Discrete Optimization of Ship Structures with Genetic Algorithms

by Hisashi Nobukawa*, *Member* Guoqiang Zhou*, *Member*

Summary

A discrete optimization method using genetic algorithms is developed for the optimization of ship structures. In this method, the constrained minimization problem is first transformed into an unconstrained one by a penalty term depending on the degree of constraint violation. Since the search procedure of the genetic algorithm is done based on the evaluation of fitness function, the unconstrained minimization problem is further converted into a maximization of the fitness function. The discrete design variables are coded into a binary string of finite length. The search procedure from generation to generation is carried out by a simple genetic algorithm with the genetic operators of reproduction, crossover and mutation. A cargo ship with large hatch opening is taken as the numerical example for the illustration purpose. The influences of penalty coefficient, population size, crossover probability and mutation probability on the optimum design are investigated. The comparison between the genetic algorithm and the multiplier method is also made. It demonstrates that the present method can handle the optimization of ship structures with discrete design variables well.

1. Introduction

In practical structural design, most design variables have the discrete nature, that is, they must be selected from a given finite set of values. For example, structural elements must be chosen from those that are already commercially available to reduce the fabrication cost. However, up to now the majority of optimization methods assume that the design variables are continuous. Their optimum results may not be feasible even though they are mathematically optimal. Discrete optimization procedures are more rational, as its every candidate design satisfying all design constraints is a practical feasible one. Therefore, the discrete nature of design variables is of great importance in solving practical optimization problems.

A few methods, such as the adaptive numerical optimization procedure¹⁾, have been suggested to deal with the discrete optimization problem, but most of them are still based on mathematical programming. These methods require additional computational effort and do not guarantee true minimum value. Obviously, there is a need for a simple yet powerful method to deal with the discrete optimization problem.

Genetic algorithms were first proposed by John Holland²⁾. Since then, they have been refined and developed to solve a wide range of problems³⁾. Genetic

algorithms are search procedures based on the mechanics of natural genetics and natural selection. They combine Darwin's principle of survival of the fittest with genetic operators abstracted from nature to form a robust search mechanism. Genetic algorithms are computationally simple, but powerful in their search for improvement. Furthermore, they are not limited by restrictive assumptions about search space, such as continuity or existence of derivatives. Many papers and dissertations have established the validity of the technique for function optimization.

In the present paper, we develop a discrete optimization method for the optimization of ship structures based on genetic algorithms. The discrete design variables are coded into a binary string of finite length. A simple genetic algorithm with the genetic operators of reproduction, crossover and mutation is used for the search procedure. A cargo ship with large hatch opening is taken as the numerical example. The influences of penalty coefficient, population size, crossover probability and mutation probability on the optimum design are investigated. The differences between the genetic algorithm and the multiplier method are distinguished.

Reference 7) presented a method for optimization of ship structural design by genetic algorithm. It showed that discrete design variables with the regular values between their maximum and minimum values can be dealt with in that method. In the present paper, we place a special emphasis on the discrete structural optimization and develop a method using genetic algorithms based on the schema theorem. The discrete design variables can be handled well regardless of their

* Faculty of Engineering, Hiroshima University

Received 10th Jan. 1996

Read at the Spring meeting 15, 16th May 1996

discrete values with or without the above-mentioned structure. Moreover, the multiplier method is compared with the genetic algorithm in this paper and the comparison results are concluded.

2. Genetic Algorithm Approach

2.1 Features of Genetic Algorithm Approach

The genetic algorithm differs from traditional optimization methods in many ways. Its main features are described as follows.

(1) Population processing

The genetic algorithm uses a population of points at a time in contrast to the single-point approach by traditional optimization methods. That means, at a given time, the genetic algorithm processes a number of designs.

In traditional optimization methods, a single point is used at a time in the search procedure. This point-to-point method is dangerous because it is easy to get trapped in a local optimum to multimodal (many-peaked) problems. By contrast, the genetic algorithm works from a population of points simultaneously, climbing many peaks in parallel, so that the probability of finding a false peak is reduced.

(2) Randomized operators

The genetic algorithm uses randomized operators in place of the usual deterministic ones. While randomized, the genetic algorithm is no simple random walk. The random operators improve the search process in an adaptive manner.

From the engineering optimization points of view, the most important goal of optimization is improvement and attainment of the optimum is much less important. As the random operators with a structured information exchange permit the genetic algorithm to conduct a highly exploitative search for improvement, the behavior of the genetic algorithm accords with the goal of engineering optimization.

(3) Separation of domain knowledge from search

The genetic algorithm utilizes only objective function (payoff) information. The derivative and other auxiliary information are not used for guiding search.

In most practical optimization problems, both objective functions and design constraints are the implicit functions to design variables. It is difficult to calculate their derivatives directly. As the genetic algorithm does not require problem-specific knowledge to carry out a search, it is very suitable for the optimization of complex system with the above-mentioned implicit character.

(4) Working on coded design variables

The genetic algorithm works on coded design variables, rather than the design variables themselves. In the genetic algorithm, the design variables are coded into finite length strings. These strings represent artificial chromosomes. Every character in the string is an artificial gene. The genetic algorithm processes

successive populations of these artificial chromosomes in successive generations.

A chromosomal representation of design variables permits an application of the genetic algorithm across a spectrum of disciplines. If the relationship between the discrete variables and the coding strings of finite length is established, the genetic algorithm can deal with the discrete optimization problems as we will discuss later.

2.2 Fitness Function

The nonlinear programming with unequal constraints can be described as follows:

$$\min f(\mathbf{X}) \quad \text{s.t. } g_i(\mathbf{X}) \geq 0 \quad i=1, 2, \dots, m \quad (1)$$

where $\mathbf{X}=(X_1, X_2, \dots, X_n)^T$ is a vector containing n design variables, $f(\mathbf{X})$ is the objective function to be minimized, $g_i(\mathbf{X})$ is the i th constraint and m is the number of constraints.

Since the genetic algorithm does the search procedure based on the fitness function, the constrained minimization problem described as Eq.(1) needs to be transformed for the purpose of solving it by the genetic algorithm. In this study, the constrained minimization problem is first transformed into an unconstrained one by a penalty term depending on the degree of constraint violation. The augmented objective function of unconstrained minimization problem with the penalty term is expressed as

$$\Phi(\mathbf{X}) = f(\mathbf{X}) \left(1 - c_p \sum_{i=1}^m [\min [0, g_i(\mathbf{X})]] \right) \quad (2)$$

where $\Phi(\mathbf{X})$ is the augmented objective function of the unconstrained minimization and c_p is the influence coefficient of the penalty term called the penalty coefficient for short.

Then the augmented objective function $\Phi(\mathbf{X})$ is converted into a corresponding fitness value. The establishment of fitness function in the genetic algorithm must follow two principles: (1) the fitness value of every individual should be positive, and (2) the best individual should have the maximum fitness value. To do this in a simple way, $\Phi(\mathbf{X})$ is subtracted from a large constant to form the fitness function. In this study, the constant value is assigned to the maximum value of $\Phi(\mathbf{X})$ from all the individuals in the population. The expression of fitness function becomes

$$F_j(\mathbf{X}) = \Phi_{\max}(\mathbf{X}) - \Phi_j(\mathbf{X}) \quad (3)$$

where $F_j(\mathbf{X})$ is the fitness function of the j th individual and $\Phi_{\max}(\mathbf{X})$ is the maximum value of the augmented function from all the individuals in the population.

2.3 Coding and Decoding of Discrete Design Variables

An essential characteristic of a genetic algorithm is the coding of the variables that describe the problem. The most common coding method is to transform the variables to a binary string of specific length. This string represents the chromosome of the problem and the bit value of the chromosome called the allele represents the number of zeros and ones in the binary string.

Coding a discrete design variable is to establish its

corresponding relationship to a binary string of specific length. For instance, with a four-bit binary string we can obtain decimal integer numbers between 0(0000) and 15(1111). It gives us some hints that a discrete design variable with sixteen discrete values can correspondingly be coded into a four-bit binary string to represent. In general, a binary string with the length l can correspondingly represent a discrete design variable with 2^l discrete values. For a specific problem, that depends on more than one variable, a multivariable coding is constructed by simple concatenating the substring of each single variable into a complete string. Each variable may have its own length for the particular application.

Decoding a discrete design variable is to transform its corresponding binary string into its discrete value. We assume that the i th design variable X_i corresponds to a binary substring with the length l_i , and further suppose that the substring has a number which r th allele's value is a_r , $a_r \in \{0, 1\}$. Decoding this binary number, we get a decimal integer u :

$$u = 1 + \sum_{r=1}^{l_i} a_r 2^{r-1} \quad (4)$$

It means that the above substring value corresponds to the u th discrete value of the i th design variable. Thus, the discrete design variable can be treated in the genetic algorithm only if it is coded into a corresponding binary substring. As the count in a binary string starts from zero and not from one, one is added in Eq. (4).

If a discrete design variable has the regular discrete values with a step between its maximum and minimum values, its discrete value can be further calculated by an expression directly. In this case, the discrete value is calculated if the maximum and minimum values of the i th design variable are given

$$X_i^u = X_{i \min} + \frac{X_{i \max} - X_{i \min}}{2^{l_i} - 1} (u - 1) \quad (5a)$$

where X_i^u is the u th discrete value of the i th design variable, $X_{i \min}$ is its minimum value and $X_{i \max}$ is its maximum one.

In addition, the discrete value is also computed if the minimum value and the step value of the i th design variable are known

$$X_i^u = X_{i \min} + (u - 1)s_i \quad (5b)$$

where s_i is the step value of the i th design variable.

2.4 Mechanics of Genetic Search

In this study, a simple genetic algorithm³⁾ that yields good results in many practical problems is introduced to do the search procedure. It consists of three genetic operators: reproduction, crossover and mutation. The mechanics of the simple genetic algorithm are surprisingly simple, involving nothing more complex than random number generation and copying strings as well as swapping partial strings.

To use a genetic algorithm, we must first code the design variables into a finite length string. With a well

defined objective function and coding, an initial population of strings is randomly generated. By decoding the individual strings of the initial population, the solution for each specific instance is determined and the value of the fitness function that corresponds to this individual string is evaluated. This applies to all members of the population. According to the fitness values of individual strings, the initial population is transmitted by the genetic operators of reproduction, crossover and mutation.

(1) Reproduction operator

Reproduction is a selection procedure whereby individual strings are selected based on their fitness values relative to that of the population. There are many ways of achieving effective reproduction. One simple scheme selects individual strings in the population according to the following probability

$$p_j = \frac{F_j(X)}{\sum_{i=1}^{n_p} F_i(X)} \quad (6)$$

where p_j is the selection probability of the j th individual string for reproduction and n_p is the size of population.

The reproduction operator emphasizes the survival of the fittest. Individuals with higher fitness values have a greater chance of being selected for mating and subsequent genetic action. Consequently, highly fit individuals live and reproduce, and less fit individuals die. Obviously, this operator is an artificial version of natural selection, a Darwinian survival of the fittest among string creatures.

(2) Crossover operator

After reproduction, a crossover operation proceeds with a specified crossover probability in two steps. First, individual strings in the mating pool generated by the reproduction operator are mated at random. Second, each pair of individual strings undergoes crossing over as follows: supposed that l is the length of the string, an integer position i along the string is decided uniformly at random between 1 and the string length less one, $[1, l-1]$. Two new strings are created by swapping all characters between position $i+1$ and l inclusively, as shown in Fig. 1(a). The resulting strings are placed in the new population continuously until the population is filled with new progenies.

(3) Mutation operator

Finally, the mutation operator is applied to the new

Fig. 1 Genetic operators of crossover and mutation

strings with a small mutation probability. A mutation is the occasional random alteration of an allele's value in a new string. In a binary coding, it means changing a 1 to a 0 and vice versa, as shown in Fig. 1(b).

Following reproduction, crossover and mutation, the new generation is ready to be tested. The above transmission process undergoes repeatedly from generation to generation until the best generation is derived or a given generation number is reached.

The genetic algorithm gets its power from the genetic operators. If reproduction according to fitness combined with crossover gives genetic algorithms the bulk of processing power, mutation plays a decidedly secondary role in the operation of genetic algorithms. Mutation is needed because it is an insurance policy against a premature loss of important genetic material at a particular position.

2.5 Schema Theorem

The power of genetic algorithms can be explained by the schema theorem³⁾. In genetic algorithms, a schema is a similarity template describing a subset of strings with similarities at certain string positions. For instance, the schema $1^{*}0$ describes all four-bit binary strings that begin with one and end with zero (1000, 1010, 1100, 1110). Here $*$ is a metasymbol used to represent either a 0 or a 1.

In order to state the schema theorem, let us introduce two terms to describe a schema: schema order and defining length. The order of a schema H , denoted by $o(H)$, is the number of fixed positions in H . The defining length of a schema H , denoted by $\delta(H)$, is the distance between the outermost fixed position in H . For the schema $H=1^{*}0$, $o(H)=2$ and $\delta(H)=3$. With these definitions, a particular schema H in a population at generation $t+1$ receives an expected number of copies under reproduction, crossover and mutation as given by the following equation

$$m(H, t+1) \geq m(H, t) \frac{F(H)}{\bar{F}} \left[1 - p_c \frac{\delta(H)}{l-1} - p_m o(H) \right] \quad (7)$$

where $m(H, t)$ is the number of schema at generation t , l is the string length, $F(H)$ is the average fitness of the strings representing schema H , $\bar{F}$ is the average fitness of the entire population, p_c is the crossover probability and p_m is the mutation probability.

Eq.(7) is the fundamental theorem of genetic algorithms, called the schema theorem. It states that relatively short, lower order schemata with above average fitness values receive exponentially increasing number of trials in subsequent generations. The schemata with these properties are called building blocks. This concept of allocating exponentially increasing number of trials to many building blocks in parallel is the essence of genetic search. As the search progresses, lower schemata combine to yield higher order schemata, thereby leading to optimal or near-optimal solutions.

3. Optimization Model of Ship Structures

In this study, a cargo ship with a large hatch opening is taken as the numerical example for the illustration purpose. Its optimization model is considered as follows.

3.1 Objective Function

The hull construction cost of hold length between two transverse bulkheads for the cargo ship is taken as the objective function. The total cost consists of the material cost and the welding cost. The material cost is further divided into the cost of plate steel and the cost of shaped steel.

All structural members are assumed to be made of mild steel. The unit price per ton of plate steel is 90,000 yens and that of shaped steel is 105,000 yens. The price of fillet welding per unit welding length in meter with unit welding leg length in centi-meter is 4,200 yens.

3.2 Design Variables

Nine dimensions of ship structural members in the midship segment, such as plate thickness, web depth, flange width and longitudinal space, are chosen as the design variables shown in Fig. 2.

3.3 Design Constraints

In this study, all design constraints are normalized to treat them equally. They are described as follows

(1) Bending strength

This constraint is considered for the following structural members

- transverse web frames
- transverse frames
- longitudinals on bottoms
- plates of bottoms and side hulls
- ship hull panels
- longitudinal girders and stringers

The normalized form of the constraint is expressed as

$$g(X) = 1 - \frac{\sigma_{\max}}{\sigma_p} \geq 0 \quad (8)$$

where $\sigma_{\max}$ is the maximum bending stress in a structural member and σ_p is the allowable bending stress.

Fig. 2 Design variables

(2) Shear strength

It is considered for

- transverse web frames
- longitudinal girders and stringers
- ship hull panels

Similar to the constraint of bending strength, this constraint is written as

$$g(\mathbf{X}) = 1 - \frac{\tau_{\max}}{\tau_p} \geq 0 \quad (9)$$

where $\tau_{\max}$ is the maximum shear stress in a structural member and τ_p is the allowable shear stress.

(3) Buckling strength

The buckling strength is needed to consider for

- longitudinals on bottoms
- plates of bottoms and side hulls

The buckling strength for these members can be expressed as

$$g(\mathbf{X}) = 1 - \frac{\sigma_{co}}{\sigma_{cr}} \geq 0 \quad (10)$$

where σ_{co} is the assuming applied buckling stress and σ_{cr} is the critical buckling stress of structural member. To evaluate elastoplastic buckling stress, Johnson's formula is applied with buckling stress and yield stress.

(4) Allowable maximum horizontal deflection of hatch opening

In this study, these kinds of constraints are evaluated by two structural analysis processes which are for

- transverse strength
- longitudinal bending and torsional strength

These constraints are described as:

$$g(\mathbf{X}) = 1 - \frac{d_{\max}}{d_a} \geq 0 \quad (11)$$

where $d_{\max}$ is the maximum horizontal deflection of hatch opening and d_a is the allowable horizontal deflection of hatch opening.

(5) Required midship sectional modulus

This constraint is written as

$$g(\mathbf{X}) = 1 - \frac{W_{\min}}{Z_{\min}} \geq 0 \quad (12)$$

where $W_{\min}$ is the required midship sectional modulus evaluated by NK rule⁵⁾ and $Z_{\min}$ is the minimum midship sectional modulus.

It needs to be noted that the upper and lower bounds for design variables are not taken as design constraints in the genetic algorithm. They are served as the upper and lower limiting values to treat in the binary substring of each variable.

3.4 Structural Analyses

In the above-mentioned design constraints, many factors such as deflections, stresses and forces are evaluated by structural analyses. Two processes of structural analyses are carried out for the cargo ship.

(1) Structural analysis for transverse strength

Three-dimensional structural analysis for transverse strength is implemented by transfer matrix method. The structure in the hold length with large hatch opening is simplified as the N segments divided by the transverse web frames, and one segment length corre-

Fig. 3 Load conditions

sponds to one transverse web frame space. Both fore and aft ends of the hold bounded by the transverse bulkheads are assumed to be fixed. The main longitudinal members are considered as the bottom girder, the side stringers and the deck girders.

(2) Structural analysis for longitudinal bending and torsional strength

For the longitudinal bending and torsional strength, the structural members are divided into the port hull panel, the starboard hull panel and the bottom hull panel. For each ship hull panel, bending moments, shear forces and axial forces in the panel plane are considered. Moreover, the shear flows are taken into account in the joint part between two ship hull panels. Similar to the structural analysis of transverse strength, both fore and aft ends of the hold bounded by the transverse bulkheads are assumed to be fixed. However, the shear deflections of transverse bulkheads at the fore and aft ends are taken into consideration.

Both above-mentioned structural analyses were in detail described in Ref. 6).

(3) Load cases

Two load cases shown in Fig. 3 are selected for calculation. One is the hogging condition for the transverse strength analysis. The other is the load condition on the oblique wave for the longitudinal bending and torsional strength. Both load cases are evaluated on the basis of NK rules⁵⁾.

4. Numerical Analyses and Discussions

A 1,750 dwt cargo ship with a large hatch opening as same as that of Ref. 6) is taken for the numerical analyses. Its principal dimensions are:

Length B. P. = 70 m; Breadth (mld.) = 12.5 m;
Depth (mld.) = 6.8 m; Draught = 3.7 m;
Deadweight = 1750 tonf.

The allowable elastic stresses for the structural members are listed in Table 1. The other allowable stresses are considered as follows: the yield stress = 2800 kgf/cm² and the assuming applied buckling stress = 2000 kgf/cm². The allowable horizontal deflection of hatch opening is taken as 12.0 mm. The number of transverse web frame spaces within the hold length for the actual ship is sixteen.

For the numerical example, all design variables are assumed to have the regular discrete values with a step

Table 1 Allowable elastic stresses for structural members

Structural Member	Bending Stress (kgf/mm ²)	Shear Stress (kgf/mm ²)
Transverse web frames	18.0	8.5
Longitudinal girders	10.0	5.5
Longitudinals	10.0	---
Transverse frames	10.0	---
Side hull & bottom plates	20.0	---
Ship hull panels	18.0	8.5

Table 2 Design variable values of cargo ship with large hatch opening

Design Variable	Minimum (cm)	Maximum (cm)	Step (cm)	Substring Length
X ₁ : thickness of bottom plates	1.00	1.75	0.05	4
X ₂ : web depth of longitudinals on bottoms	10.00	17.50	0.50	4
X ₃ : web depth of transverse frames	10.00	17.50	0.50	4
X ₄ : depth of upper side trans. web frames	45.00	60.00	1.00	4
X ₅ : thickness of deck girder plate	1.00	1.75	0.05	4
X ₆ : thickness of side stringer plate	1.00	1.75	0.05	4
X ₇ : longl. space of longitudinals on bottoms	55.00	86.00	1.00	5
X ₈ : thickness of side hull plates	1.00	1.75	0.05	4
X ₉ : flange width of upper side trans. web frames	10.00	17.50	0.50	4
Multivariable String Length				37

between their minimum and maximum values. Their detail values including minimum, maximum, step and corresponding binary substring length are listed in Table 2. By simple concatenating each single variable substring, a multivariable string of thirty-seven-bit length is constructed. It is worth noticing that the present methods can also treat discrete design variables without the above-mentioned structure. In such a case, it is necessary to supply an additional list of discrete values that the design variables can take.

The maximum given number of generation for all numerical cases is assigned to be one hundred.

4.1 Effect of Penalty Coefficient

The effect of penalty coefficient c_p on the optimum design is first investigated. All cases are executed under the following genetic operating parameters: crossover probability, $p_c=0.8$, mutation probability, $p_m=0.0003$, and population size, $n_p=50$. For different problem, there may be a different penalty coefficient suitable for it. In this study, nine cases with $c_p=0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0$ and 10.0 are computed. The results are plotted in Fig. 4. The best solution corresponds to the case that $c_p=0.05$. In the following numerical cases, the genetic operating parameters will keep as the same as those taken in the best solution case unless there is a specific statement to them.

The generation history for the above best solution

Fig. 4 Cost optimum values varied with penalty coefficient c_p

Fig. 5 Generation history of cost value and fitness value

case is shown in Fig. 5. As the average fitness value increases and improves from generation to generation, the total cost value (that is, the objective function value) gradually converges to an optimum or near-optimum solution. With the multivariable string coding illustrated in Table 2, the number of possible design variable combinations is 2×16^9 ($\cong 1.4 \times 10^{11}$). In the best solution case, the near optimum solution is obtained after about 50 generations that represent 2500 function evaluations. Obviously, the genetic algorithm can search an optimum or near-optimum solution utilizing only a very small fraction of candidate solution space. It demonstrates the genetic algorithm is an effective and robust search method.

4.2 Analyses of Genetic Operating Parameters

In this section, the influence of genetic operating parameters on the optimization solution is analyzed. Those genetic operating parameters include population size, crossover probability and mutation probability.

(1) Population size n_p

The population size essentially determines the number of solutions that are processed for each generation.

The larger the population size is, the better the chances of obtaining an optimum solution are. However, the computing time spends more with larger population size. Thus, there is a need to find an appropriate population size that can lead to a good result but the computing time is not too long.

Population sizes ranging from 10 to 90 are used to investigate the effect of generation size on the optimization solution. For each population size, three tests are performed in order to avoid the contingency. Then the average value of three optimum cost values for each population size is calculated. The relative percentage of optimum cost is further computed by an expression, $(\bar{C}_{\max} - \bar{C})/(\bar{C}_{\max} - \bar{C}_{\min}) \times 100\%$. Here, $\bar{C}$ is the average value of optimum cost for each population size, $\bar{C}_{\max}$ is the maximum value in all average values of optimum cost and $\bar{C}_{\min}$ is the minimum one.

Obviously, the solution with a higher percentage value is a better one. The percentage value against the population size is plotted in Fig. 6. It can be seen that better solutions are obtained with larger population size as expected. It is difficult to recommend a population size for all optimization problems, as the appropriate population size may vary depending on the peculiarities of problem. In the present study, it appears that the moderate population size is about 50 since little improvement is observed with larger population size over 50.

(2) Crossover probability p_c

Five trials with $p_c = 0.6, 0.7, 0.8, 0.9$ and 1.0 are carried out. The generation history of total cost value with different crossover probability is plotted in Fig. 7. It has been found that the total cost value converges prematurely with smaller crossover probability, such as the

Fig. 6 Influence of population size

Fig. 7 Generation history of total cost value with different crossover probability p_c

cases of $p_c = 0.6$ and 0.7 . The final total cost values in three cases with crossover probability $0.8, 0.9$ and 1.0 are almost the same, but the latter two cases converge slowly. It means that smaller crossover probability may result in a premature solution but larger crossover probability may cause the solution to converge slowly. Therefore, the crossover should be done with a proper probability in the genetic algorithm. In the present study, it has been found that $p_c = 0.8$ is suitable for the optimization problem.

(3) Mutation probability p_m

The numerical cases in which the mutation probability changes by $0.1, 0.03, 0.003, 0.0003$ and 0.00003 are computed. The results are shown in Fig. 8. With larger mutation probabilities, such as the cases of $p_m = 0.1$ and 0.03 , the solution changes violently and unsteadily. It is difficult to get the convergent result. In general, a lower mutation probability gives better solution for the problem. However, the premature may happen if the mutation probability is too small, such as the numerical case with $p_m = 0.00003$. In this study, it is seen that the total cost value in the case of $p_m = 0.0003$ converges to a good solution steadily. Thus, the mutation probability should be proper small in the genetic algorithm. As

Fig. 8 Generation history of total cost value with different mutation probability p_m

mutation probability is very small, mutation is appropriately considered as a secondary mechanism in the genetic algorithm. When used sparingly with reproduction and crossover, it is an insurance policy against a premature loss of important genetic material in the genetic algorithm.

4.3 Comparison with Multiplier Method

The same numerical example was investigated by multiplier method in Ref. 6). This makes it possible to compare the present method with the multiplier method. The comparison of optimum results between two methods is illustrated in Table 3. The following observations are made through the comparison of the genetic algorithm and the multiplier method:

(1) The genetic algorithm can treat the discrete optimization problem well only if we code the discrete variables into a corresponding binary string of finite length. Owing to the continuity assumption, the optimum results by the multiplier method may be unfeasible because they may not match with the structural elements available commercially. If the nearest feasible discrete values are assigned to the design variables, the total cost value is likely to increase, as shown in Table 3. The present method based on the genetic algorithm

Table 3 Comparison of optimum results between genetic algorithm and multiplier method

Comparison Item		Actual Ship		Multiplier Method	Genetic Algorithm
Design	X_1	1.200	1.185	1.200	1.400
	X_2	12.500	14.644	14.700	17.500
	X_3	12.500	12.373	12.400	14.000
Variable	X_4	50.000	54.931	55.000	59.000
	X_5	1.400	0.981	1.000	1.000
	X_6	1.000	0.998	1.000	1.050
Values (cm)	X_7	65.625	65.632	65.000	76.000
	X_8	1.200	1.064	1.100	1.050
	X_9	15.000	12.159	12.500	13.000
Cost Values (10^3 yen)	Material	24,991	23,483	24,188	24,888
	Welding	24,783	23,046	24,686	21,311
	Total	49,774	46,529	48,875	46,199
Iteration Number of Structural Reanalysis		883		2,500	

is more rational in view of the discrete nature of practical optimization problems.

(2) The genetic algorithm does not require gradient computations. Hence, problems where gradient computations are difficult can be solved by the genetic algorithm very well. Though the multiplier method can utilize some algorithms to minimize a function without calculating derivatives, the numerical calculation for the line search making use of some properties of conjugate directions could not be avoided.

(3) The major computational tasks in the genetic algorithm are the function evaluations and analyses. Although the gradient computations are absent, the genetic algorithm is slower compared to the multiplier method. It is proved to be true in this study. From the results shown in Table 3, the iteration number of structural reanalysis in the genetic algorithm is about 2500, while that in the multiplier method is less than 1000. Since the computing time is mainly spent in the structural reanalysis for the numerical example, the genetic algorithm is obviously slower than the multiplier method.

(4) The global convergence of the multiplier method has been mathematically proved. An algorithm is said to be globally convergent if it converges to a solution from any starting point. It should not be confused with the problem of finding a global optimum. In the genetic algorithm, the schema theorem gives a mathematical basis for the gain or loss of schema in successive generations. In other words, it proves that the overall fitness can be improved from generation to generation in the genetic algorithm. However, it cannot be mathematically shown that the algorithm converges to a definite optimum. Since many optimization problems solved by the genetic algorithm give better results compared to the results of traditional methods, the genetic algorithm is acceptable for engineering design. From the results of Table 3, the results of the genetic algorithm are better than those of the multiplier method. The major reason is that the multiplier method is a point-to-point search method. Along with the

approximation of the numerical algorithm, it may get trapped in a local optimum. By contrast, the genetic algorithm works from a population of points simultaneously, thus, the chance of finding an optimum or near-optimum solution is increased. From the comparison results in Table 3, the multiplier method finds a solution depending on the initial design but the genetic algorithm may search a better solution at random from the whole space of candidate solutions.

(5) Unlike the multiplier method, the upper and lower bounds for variables are not taken as design constraints in the genetic algorithm. They are served as the upper and lower limiting values to treat in the binary substring of each variable.

5. Conclusion Remarks

Using the proposed method, the numerical example of a cargo ship with large hatch opening is investigated for the purpose of illustration. It demonstrates that the genetic algorithm is an effective and robust search method for structural optimization. If the discrete design variables are correspondingly coded into a binary string of finite length, the present method handles the discrete optimization problems efficiently. As most design variables in practical structural optimization problems are discrete in nature, this ability makes the genetic algorithm a good choice for structural optimization problems.

Though the genetic algorithm is slower compared to the multiplier method, it may search a better solution as a population processing method. On the other hand, the genetic algorithm does not require gradient computations. This is another important reason why the genetic algorithm is broadly applied to engineering optimization.

For different optimization problems, there may be a

different set of parameters including the penalty coefficient, c_p , the generation size, n_p , the crossover probability, p_c , and the mutation probability, p_m , to be suitable for it. It is necessary to analyze the influence of those parameters on the optimum design if the present method is applied.

Through the numerical analyses, it shows the power of the genetic algorithm for the discrete optimization problems. There is further research scope to improve the search efficiency of the genetic algorithm by using other genetic techniques, such as advance genetic operators and more efficient search strategies.

References

- 1) Arora, J. S.: Introduction to Optimum Design, McGraw-Hill, New York, 1989.
- 2) Dhingra, A. K., and Lee, B. H.: "A Genetic Algorithm Approach to Single and Multiobjective Structural Optimization with Discrete-Continuous Variables", *Int. J. Numer. Methods Eng.*, Vol. 37, 1994, pp. 4059-4080.
- 3) Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass., 1989.
- 4) Holland, J. H.: Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, Mich., 1975.
- 5) Nippon Kaiji Kyokai: Rules and Regulations for the Construction and Classification of Ships, Amendment, 1994 (in Japanese).
- 6) Nobukawa, H., Zhou, G., Kitamura, M., and Kobayashi, S.: "Structural Optimization for Small Container Ship with Large Hatch Opening", *Trans. West Japan Soc. of Naval Arch.*, Vol. 90, 1995, pp. 181-197.
- 7) Okada, T., and Neki, I.: "Optimization of Ship Structural Design by Genetic Algorithm", *J. Soc. Naval Arch. of Japan*, Vol. 171, 1992, pp. 259-266 (in Japanese).