adopt a self-adaptive strategy (variable with respect to k) or be simply constant. The self-adaptive strategy makes the convergence rate required for each generation faster at the expense of the computation time and *vice versa*.

In many studies [16,17], the same μ is used regardless of each parental individual, i.e., the symmetric recombination. Symmetric distribution sometimes leads to good convergence for just unimodal, simple problem, while asymmetric one improves robustness of the algorithm in multiobjective optimization.

Fig. 6 illustrates the difference of the recombination methods, respectively. The parental point is marked with 'o' and the offspring point that is possible is marked with '*'. The cross-over operator in GAs uses variables coded using binary strings of size 7, respectively, and the cross-over point is changed from 1 to 20 bit. The symmetric and asymmetric recombination uses the normal distribution and the offspring point is selected randomly. From the above figure, we find the fact that asymmetric one is more effective because the offspring point is born suitably inside and outside the parental point, that is, both local search and global search are possible.

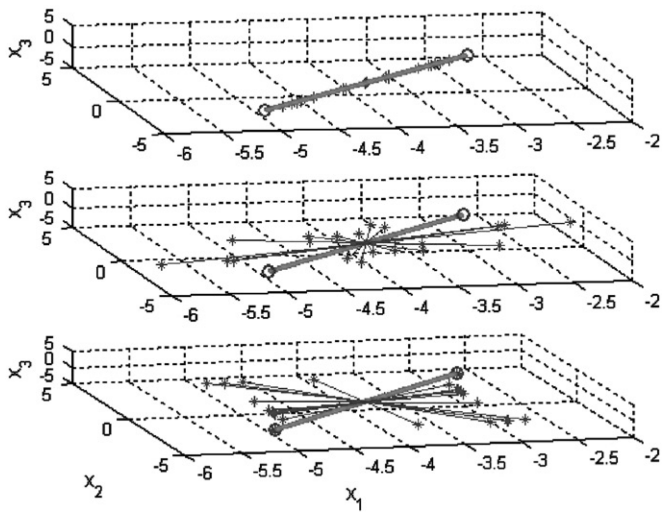

Fig. 6. Comparison of symmetric, asymmetric and genetic cross-over.

The mutation can also be achieved simply by implementing

$$\mathbf{x}^{k+1} = rand(\mathbf{x}_{min}, \mathbf{x}_{max}). \tag{23}$$

Note that the mutation may not be necessary since it can allow individuals to alter largely with small possibility, when the coefficient μ_i^k is large.

The same evaluation of the fitness and selection as GAs can be conducted.

4.1.4. Comparison of GAs with CEAs

$$\begin{array}{ll}
\text{Func. I} & f_1(x) = \sum_{i=1}^{n} x_i^2, x \in R^n; n = 30, \\
& -5.12 \leqslant x_i \leqslant 5.12, x^* = [0, \ldots, 0]^{\mathrm{T}},\ f_1(x^*) = 0 \\
\text{Func. II} & f_2(x) = 6n + \sum_{i=1}^{n} [x_i], x \in R^n; n = 5, \\
& -5.12 \leqslant x_i \leqslant 5.12, x^* = [-5.12, \ldots, -5]^{\mathrm{T}},\ f_2(x^*) = 0 \\
\text{Func. III} & f_3(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i), x \in R^n; n = 20, \\
& -5.12 \leqslant x_i \leqslant 5.12, x^* = [0, \ldots, 0]^{\mathrm{T}},\ f_3(x^*) = 0, \\
\text{Func. IV} & f_4(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos\left(\pi x_i/2\right), x \in R^n; n = 20, \\
& -5.12 \leqslant x_i \leqslant 5.12, x^* = [0, \ldots, 0]^{\mathrm{T}},\ f_3(x^*) = 0.
\end{array} \tag{24}$$

As it is impossible to predict the behavior of the algorithms by theoretical considerations, a set of test functions having continuous search space were prepared to demonstrate the capability of both the Simple GA (SGA) and CEA. The mathematical characteristics of the test functions are unimodal/multimodal, quadratic/non-quadratic, convex/non-convex and continuous/discontinuous. The test functions are as Eq. (24). Three-dimensional graphical representations of these functions are shown in Fig. 7.

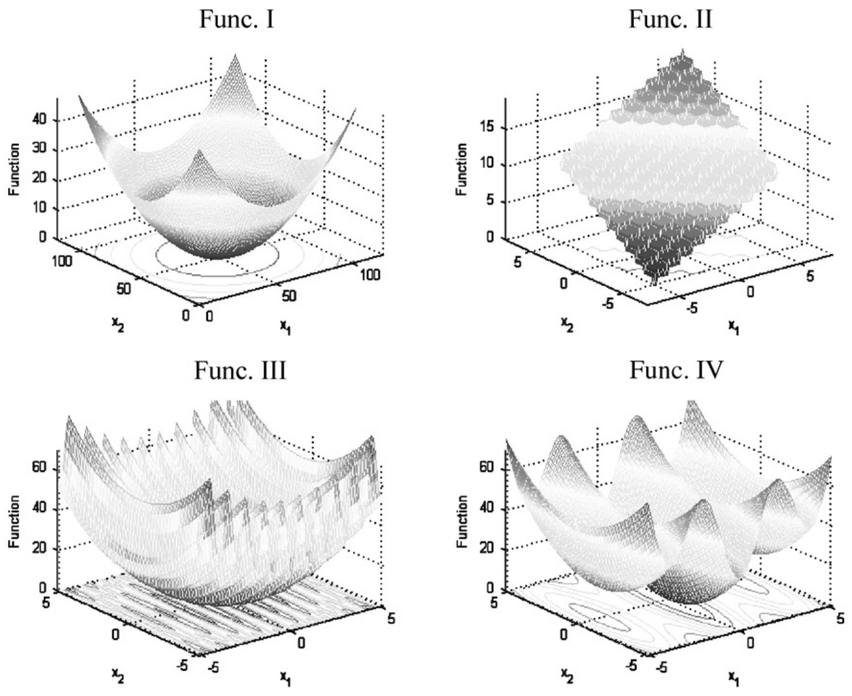

Fig. 7. 3-D representation of the test functions.

Table 1
Internal parameters for both the algorithms

	SGAs	CEAs
Population size	50	50
Bit length	7 per variable	—
Crossover rate	0.6	—
Mutation rate	0.001	0.001
Standard deviation	—	0.5

Internal parameters selected for both the algorithms are listed in Table 1. The cross-over and mutation rates in SGAs, 0.6 and 0.001, are typically used in many publications. The standard deviation of CEAs is set to be constant 0.5. The iteration is repeated until the number of generation reaches 250 or the difference of average objective function and minimum objective function is less than 0.0001.

The results of the search performance of both the algorithms are shown in Fig. 8. In Fig. 8, real lines indicate the outcome from CEAs, whereas the results by SGAs are indicated by the broken lines. The result of the first test clearly reflects the superiority of CEAs on the unimodal function optimization. The second result then indicates that CEAs is almost similar to SGAs when the

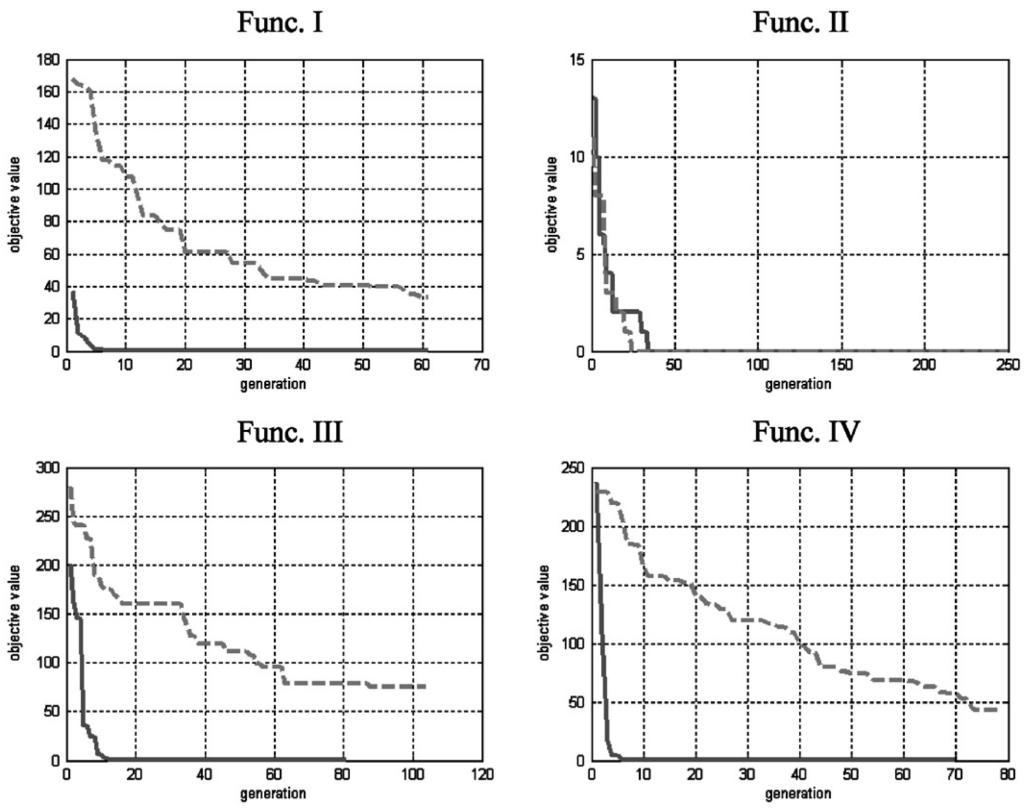

Fig. 8. Comparison of generation history of CEAs with that of SGAs.

function is partially discontinuous. CEAs have a better performance of Funcs. III and IV than SGAs. SGAs are nearly in the state of premature convergences, having a slow convergence rate before reaching the global optimum, for the tests.

4.2. *Parallel EAs*

While EAs have proven effective for global optimization, this capability is purchased at a high computation cost. The basic motivation behind many studies of parallel EAs was to reduce the processing time needed to reach an acceptable solution. This was accomplished implementing EAs on different parallel architectures.

It is recognized that there are different ways to parallel EAs. The first approach in parallelizing EAs is to do a global parallelization [20,21]. In this class of parallel EAs, the evaluation of individuals and the application of evolutionary operators are explicitly parallelized. Every individual has a chance to mate with all the rest. Therefore, the semantics of the operators remain unchanged. This method is relatively easy to implement and a speed-up proportional to the number of processors can be expected.

The second approach in parallelizing EAs uses coarse-grained parallelism [22–24]. The population is divided into a few subpopulations keeping them relatively isolated from each other.

This model of parallelization introduces a migration operator that is used to send some individuals from one subpopulation to another. Two models for population structures are used in different implementations of coarse-grained EAs: the island model and the stepping stone model. The population in the island model is partitioned into small subpopulations by geographic isolation and individuals can migrate to any other subpopulation. In the stepping model, the population is partitioned in the same way, but migration is restricted to neighboring subpopulations. Sometimes coarse-grained parallel EAs are known as distributed EAs since they are usually implemented in distributed memory MIMD computers.

The third approach in parallelizing EAs uses fine-grained parallelism [25,26]. Fine-grained parallel EAs partition the population into a large number of very small subpopulations. The ideal case is to have just one individual for every processing element available. This model calls for massively parallel computers.

In this study, global parallelization is used because the evaluation is the most time consuming and it is easy to implement and a speed-up proportional to the number of processors can be expected. In global parallelization, the evaluation can be parallelized assigning a subset of individuals to each of the processors available. There is no communication between the processors during the evaluation because the fitness of each individual is independent from all the others. Communication only occurs at the start and at the end of the evaluation phase. The evaluation process of the parallel computation system, which is based on PC-based cluster and is organized following the master–worker paradigm, is shown in Fig. 9. On a parallel implementation based on the master–worker paradigm, the master computes the evolutionary operations and the slaves compute the object function values. Therefore, the amount of communication is rather small. Hence, a parallel implementation based on the master–worker prototype can be efficient. The specification of the adopted PC cluster systems is summarized in Table 2.

4.3. Parallel EAs for crack identification

For the crack identification, it is known that crack parameters such as the location and depth of a crack can be determined from the measured eigenfrequencies of structure. This can be classified

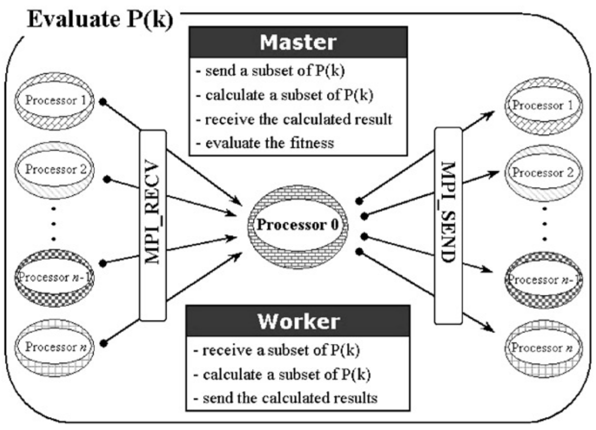

Fig. 9. Implementation of evaluation process on PC-based cluster.

Table 2
Cluster systems

	Intel cluster	Alpha cluster
CPU	Pentium III (700 MHz) × 16	DEC Alpha (533 MHz) × 24
Memory	256 MB/PE	768 MB/PE
OS	Linux2.2.14	Linux2.2.16
Network	FastEthernet TCP/IP	FastEthernet TCP/IP
Communication library	MPICH-1.2.1	MPICH-1.2.1

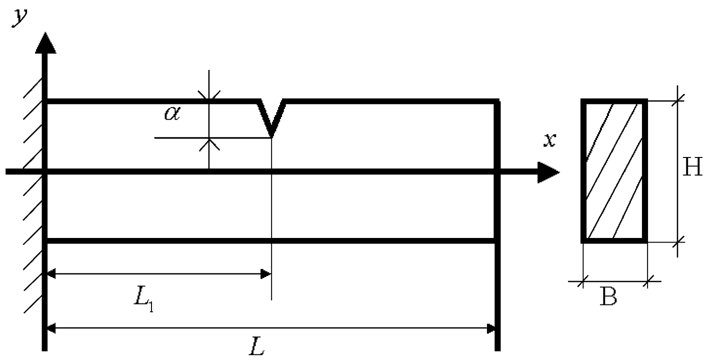

Fig. 10. Model of the cracked clamped-free beam.

as the inverse problem. In this study, parallel evolutionary technique is adopted to identify the crack parameters in a structure.

The parameters which identify the crack are estimated by parallel CEAs described in Sections 4.1–4.2 using the computational structure analysis which is presented in Section 3. Crack identification problem can be constructed in terms of optimization with CEAs. The optimization problem to be formulated is defined as follows:

$$\min_{\alpha, L_1} F(\alpha, L_1) = \sum_{i=1}^{3} w_i (f_i - f_i^*)^2,$$
$$\alpha^{lower} \leqslant \alpha \leqslant \alpha^{upper}, L_1^{lower} \leqslant L_1 \leqslant L_1^{upper}, \quad (25)$$

where α, L_1 is the depth and location of a crack, w_i is a weighting factor and f_i is the eigenfrequency, which is functions of α and L_1, and f_i^* is the measured or reference eigenfrequency.

5. Numerical simulation

5.1. Example problem 1: clamped-free beam

The clamped-free beam of Fig. 10 has a length of $L = 3$ m, rectangular cross-section $B \times H = 0.2$ m $\times$ 0.2 m and contains a crack of depth α at a distance L_1 from the clamped end. The material properties are $E = 2.07 \times 10^{11}\,\mathrm{Nm}^{-2}$, $\nu = 0.3$, and $\rho = 7700\,\mathrm{kg\,m}^{-3}$. The beam is discretized into 12 two-node finite elements.

CEAs for this study is set-up: population size, $N = 100$; standard deviation, $\sigma = 0.5$; mutation rate, $M = 0.001$. Also, proportional selection method is adopted for the selection process.

Two cases are considered. (a) A crack of depth α of 0.03 m exists at L_1 of 1.1 m. The first three eigenfrequencies are obtained computationally based on the theory described in Section 3: $f_1^* = 115.78\,\mathrm{rad/s}$, $f_2^* = 725.60\,\mathrm{rad/s}$, $f_3^* = 2027.36\,\mathrm{rad/s}$. (b) A crack of depth α of 0.07 m exists at L_1 of 2.4 m. The first three eigenfrequencies are obtained computationally based on the theory described in Section 3: $f_1^* = 116.60\,\mathrm{rad/s}$, $f_2^* = 717.94\,\mathrm{rad/s}$, $f_3^* = 1901.00\,\mathrm{rad/s}$.

Parallel CEAs have been applied to this example problem. The performance of the parallel CEA is measured by both the speed-up ratio and the parallel efficiency. The speed-up ratio S_n and the parallel efficiency E_n are defined as follows:

$$S_n = \frac{T_1}{T_n},$$
$$E_n = \frac{S_n}{n} = \frac{T_1}{nT_n}, \tag{26}$$

where T_1, T_n are total times for solving the problem using one processor and n processors. Fig. 11 shows the results obtained by both Intel cluster and Alpha cluster.

Parallel CEAs have a near-linear speed-up with both Intel cluster and Alpha cluster. The parallel efficiency is about 91.6% in Intel cluster and about 93.6% in Alpha cluster. This is due to the fact that the communication cost between processes is relatively small compared with the computation cost.

Figs. 12 and 13 illustrate convergence history of the objective function for case (a) and case (b), respectively. We can know that the result of crack identification is acceptable irrespective of the

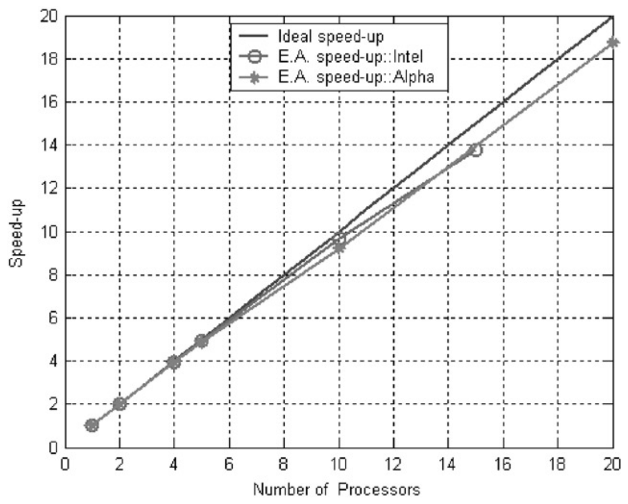

Fig. 11. Speed-up of the parallel algorithm.

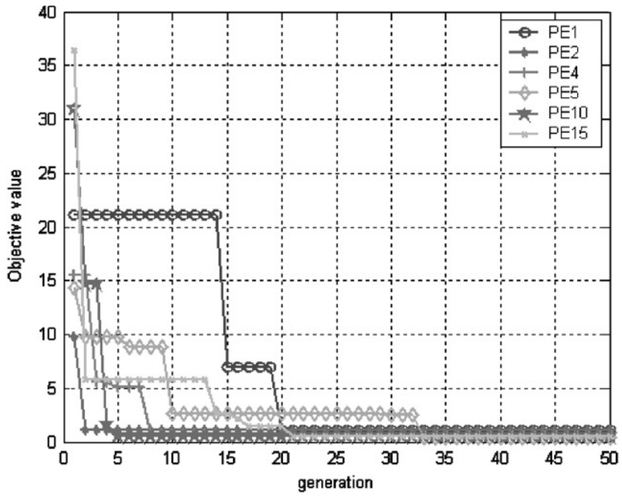

Fig. 12. Generation history in Case (a).

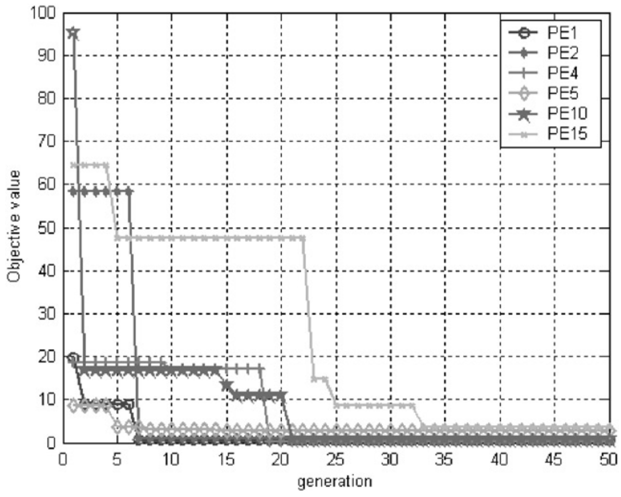

Fig. 13. Generation history in Case (b).

Table 3
Crack identification result: clamped-free beam

	Case (a)			Case (b)		
	Reference value	Result value	Relative error (%)	Reference value	Result value	Relative error (%)
α	0.03	0.029	3.3	0.07	0.068	2.9
L_1	1.1	1.09	0.9	2.4	2.38	0.8
f_1	115.78	115.981	0.2	116.60	116.866	0.2
f_2	725.60	726.258	0.1	717.94	717.333	0.1
f_3	2027.36	2027.489	0.01	1901.00	1901.631	0.03

number of processor elements and an initial population of individual. The result of Table 3 shows that the location and depth of a crack are estimated by parallel CEAs within 3.3% error. Also, the corresponding eigenfrequencies are very close to the reference values within 0.2% error.

5.2. Example problem 2: clamped–clamped plane frame

A crack in the clamped–clamped plane frame, shown in Fig. 14 is identified using the methodology of this study. The frame has the following basic dimensions; $b \times h = 0.16\,\text{m} \times 0.2\,\text{m}$ and $B \times H = 0.008\,\text{m} \times 0.016\,\text{m}$. The frame is discretized into 28 two-node finite elements. Since there is a vertical axis of symmetry, the crack location for the half of the frame is considered.

The application of parallel CEAs is set-up: population size, $N = 100$; standard deviation, $\sigma = 0.5$; mutation rate, $M = 0.001$; proportional selection method.

Two different cases are considered. (a) A crack of depth α of 0.003 m exists at L_1 of 0.15 m. The first three eigenfrequencies are obtained computationally based on the theory described in Section 3: $f_1^* = 267.31$ Hz, $f_2^* = 1150.46$ Hz, $f_3^* = 1798.93$ Hz. (b) A crack of depth α of 0.006 m exists at L_1 of 0.24 m. The first three eigenfrequencies are obtained computationally based on the theory described in Section 3: $f_1^* = 266.96$ Hz, $\mathrm{f}_2^* = 1151.18$ Hz, $f_3^* = 1828.79$ Hz.

Fig. 15 shows the speed-up obtained by both Intel cluster and Alpha cluster. The parallel efficiency is about 94.4% in Intel cluster and about 96.0% in Alpha cluster.

Figs. 16 and 17 illustrate convergence history of the objective function for case (a) and case (b), respectively. The result of Table 4 shows that the location and depth of a crack are estimated by parallel CEAs within 3.1% error. Also, the corresponding eigenfrequencies are very close to the reference values within 0.001% error.

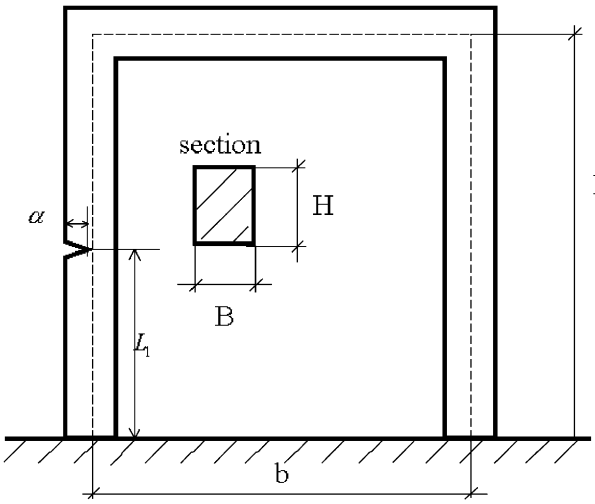

Fig. 14. Clamped–clamped plane frame.

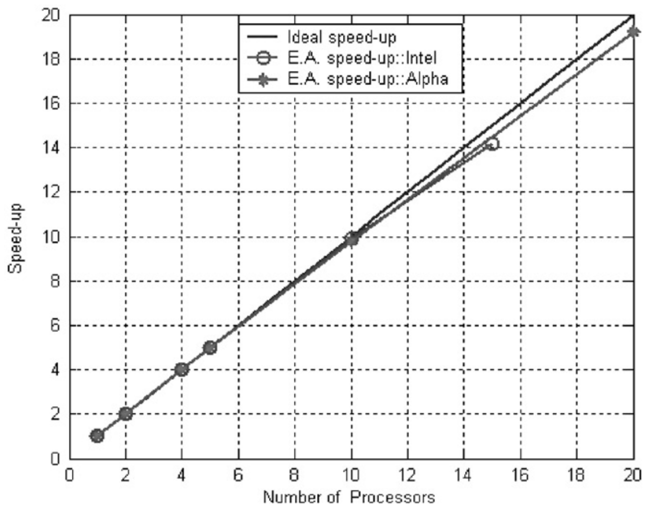

Fig. 15. Speed-up of the parallel algorithm.

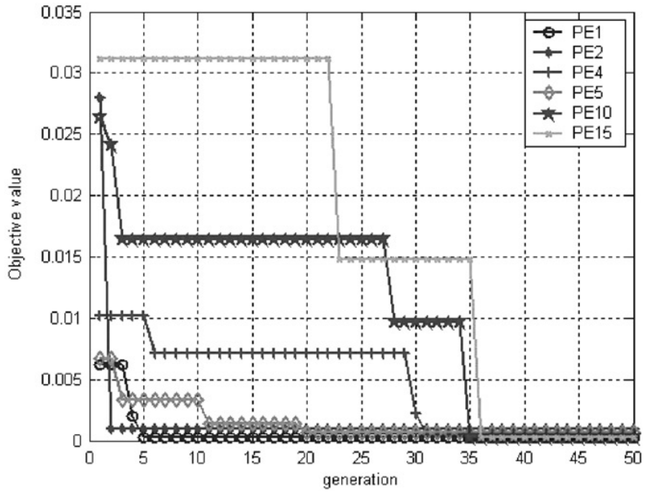

Fig. 16. Generation history in Case (a).

6. Conclusions

A methodology of parallel CEAs for the crack identification from the eigenfrequencies is proposed based on the fact that a crack has an important effect on the dynamic behavior of a structure. To estimate the crack parameters CEAs are adopted in parallel computing environmemnt.

The effectiveness of this technique is confirmed by two example problems. The crack parameters of the clamped-free beam problem are estimated within 3.3% error. In the case of the clamped–clamped plane frame problem the estimation has shown agreement within 3.1% error. It can be concluded that good agreements are obtained between the depth and location of the

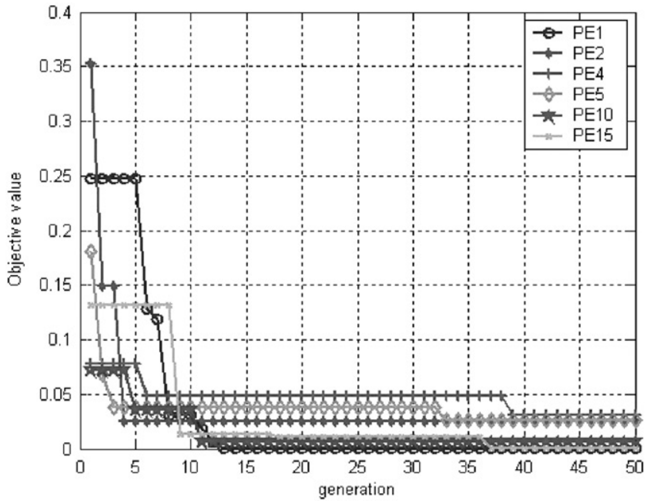

Fig. 17. Generation history in Case (b).

Table 4
Crack identification result: clamped–clamped plane frame

	Case (a)			Case (b)		
	Reference value	Result value	Relative error (%)	Reference value	Result value	Relative error (%)
α	0.003	0.0029	2.4	0.006	0.0062	3.1
L_1	0.15	0.149	0.53	0.24	0.238	0.71
f_1	267.31	267.308	0.1e−2	266.96	266.956	0.1e−2
f_2	1150.46	1150.463	0.2e−3	1151.18	1151.192	0.1e−3
f_3	1798.93	1798.925	0.3e−3	1828.79	1828.770	0.1e−2

estimated crack and those of the reference one and the results are promising with high parallel efficiency over about 91%.

Parallel CEAs can be applied to crack identification problems as a viable and generic problem-solving technique.

Acknowledgements

The authors are grateful for the support provided by a grant from the Korea Science and Engineering Foundation (KOSEF) and Safety and Structural Integrity Research Center at the SungKyunKwan University.

References

[1] T.C. Chondros, A.D. Dimarohonas, Identification of cracks in circular plates welded at the contour, in: ASME Design Engineering Technical Conference, St. Louis, 1979.
[2] T.C. Chondros, A.D. dimarogonas, Identification of cracks in welded joints of complex structures, Journal of Sound and Vibration 69 (1980) 531–538.

[3] G.D. Gounaris, A.D. Dimarogonas, A finite element of a cracked prismatic beam in structural analysis, Computational Structure 28 (1988) 309–313.

[4] G.D. Gounaris, V. Papazoglou, Three-dimensional effects on the natural vibration of cracked Timoshenko beams in water, Computational Structure 42 (1992) 769–779.

[5] T. Inagaki, H. Kanki, K. Shiraki, Transverse vibrations of a general cracked rotor bearing system, Journal of Mechanical Design 104 (1981) 1–11.

[6] P.S. Leung, The effects of a transverse crack on the dynamics of a circular shaft, in: Rotordynamics'92, International Conference on Rotating Machine Dynamics, Venice, 1992.

[7] N. Anifantis, P. Rizos,A. D. Dimarogonas, Identification of cracks on beams by vibration analysis, in: 11th Biennial ASME Conference on Mechanical Vibration and Noise, Boston, 1987.

[8] A.D. Dimarogonas, G. Massouros, Torsional vibrations of a shaft with a circumferential crack, Engineering Fracture Mechanics 15 (1981) 439–444.

[9] H.G. Nikolakopoulos, D.E. Katsareas, C.A. Papadopoulos, Crack identification in frame structures, Computer & Structures 64 (1-4) (1997) 389–406.

[10] M.W. Suh, J.M. Yoo, J.H. Lee, Crack identification using classical optimization technique, Journal of Key Engineering Materials 183–187 (2000) 61–66.

[11] T. Furukawa, G. Yagawa, Inelastic constitutive parameter identification using an evolutionary algorithm with continuous individuals, International Journal for Numerical Methods in Engineering 40 (1997) 1071–1090.

[12] C.A. Papadopoulos, A.D. Dimarogonas, Coupled longitudinal and bending vibrations of a cracked shaft, Journal of Vibrational Acoustics Stress Reliability Design 110 (1988) 1–8.

[13] H. Tada, The Stress Analysis of Cracks Handbook, Del Research Corporation, PA, 1973.

[14] W.D. Pilkey, W. Wunderlich, Mechanics of Structures Variational and Computational Methods, CRC Press, Boca Raton, FL, 1994.

[15] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[16] S. Obayashi, Multidisciplinary design optimization of aircraft wing platform based on evolutionary algorithms, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, IEEE, La Jolla, CA, 1998.

[17] A. Qing, C.K. Lee, L. Jen, Microwave imaging of parallel perfectly conducting cylinders using real-coded genetic algorithm, Journal of Electromagnetic Waves and Application 13 (1999) 1121–1143.

[18] M.-B. Shim, T. Furukawa, S. Yoshimura, G. Yagawa, M.-W. Suh, Efficient multi-point search algorithms for multiobjective optimization problem, Proceedings of Conference on Computational Engineering Science 5 (2) (2000) 459–462.

[19] M.-B. Shim, T. Furukawa, S. Yoshimura, G. Yagawa, M.-W. Suh, Pareto-based continuous evolutionary algorithms for multiobjective optimization, Engineering Computations: International Journal for Computer-Aided Engineering and Software 19(1) 2002.

[20] T.C. Fogarty, R. Huang, Implementing the genetic algorithm on transputer based parallel processing systems, in: H.-P. Schwefel, R. Männer (Ed.), Parallel Problem Solving from Nature, 1st Workshop, Dortmund, Germany, 1990, 145–149.

[21] D. Abramson, J. Abela, Aparallel genetic algorithm for solving the school timetabling problem, in: Proceedings of the 15th Australian Computer Science Conference (ACSC-15), Vol. 14, 1992, pp. 1–11.

[22] P. Grosso, Computer simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model, Ph.D. Thesis, University of Michigan, 1985.

[23] R. Tanese, Parallel genetic algorithms for a hypercube, in: J.J. Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987.

[24] T. Belding, The distributed genetic algorithm revisited, in: D. Eshelmann (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1995.

[25] G. Robertson, Parallel implementations of genetic algorithms in a classifier system, in: J.J. Grefenstette (Ed.), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987.

[26] V. Gordon, D. Whitley, A. Böhm, Dataflow parallelism in genetic algorithms, in: R. Männer, B. Manderick (Eds.), Parallel Problems Solving from Nature II, Brussels, Belgium, 1992, pp. 533–542.