Inverse Problems, Design and Optimization Symposium
Rio de Janeiro, Brazil, 2004

TOPOLOGICAL DESIGN OF STRUCTURES USING POPULATION-BASED OPTIMISATION METHODS

Sujin Bureerat
Department of Mechanical Engineering,
Khon Kaen University, Thailand, 40002
sujbur@kku.ac.th

ABSTRACT

In this paper, a number of well-established population-based optimisation methods i.e. Genetic Algorithms, Simulated Annealing and Population Based Incremental Learning are briefly reviewed and compared in terms of their philosophical basis. The use of the optimisation methods for topology optimisation is demonstrated. The paper also presents an efficient numerical technique to prevent checkerboard formation in topology design. From the optimum solutions obtained using the various optimisation methods with many design conditions, the performances of the methods are compared and discussed.

INTRODUCTION

Topology optimisation is a special type of structural shape optimisation. This design process is employed when designers need to find a new structural configuration for particular use as shown in figure 1. In topology optimisation problem, with a given design domain, the task is to find the best structural layout that gives the optimum of desired objective functions e.g. weight, system compliance, deflection and natural frequency whilst fulfilling design constraints [1]. In numerical process, by the use of Finite Element Method (FEM) for structural analysis, topology optimum design can be performed by discretising a structure into a number of connected finite elements. Design variables determine the distribution of element density, which means that elements with nearly zero density represent holes on the structure whereas other elements indicate the existence of structural material.

One of the most preferable optimisation methods for topology design is Optimality Criteria Method (OCM) [2] as it is arguably the most powerful method for this task. Also, the classical gradient-based methods such as Sequential Linear Programming (SLP) and the Method of Moving Asymptotes (MMA) were implemented [1]. There have been a few publications concerning the applications of population-based methods for topology design and most of them referred to Genetic Algorithms (GAs) e.g. [3-5]. Despite the capability of reaching a global optimum of GAs, the methods seem to be ineffective when used in topology design because they are time consuming and have no consistency. This is due to the large number of topological design variables and, therefore, a huge number of function evaluations are needed when performing GAs for such design. However, it cannot be totally concluded that all the population-based or evolutionary methods are inferior since other evolutionary methods are rarely applied.

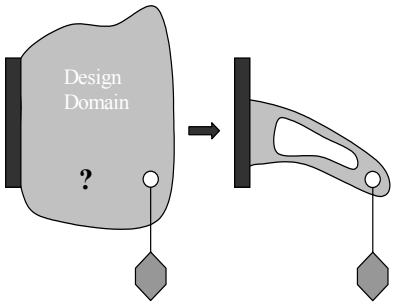

Figure 1 Topology Optimisation

This paper demonstrates the use of population-based optimisation methods, often called evolutionary algorithms, for solving structural topology optimisation. As four-node quadrilateral membrane finite element is used for structural analysis, the checkerboard suppression scheme is also performed. Structural topology is presented by binary string whereas the objective function is

the weighted sum of structural mass, compliance and checkerboard penalty. The evolutionary algorithms consist of Genetic Algorithms with three different sets of GA operators, Simulated Annealing (SA), Population Based Incremental Learning (PBIL) and Stud-Genetic Algorithm (Stud-GA). The methods are briefly reviewed and implemented on a number of topology optimisation problems. The main investigation is aimed at comparing the performance of evolutionary methods that used elites to create new population versus the methods that use traditional crossover to create population. The optimum results obtained from using the various methods are obtained, illustrated and compared in terms of convergence rate and consistency.

POPULATION-BASED OPTIMISATION METHODS

Population-based methods sometimes referred to as Evolutionary Algorithms (EAs) are the optimisation methods that search for optima based upon evolutionary mechanisms [6]. The methods start the search with a group of initial solutions called population and the population are then evolved, in some manners, generation by generation until reaching the optimum. The evolutionary methods presented here are as follows:

Genetic Algorithms

Genetic algorithms are probably the best known of the evolutionary algorithms. This approach can be best thought of as mimicking Darwinian natural selection in that a population of solutions (genes) is generated and then the next generation is produced by mating pairs of these genes. The genes have the opportunity of being selected based on their merit. The selected genes are reproduced by means of crossover and mutation yielding the new population or next generation. The next generation is iteratively evolved until an optimum is achieved [7].

For the GA in this paper, topology design variables are parameterised by a series of binary strings. A new population can be obtained by two genetic operators that are crossover and mutation. In crossover operator, a pair of genes is selected based upon a probability according to their fitness. The so-called one-point crossover is illustrated in figure 2. A cutting point is randomly selected and the second part of the pairs is swap resulting in the new pair of offspring. With this concept, multiple-point crossover can be achieved by randomisation of multiple cutting points and exchanging some cutting parts of parents' genes. This process is repeated until the offspring have the same size as their parents. Note that, generally, each pair of selected gene is allowed to be crossed over by the predefined probability.

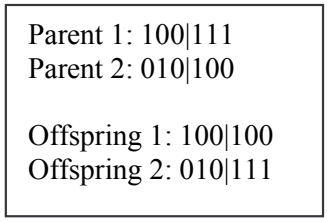

Figure 2 one-point crossover in GAs

For mutation, an element in a gene is chosen at random and is changed from '1' to '0' (or vice versa) this operator is also allowed to take place on each current gene by a given probability as crossover operator. In addition to these two main operators, the best genes from each generation are saved directly to the next generation, ensuring that the best solutions are not lost and some new-blood, randomly created genes, are the additional members to the new population so as to prevent premature convergence.

Stud-Genetic Algorithm

A slight modification of classical GA is called Stud-GA which claims to improve significantly the performance of the traditional GA whilst maintaining its simplicity and binary string representation [8]. Rather than maintaining a large population of different solutions, the best gene (stud) from an initial population is chosen and then the new population is generated by mutating the stud until the offspring have the same size as the initial population. After mutation, bit positions of each offspring are allowed to be shuffled by the given probability. The process is repeated until convergence is reached

Population-Based Incremental Learning

Population Based Incremental Learning (PBIL) [9] has a different feature from GA in that the population is represented by the probability vector of being '1' of each bit position of binary strings. Figure 3 shows probability vectors used in PBIL where row vectors of the population matrix represent genes. It is shown that one probability

vector can form a variety of populations. A probability of 1 indicates that all the bits in a column of the population are '1' whereas a probability of '0' means that the column is full of zeros.

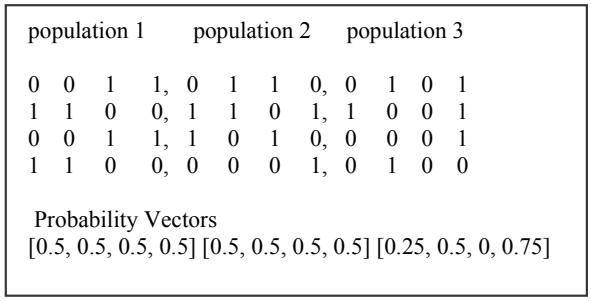

Figure 3 Populations and their probability vectors

Initially, the search procedure starts with the initial probability vector whose elements are full of '0.5'. An initial population corresponding to the probability vector is created. The best gene is selected and the next probability vector P_i^{new} is found using the relation

$$P_i^{new} = P_i^{old}(1 - LR) + b_i LR \qquad (1)$$

where LR is called the learning rate, a value between 0 and 1 that is gradually reduced during the optimisation process and b_i is the i^{th} bit of the best gene. It is also useful to apply mutation to the probability vector at some predefined probability such that

$$P_i^{new} = P_i^{old}(1 - ms) + \text{rand}(0 \text{ or } 1).ms \qquad (2)$$

where ms is the amount of shift used in the mutation. The best gene is also carried over into the next generation, as with the traditional GA approach. The probability vector is updated iteratively until convergence is reached.

Simulated Annealing

Simulated annealing [10&11] sometimes is classified to be evolutionary method as GA and the others. The method is based upon mimicking the random behaviour of molecules during the annealing process, which involves slow cooling from a high temperature. As the temperature cools, the atoms line themselves up and form a crystal, which is the state of minimum energy in the system. However, if the metal is cooled too quickly, the minimum energy state is not reached. The basic algorithm follows and is usually formulated as a minimisation problem.

The search procedure of SA is to start with a single initial solution with fitness f is taken and then adjusted in some manner to produce a candidate solution with fitness f'. If $f' < f$, then f' is taken onto the next iteration, however, in cases that $f' > f$, the candidate value may still be chosen depending upon the Boltzmann probability

$$\Pr = e^{(f'-f)/T} \qquad (3)$$

where T is the annealing temperature, which will be gradually reduced during the process. The key role of SA search is that the discovery of a new candidate. As traditionally SA is the method without using derivatives, the candidate is created, by mutating on a current solution. The more effective procedure is that creating a set of new candidates and then selecting the best of them to be compared with their parent.

TOPOLOGY OPTIMSATION USING EVOLUTIONARY ALGORITHMS

One of the most well-known topology optimisation problems is the classical compliance minimisation of plate structures, which can be posed as:

$$\text{Min: } c(\boldsymbol{\rho}) = \mathbf{U}^T\mathbf{K}\mathbf{U} \qquad (4)$$
$$\text{Subject to } m(\boldsymbol{\rho}) < m_0$$

where the vector of design variables $\boldsymbol{\rho}$ is the vector of element densities of the discretised structure, c is structural compliance, $\mathbf{U}$ is structural displacement due to applied forces $\mathbf{F}$, $\mathbf{K}$ is structural stiffness matrix and m is structural mass.

Since most of the population-based methods were developed for unconstrained optimisation, for simplicity, the compliance minimisation can be modified to be weighted-sum objective function as

$$\text{Min: } f(\boldsymbol{\rho}) = w_1\mathbf{U}^T\mathbf{K}\mathbf{U} + w_2 m \qquad (5)$$

where w_i are the weighting factors. It has been found that, with a proper set of weighting factors [12], this design strategy is as effective as the constrained problem in (4). As the structural analysis is carried out by membrane finite

element analysis, it should be noted that checkerboard patterns could occur on the resulting topology design because of numerical instability. To prevent such undesirable phenomenon, additional objective called checkerboard penalty *Cb* [13] is introduced to the optimisation problem. The checkerboard penalty is the slight modification of that presented in [14]. The design problem is, therefore, of the form:

$$\text{Min: } f(\boldsymbol{\rho}) = w_1 \mathbf{U}^T \mathbf{K} \mathbf{U} + w_2 m + w_3 Cb \quad (6)$$

From a discretised rectangular design domain as shown in figure 4, let the structure have $(m+1)\times(n+1)$ elements. Thus, there are $m\times n$ interior nodes as shown and, at each interior node, there are 4 elements surrounding it. At a current interior node, if the 4-element density pattern matches any of the two cases in figure 5, the local checkerboard penalty value is one, otherwise, it is zero.

$$cb_i = \begin{cases} 1, \text{case\#1 and \#2} \\ 0, \text{otherwise} \end{cases} \quad (7)$$

The total checkerboard penalty value, *Cb*, of a typical structure can be computed as

$$Cb = \sum_{i=1}^{m\times n} cb_i \,. \quad (8)$$

It can be concluded that $Cb = 0$ represents a topology without checkerboard which is the minimum point of the penalty function.

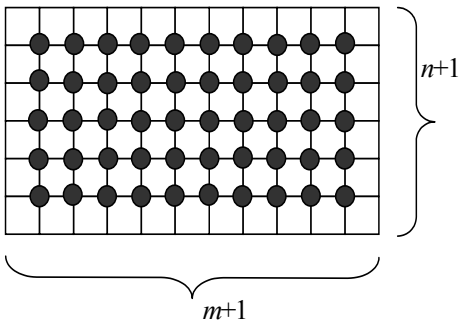

Figure 4 Discretised structure

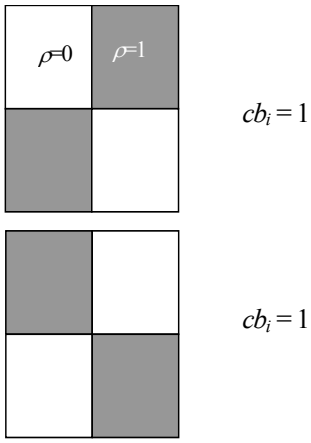

Figure 5 Patterns to be penalised

As being universal methods, any aspect of design variables can be used with EAs. For this work, topology of a structure is represented by series of binary strings with the same size as the number of structural elements. A bit '1' gives material existence while '0' represents void in the structure. Also note that in this paper '1' means 1 unit of density whereas a bit '0' represents 0.000001 unit of density in stead of zero-thickness so as to prevent singularity in global stiffness matrix of structural system.

TESTING CASES OF DESIGN

In this study, the population-based methods are implemented on the compliance minimisation design of a classical MBB beam shown in figure 6. The beam, made up of material with $E = 200\times10^9$ N/m^2 and $\nu = 0.3$, is meshed to be 20×10 finite elements. The point load is set to be 100 N. The optimisation algorithms that are used to solve the problem are:

- GA01 genetic algorithm with probability of crossover and mutation being 0.9 and 0.1 respectively.
- GA02 genetic algorithm with probability of crossover and mutation being 0.1 and 0.9 respectively.
- GA03 genetic algorithm with probability as used in GA01 plus evolutionary direction operator [15].
- StudGA with 0.01 probability to shuffle gene.
- PBIL with 0.01 probability of mutation.
- SA with initial temperature $T_0 = 10$ and the predefined final temperature $T_1 = 0.001$.

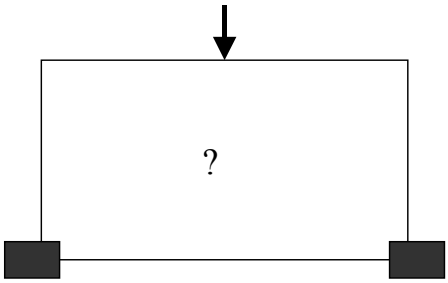

Figure 6 MBB beam

The design test cases are categorised by objective function, number of iterations or generations and population size as follows:

1. The objective function excludes checkerboard penalty as in (5) where $[w_1,w_2] = [0.25\ 0.75]$. The number of generations and population size for the optimisation methods are 100 and 50 respectively, in exception of SA which uses 250 loops whereas 20 candidates to challenge their parent are created on each iterations.
2. The objective function includes checkerboard penalty as in (6) where $[w_1\ w_2\ w_3] = [0.14\ 0.43\ 0.43]$. The number of iterations and population size are the same as in case 1.
3. The objective function is the same as in case 2. All the methods are performed with the same number of iterations and population size i.e. 250 iterations and 20 individuals in one generation. This design case is set to examine the effect of population size on evolutionary search.

In each design case, all the methods start with the same initial population. Each method is applied to solve each design case five runs and the best result of each run is taken as the optimum result. This means that the mean value of five optimum objectives measures the convergence rate while the standard deviation of the optimum objectives measures the consistency of the optimisation methods. Note that population size and generations are intentionally set for benchmarking the performance of the methods which means that some methods may not reach or even close to the strict optimum. The reason of using those design parameters is to point out the superiority of some particular methods to the others on this type of design.

NUMERICAL RESULTS

The numerical results of case 1 are shown in table 1 and illustrated in figure 7&8. The search history of the methods is shown in figure 9. From the table, Stud-GA is most consistent and also gives the best convergence rate with SA comes second.

Table1 Optimum results of case 1

No.	GA01	GA02	GA03	StudGA	PBIL	SA
1	1.8131	1.6314	1.7549	1.4757	1.5664	1.4806
2	1.8026	1.6145	1.7464	1.4774	1.5511	1.4852
3	1.7971	1.6160	1.7066	1.4899	1.5747	1.4756
4	1.8260	1.6180	1.6726	1.4917	1.5868	1.4894
5	1.8092	1.6413	1.6862	1.4880	1.5423	1.4982
AV	1.8096	1.6242	1.7133	1.4845	1.5643	1.4858
STD	0.0099	0.0104	0.0324	0.0066	0.0160	0.0077

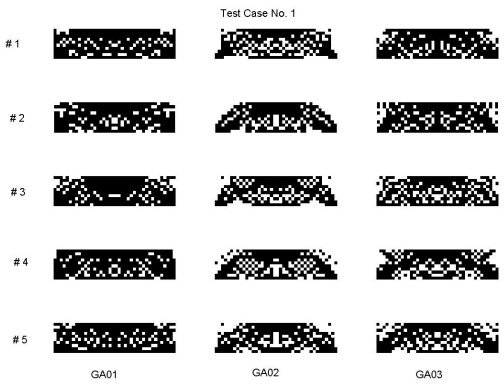

Figure 7 Optimum results of case 1, GA01 GA02 GA03

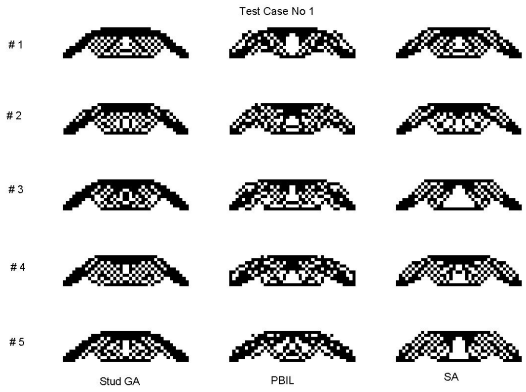

Figure 8 Optimum results of case 1, Stud-GA PBIL SA

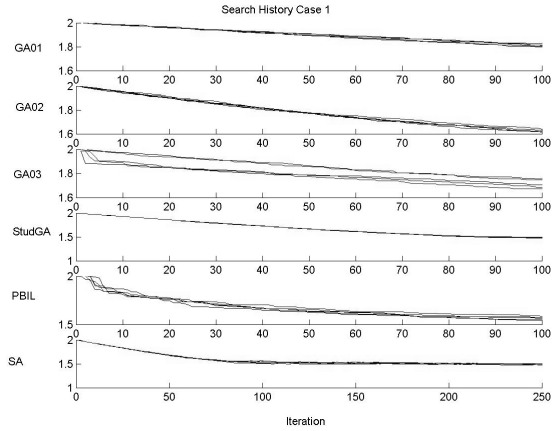

Figure 9 Search history of case 1

In design test case number 2, the results are in table 2 and the optimum results from five attempts are illustrated in figure 10&11 whilst search history is in figure 12. The most consistent is GA03 while SA gives the best convergence rate. Although the GA03 has the lowest standard deviation from five tries the obtained results are still far from the real optimum. Therefore, SA is the best in overall.

Table 2 Optimum results of case 2

No.	GA01	GA02	GA03	StudGA	PBIL	SA
1	1.8411	1.7120	1.7755	1.5411	1.6632	1.5026
2	1.8235	1.6811	1.7647	1.5485	1.7423	1.5592
3	1.8284	1.7221	1.7598	1.5630	1.6503	1.4935
4	1.8123	1.7170	1.7811	1.6225	1.7167	1.5069
5	1.8320	1.7100	1.7678	1.5483	1.6474	1.4976
AV	1.8274	1.7085	1.7698	1.5647	1.6840	1.5120
STD	0.0095	0.0143	0.0076	0.0298	0.0384	0.0240

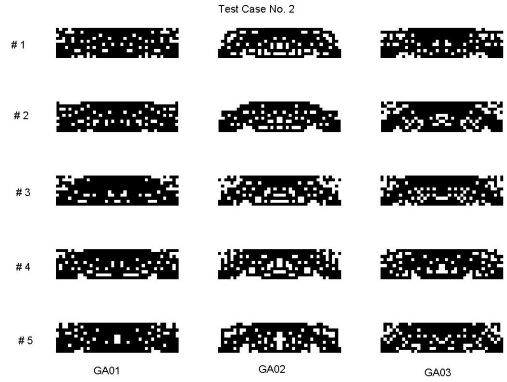

Figure 10 Optimum results of case 2, GA01 GA02 GA03

Figure 11 Optimum results of case 2, Stud-GA PBIL SA

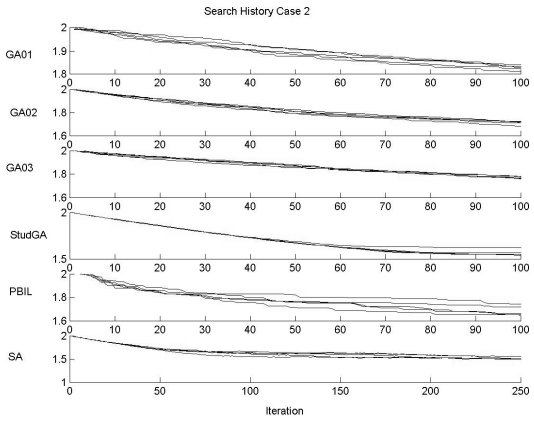

Figure 12 Search history of case 2

The results of case 3 are in table 3, figure 13&14 and search history in figure 15. In this test case, the best method in terms of convergence rate and consistency is SA.

Table 3 Optimum results of case 3

No.	GA01	GA02	GA03	StudGA	PBIL	SA
1	1.7595	1.6029	1.7514	1.5298	1.7382	1.4827
2	1.7752	1.6684	1.7146	1.5592	1.7914	1.5057
3	1.7634	1.6374	1.7640	1.6362	1.7932	1.4959
4	1.7325	1.5735	1.7147	1.5370	1.8440	1.4915
5	1.7750	1.5865	1.6996	1.6101	1.7873	1.5271
AV	1.7611	1.6137	1.7288	1.5745	1.7908	1.5006
STD	0.0156	0.0347	0.0245	0.0418	0.0335	0.0152

Figure 13 Optimum results of case 3, GA01 GA02 GA03

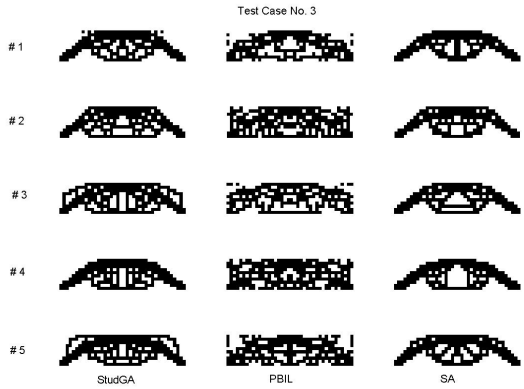

Figure 14 Optimum results of case 3, Stud-GA PBIL SA

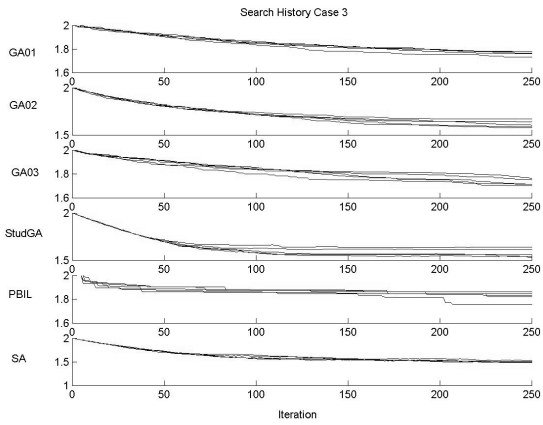

Figure 15 Search history of case 3

CONCLUSIONS AND DISCUSSIONS

From the numerical experiments, it can be concluded that mutation operator is the more powerful mechanism than crossover in topology design of plate-like structures since the design results obtained from Stud-GA and SA are apparently superior to that obtained from the classical GAs. This is also supported by GA02 overcoming GA01 and GA03. The SA and Stud-GA are equally good for topology design with SA slightly better when dealing with checkerboard. GA01 with crossover as the main operator is the worst. GA02 is the best out of the three GAs. The evolutionary direction in [15] can enhance the search performance of GA with crossover yet is not implemented with mutation domination as GA02. For the intermediate performance method as PBIL, the search by this approach can be affected by its population size which can be said that the greater size yields the better designs.

The checkerboard penalty technique can effectively prevent checkerboard patterns in topology design. However, numerical experimens in the prevoius work [13] showed that the proper set of weighting factors [w_1 w_2 w_3] needs to be given so that the checkerboard-free topology design is achievable. The higher value of w_3 normally results in the checkerboard-free topology.

The methods that use elite or their best genes for generating new population are better than that uses traditional crossover as classical GAs. However, it cannot be a definite conclusion as the evolutionary methods are merely implemented on one structural design problem.

REFERENCES

1. M.P. Bensϕe and O. Sigmund, *Topology Optimization Theory, Method and Applications*, Springer-Verlag, Berlin Heidelberg, 2003.

2. O. Simund, A 99 Line Topology Optimization Code Written in MATLAB, *Struct. Multidisc. Optim*, 21, 120 – 127 (2001).

3. C. Kane, F. Jouve & M. Schoenauer, Structural Topology Optimization in Linear and Nonlinear Elasticity Using Genetic Algorithms, *21st ASME Design Automatic Conf.*, Boston, (1995).

4. C. Kane and M. Schoenauer, Topological Optimum Design Using Genetic Algorithms, *Control & Cybernetics*, 25, **5**, 1059 – 1088 (1996).

5. M. Jakiela, C. Chapman, J. Duda, A. Adewuya, and K. Saitou, Continuum structural topology design with genetic algorithms, Comp. Methods in App. Mech. and Eng., 86:2, 339–356 (2000).

6. S. Bureerat and J. E. Cooper, Evolutionary Methods for the Optimisation of Engineering

Systems, *Opt. in Control: Methods and Applications, IEE*, London, 1/1-1/10 (1998).

7. G. Lindfield and J. Penny, *Numerical Methods Using MATLAB*, Ellis Horwood, 1995, p. 284 - 294.

8. W. Khatib and P. Fleming, The Stud GA: A Mini-Revolution, *5th Int. Conf. on Parallel Problem Solving From Nature*, (1998).

9. S. Baluja, Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, *CMU_CS_163*, (1994).

10. S. Kirkpatrik, C. D. Gelatt Jr. and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, 220, **4598**, 671-680 (1983).

11. W.A. Benage and A. K. Dhingra, Single and Multiobjective Structural Optimization in Discrete-continuous Variables Using Simiulated Annealing, Int. J. of Num. Methods in Eng., 38, 2753-2773 (1994).

12. T. Kunakote and S. Bureerat, Structural Topology Optimisation Using Evolutionary Algorithms, *17th ME-NETT*, Thailand, (2003). (in Thai).

13. S. Bureerat and T. Kunakote, A Simple Checkerboard Suppression Technique for Topology Optimisation Using Simulated Annealing, *17th ME-NETT*, Thailand, (2003).

14. A. Poulsen, A simple Scheme to Prevent Checkerboard Patterns and One-Node Connected Hinges in Topology Optimization, *Struct. Multidisc. Optim*, 24, 396 – 399 (2002).

15. K. Yamamoto and O. Inoue, New Evolutionary Direction Operator for Genetic Algorithms, *AIAA,* 33*,* **10**, 1990-1993 (1995).