Journal of
Ship&Ocean
Technology
©DECEMBER 2005

SOTECH Vol. 9, No. 4, pp. 35 ~ 46, 2005

Real-coded Micro-Genetic Algorithm for Nonlinear Constrained Engineering Designs

Yunyoung Kim[1], Byeong-Il Kim[1], Sung-Chul Shin[1]

[1] Dept. of Naval Architecture and Ocean Engineering, Mokpo National Maritime University, Jeonnam, Korea; E-mail: yunyoungkim@gmail.com

Abstract

The performance of optimisation methods, based on penalty functions, is highly problem-dependent and many methods require additional tuning of some variables. This additional tuning is the influences of penalty coefficient, which depend strongly on the degree of constraint violation. Moreover, Binary-coded Genetic Algorithm (BGA) meets certain difficulties when dealing with continuous and/or discrete search spaces with large dimensions. With the above reasons, Real-coded Micro-Genetic Algorithm (RμGA) is proposed to find the global optimum of continuous and/or discrete nonlinear constrained engineering problems without handling any of penalty functions. RμGA can help in avoiding the premature convergence and search for global solution-spaces, because of its wide spread applicability, global perspective and inherent parallelism. The proposed RμGA approach has been demonstrated by solving three different engineering design problems. From the simulation results, it has been concluded that RμGA is an effective global optimisation tool for solving continuous and/or discrete nonlinear constrained real-world optimisation problems.

Keywords: real-coded micro-genetic algorithm, binary-coded genetic algorithm, penalty function, global optimisation

1 Introduction

In the traditional optimisation methods, each of them is specialised in solving a particular type of problem. When faced with a different type of problem, the same method may not work as well. In general, it has known that an effective way to solve the nonlinear constrained optimisation problems is to transform it into a sequence of unconstrained minimisation. Several methods, which are based on penalty functions, have been proposed for handling nonlinear/linear constraints by genetic algorithm for numerical optimisation problems. The performance of these methods is highly problem-dependent and many methods require additional tuning of some variables. This additional tuning is the influences of penalty coefficient, which depend strongly on the degree of constraint violation(Koziel and Michalewicz, 1999). Binary-coded Genetic Algorithm (BGA) meets certain difficulties when dealing with continuous and/or discrete search spaces with large dimensions(Herrera et al, 1998).

With the above reasons, Real-coded Micro-Genetic Algorithm (RμGA) is proposed to find the global optimum of continuous and/or discrete nonlinear constrained engineering problems without handling any of penalty functions. Micro-Genetic Algorithm (μGA)

explores in a small population with some genetic operators to find the global optimum solution-spaces(Kim et al, 2004; Kim et al, 2005). The proposed approach has the robustness of parallel exploration and asymptotic convergence with real value parameters. Therefore, RμGA can help in avoiding the premature convergence and search for better global solution, because of its wide spread applicability, global perspective and inherent parallelism. The proposed RμGA approach has been demonstrated by solving three different engineering design problems.

From the simulation results, it is shown that RμGA implementation overcomes the poor convergence properties and finds the global optimum solution than results obtained from other optimisation methods. Therefore, it has been concluded that the proposed RμGA is an effective optimisation tool for solving continuous and/or discrete nonlinear real-world optimisation problems.

2 Real-coded micro-genetic algorithm (RμGA)

Most real-world problems may not be handled using binary representations and an operator set consisting only of binary crossover and binary mutation in Genetic Algorithm (Davis, 1989). The reason is that nearly every real-world domain has associated domain knowledge that is of use when one is considering a transformation of a solution in the domain. Davis(1989) believes that the real-world knowledge should be incorporated into the GA, by adding it to the decoding process or expanding the operator set. Real coding allows the domain knowledge to be easily integrated into Real-coded Genetic Algorithm (RGA) for the case of problems with non-trivial restrictions.

RGA is one of the optimisation methods with multi-point approaches. A solution is directly represented as a vector of real-parameter decision variables. Starting with a population of such solutions (usually randomly created), a set of genetic operators (such as crossover and mutation) is performed to create a new population in an iterative manner. Although most RGA differs from each other mainly in terms of their crossover and mutation operators, they mostly follow one of a few algorithmic models.

RμGA explores in a small population with multiple genetic operators to find the global optimum solution-spaces. RμGA offers the advantage that the continuous parameters can gradually adapt to the fitness landscape over the entire search space whereas parameter values in binary implementations are limited to a certain interval and resolution. RμGA blurs the distinction among genotype and phenotype, since in many problems the real number vector already embodies a solution in a natural way. A highlighted advantage of the RμGA is the capacity for the local tuning of the solutions. For example, Legendre-Gauss mutation allows the tuning to be produced in a more suitable and faster way than in the BGA, where the tuning is difficult because of the Hamming cliff effect. Moreover, RμGA is a steady-state, elite-preserving, and computationally fast algorithm for creating offspring near parents than anywhere in the search-space. The main skeleton of the proposed RμGA, based on an idea of Michalewicz(1994), is illustrated in Figure 1.

RμGA finds the global optimum solution by maintaining two types of population as follows: search population and steering population (Kim et al, 2005). Initial populations, consisting of five chromosomes, are generated in a random fashion to serve as the starting feasible solution-spaces. The populations, satisfying the engineering design constraints, serve as a reservoir of information about the environment and as a basis for generating new trials. The search population, which satisfies linear constraints of the problem, is a population for searching the solution-spaces in each generation. A development of the

search population influences evaluations of individuals in the steering population, satisfying all constraints.

At each generation step, a feasible search-space is searched by making steering points from the search points. Some steering points are moved into the population of search points, where they undergo transformation by specialised operators. That is, the fitter chromosome is selected to produce offspring, which inherit the best characteristics of the parents, for the next generation step. RμGA terminates the optimisation procedure when a pre-specified number of generations is elapsed. Then, the result is hopefully a population that is substantially fitter than the original.

In the reproductive plan of RμGA, the main challenge is how to use a pair of decision variable vectors to create a new pair of offspring vectors or how to perturb a decision variable vector to a mutated vector in a meaningful manner. That is, the most important skill is how to make a reproductive plan for better searching technique due to the small population size. Therefore, multiple genetic operators are adopted for exploration of new solution-spaces, and will be discussed in the following section 3.

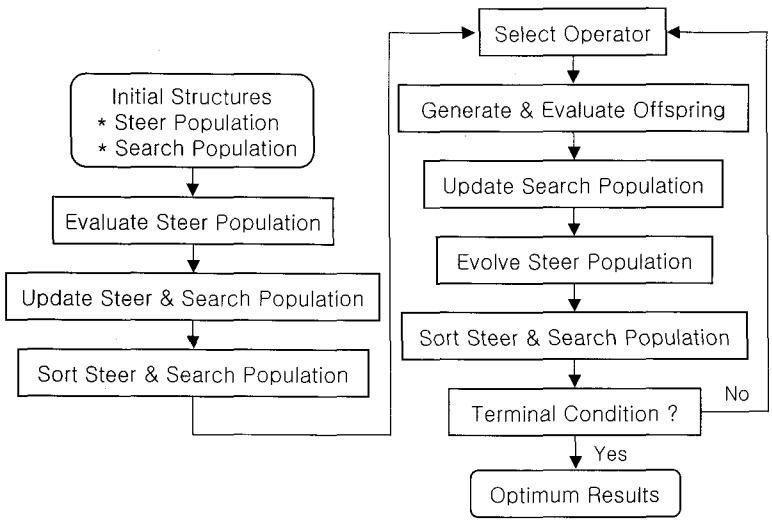

Figure 1: Skeleton of the proposed RμGA

3 Genetic operators

As a result of the modified genetic representation, RμGA requires customisation of the standard mutation and crossover operators. In general, the distinction between a crossover and a mutation operator lies mainly in the number of parent solutions used in the perturbation process. If only one parent solution is used, it should be called a mutation operator.

Since most crossover operators do not use any fitness information for global optimum, there is no reason for a crossover operator to steer the search in any particular direction. The best that a crossover operator can do is to keep the mean of the offspring population the same as that of the parent population and alter the population variance. The crossover operator creates each decision variable of offspring vectors from one or more parent vectors at a time with a probability. A gene with real value is mutated in a dynamic range.

The proposed RμGA consists of 5 types of genetic operators: heuristic crossover, arithmetic crossover, boundary mutation, metropolis mutation and legendre-gauss mutation

For the convenience of understanding of the following algorithms, some definitions are delineated as follows:

Parents	: $P^k = (x_1^k, \cdots, x_i^k, \cdots, x_n^k),\ k = 1,2$
Offspring	: $O^k = (v_1^k, \cdots, v_i^k, \cdots, v_n^k),\ k = 1,2$
λ	: Random number drawn uniformly from the interval [0,1],
τ	: Unbiased coin flip with a value of zero or one,
P	: Parameter for parent solution,
O	: Parameter for offspring solution,
x_i	: i-th design variable in parent solution,
v_i	: i-th design variable in offspring solution,
k	: Index for selected solution.

3.1 Arithmetic crossover operator

Arithmetic crossover (Michalewicz and Janikow, 1991) generates an offspring O^1 by $v_i^1 = \lambda x_i^1 + (1-\lambda)x_i^2$ from parents P^k. The random number λ is chosen by carefully calculating its maximum allowed value in all decision variables so that the resulting offspring does not exceed the lower or upper limits (domain constraints). This operator explores points in the search-space which belong to line connecting its parent, P^k.

3.2 Heuristic crossover operator

This operator, introduced by Wright(1991), is a unique crossover for the following reasons: (1) it uses values of the objective function in determining the direction of the search, (2) it produces only one offspring, and (3) it may produce no offspring at all. The operator generates a single offspring O^1 by $v_i^1 = \lambda\left(x_i^2 - x_i^1\right) + x_i^2$ from parents P^k. The parent P^2 is not worst than P^1, i.e., $f\left(x_i^2\right) \leq f\left(x_i^1\right)$ for minimisation problems and vice-versa for maximisation problems. It is possible for this operator to generate an offspring vector which is not feasible.

3.3 Legendre-gaussian mutation operator

This mutation, introduced by the current authors, consists in adding a random value from a Legendre-Gauss integral distribution to each element of an individual's vector to create a new offspring. This operator is based on the Legendre-Gauss integral formula with a specified number of abscissae over the interval [-1,1]. If a vector x_i^1 is selected for this mutation from the set of movable vectors, the resulting offspring is

$$v_i^1 = \frac{1}{2m}\sum_{j=1}^{m}\left\{\left(U_i - L_i\right)w_j + \left(U_i + L_i\right)\right\} \qquad (1)$$

where U_i and L_i are the newly specified limits (upper and lower, respectively) of the i-th vector at each generation step. The limits are generated by $U_i = x_i^1 + \lambda_1 x_i^1$ and $L_i = x_i^1 - \lambda_2 x_i^1$. w_j is the weigh of the designated evaluation point in the sum. We adopted 10 points Legendre-Gaussian integration. Therefore, m is 10 as the designated evaluation points. λ_1 and λ_2 are random numbers drawn uniformly from the interval [0,1]. This mutation is one of the operators responsible for the fine tuning capabilities of the system.

3.4 Metropolis mutation operator

The idea of this operator, suggested by the current authors, is to use Simulated Annealing (SA) to achieve a fast convergence in search-spaces with feasible solutions. Kirkpatrick et al(1983) took the idea of the Metropolis algorithm and applied it to combinatorial optimisation problems. SA is a technique that has attracted significant attention, being suitable for optimisation problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema. That is, SA introduces a more sophisticated way of moving from the current solution to one of its neighbours, accepting with a certain probability to move also when the quality of the new solution is worst than the previous one. Metropolis mutation generates an offspring O^1 from a parent P^1 by

$$v_i^1 = \begin{cases} x_i^1 + \lambda(x_i^U - x_i^1), & \text{if } \tau = 1 \\ x_i^L + \lambda(x_i^1 - x_i^L), & \text{if } \tau = 0 \end{cases} \tag{2}$$

Consider a huge number of particles of fixed volume at some temperature T. Since the particles move, the system can be in various states. The probability that the system is in a state of certain energy E is given by the Boltzmann distribution $\text{Prob}(E) \approx \exp(-E/\kappa T)$. The energy state E is the fluctuation of the objective function value between an offspring and a parent in the optimisation process. The quantity κ (Boltzmann's constant) is a constant of nature that relates temperature to energy.

SA sometimes undergoes uphill search as well as downhill search. The simulation in the Metropolis algorithm calculates the new energy of the system. If the energy has decreased then an offspring is selected for next generation step. If the energy has increased then an offspring is accepted using the probability returned by the Boltzmann distribution.

3.5 Boundary mutation operator

The diversity of the population should be kept to be at certain minimal level during computation. This brings a negative effect of convergence deceleration that could be oppressive within high problem dimensions. This is why we have introduced boundary mutate operator (Michalewicz and Janikow, 1991). The aim of this mutation is to grub the near neighbourhood of better chromosomes and so make searching for more and more precious solution faster. Therefore, the mutation is very good operator especially when best chromosomes have to contain vector with values near their boundary values. Boundary mutation works like uniform mutation, but replaces the value of the chosen vector with either the upper or lower bound for that vector (chosen randomly).

Let P^1 be a parent determined for mutation. If the i-th component is the selected vector, the resulting offspring is $O^1 = (v_1^1, \cdots, v_i^1, \cdots, v_n^1)$, where v_i^1 is either L_i or U_i with equal probability (L_i and U_i denote the new lower and new upper bound, respectively). The dynamic values (L_i and U_i) are easily calculated from the set of constraints.

4 Continuous/discrete nonlinear engineering designs

Three different problems are selected and demonstrated for the superior performance of RμGA. The proposed RμGA is performed on a personal computer with 3.2GHz Pentium CPU and 1GB RAM.

4.1 Helical compression spring design

In a helical compression spring design with the minimum volume, three design variables are used: the number of coils N (integer variable), the wire diameter d (discrete variable), and the mean coil diameter D (continuous). The possible wire diameters are 42 non-equispaced values for ASTM A228 material. The detail discussion of this design model and its solution was fully described in some of literatures (Kannan and Kramer, 1994, Deb and Goyal 1997). These three variables are redefined in terms of the design vector X; $X=[x_1,x_2,x_3]^T=[N,d,D]^T$. The objective function and constraints are:

$$\text{Minimise } F = 0.25\pi^2 x_2^2 x_3 (x_1+2) \tag{3}$$

With

$$g_1 = S - \frac{8KP_{\max}x_3}{\pi x_2^3} \geq 0 \qquad g_2 = l_{\max} - \frac{P_{\max}}{\kappa} - 1.05(x_1+2)x_2 \geq 0$$

$$g_3 = x_2 - d_{\min} \geq 0 \qquad g_4 = D_{\max} - (x_2+x_3) \geq 0$$

$$g_5 = D/d - 3 \geq 0 \qquad g_6 = \delta_{pm} - \delta_p \geq 0$$

$$g_7 = \frac{P_{\max}-P}{\kappa} - \delta_w \geq 0$$

The parameters used above are:

$$K = \frac{4x_3 - x_2}{4(x_3-x_2)} + \frac{0.615x_2}{x_3} \qquad P = 300\text{ lb} \qquad P_{\max} = 1000\text{ lb}$$

$$\kappa = \frac{Gx_2^4}{8x_1x_3^3} \qquad G = 11.5\text{ Mpsi} \qquad l_{\max} = 14\text{ in}$$

$$\delta_P = \frac{P}{\kappa} \qquad \delta_w = 1.25\text{ in} \qquad \delta_{pm} = 6\text{ in}$$

$$S = 0.189\text{ Mpsi} \qquad D_{\max} = 3\text{ in} \qquad d_{\min} = 0.2\text{ in}$$

The variable N is coded as an integer variable between 1 to 32. The wire diameter d is coded as a discrete variables. The variable D takes any real value in subsequent iterations. The solution obtained by RμGA has outperformed than previous optimal solutions reported as shown in Table 1. This problem was solved earlier using a Genetic Adaptive Search (GeneAS) method (Deb and Goyal, 1997) and the Branch and Bound (BB) method (Sandgren 1988). The total number of trial iterations is 3.0×10^4 in calculation of RμGA. The calculating times are about 3.1 seconds. Figure 2 shows the evolution process as the best-so-far fitness of the objective function.

Table 1: Optimal solutions of spring design

	N	d (in)	D (in)	F
RμGA	9	0.283	1.223	2.658
GeneAS	9	0.283	1.226	2.665
BB	10	0.283	1.180	2.798

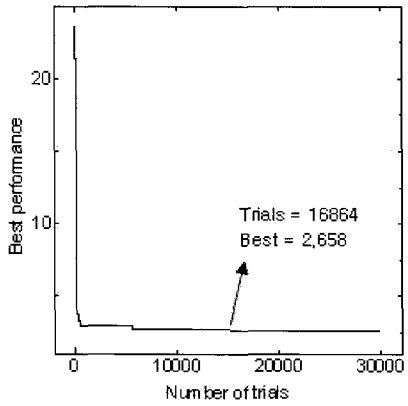

Figure 2: Evolution process (spring design)

4.2 Welded beam design

A welded structure is shown in Figure 3. A beam *A* is welded to a rigid support member *B*. The welded structure is to support a force *P* of 6,000 lb. The length *L* is assumed to be specified at 14 inch. The objective is to find a feasible combination of design variables so that the cost function is minimised. The design variables in this design model are: leg length (*h*), weld length (*w*), beam thickness (*t*) and beam breadth (*b*). For notational convenience we redefine these four variables in terms of the design vector X:

$$X = [x_1, x_2, x_3, x_4]^T = [h, w, t, b]^T$$

The objective function consists of two major cost components: welding labour cost and material cost. The objective function is

$$\text{Minimise } F = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (L + x_2) \qquad (4)$$

There are several functional relationships between the design variables which delimit the region of feasibility. These relationships, expressed in the form of inequalities, represent the design model. The inequality constraints and boundary conditions are

$$g_1 = \tau_d - \tau \geq 0 \qquad g_2 = \sigma_d - \sigma \geq 0 \qquad g_3 = x_4 - x_1 \geq 0$$
$$g_4 = P_c - P \geq 0 \qquad g_5 = 0.25 - \delta \geq 0$$
$$0.125 \leq x_1 \leq 10 \qquad 0.1 \leq x_2, x_3, x_4 \leq 10$$

The design shear stress (τ_d) of weld and the design normal stress (σ_d) for beam material are adopted to be 13,600 psi and 30,000 psi in this design model, respectively. The detail discussion of this design model and its solution was fully described in Ragsdell and Phillips(1976) and Deb(1991). They solved this problem using Geometric Programming(GP) and Genetic Algorithms(GA) to minimise cost function. The same parameters used in this design model are adopted for the proposed RμGA to compare the optimal simulation results. The simulation results are listed in Table 2. In computations of RμGA, the total number of trial iterations is 3.0×10^4. The CPU running times are about 1.2 seconds. Figure 4 shows the evolution process as the best-so-far fitness of the objective function.

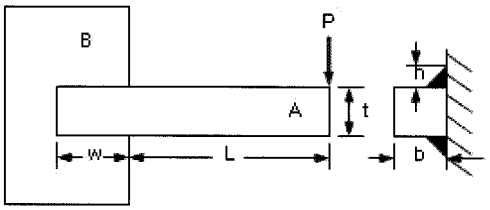

Figure 3: Weld beam model

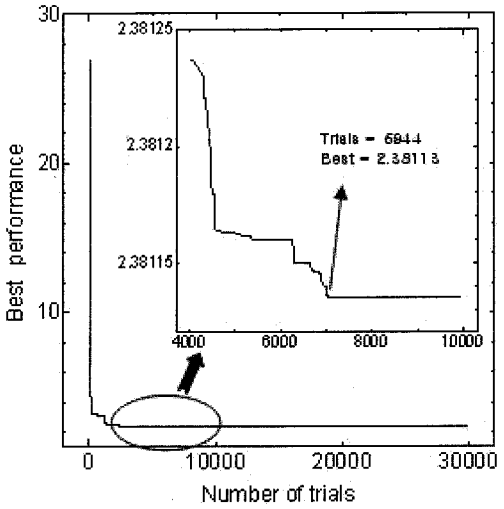

Figure 4: Evolution process (Welded beam)

Table 2: Simulation results of a weld beam model

	x_1 (in)	x_2 (in)	x_3 (in)	x_4 (in)	F
RμGA	0.2444	6.2186	8.2915	0.2444	2.38113
GP	0.2455	6.1960	8.2730	0.2455	2.38594
GA	0.2489	6.1730	8.1789	0.2533	2.43312

4.3 Hatchcoverless container ship

The optimum midship section design of container ship belongs to the nonlinear constrained optimisation problem. In order to compare the effectiveness of the proposed RμGA, two deterministic search methods were compared on the actual hatchcoverless container ship (Choi 1993) by optimising the midship section design. These methods are Hooke & Jeeves Pattern (HJP) search method and Nelder & Mead Simplex (NMS) search method. The Rules and Regulations of Lloyd's Register of shipping are adopted (LR 2002). The objective function is the total sectional area of all the longitudinal members for the midship section.

$$\text{Minimise } F = Td \times (Wd + Ws) + \sum_{i=1}^{NP} (WP_i \times TP_i) + \sum_{j=1}^{NL} AL_j + Hb \sum_{k=1}^{NS} TS_k + (0.5 \times TC \times Hb) \quad (5)$$

where,

Td :	Thickness of deck plate	Wd :	Width of deck plate
Ws :	Width of sheerstrake	WP_i :	Width of *i-th* plate (except deck plate)
TP_i :	Thickness of *i-th* plate (except deck plate)	NP :	Number of plates (except deck plate)
NL :	Number of longitudinal members	AL_j :	Sectional area of *j-th* longitudinal members
NS :	Number of side girders	TS_k :	Plate thickness of *k-th* side girder
Hb :	Double bottom height	TC :	Plate thickness of C.L. girder

Design variables are the space and geometric beam configurations (thickness and height) of longitudinal stiffeners, and illustrated in Figure 5. The constraints are mentioned as follows:

$$g_j \geq 0 \quad (j = 1,2,\cdots,15)$$

where,

$$g_{1-4} = \frac{X_i}{S_{\min}} - 1 \ (i = 1,6,7,8), \quad g_{5-8} = 1 - \frac{X_i}{S_{\max}} \ (i = 1,4,7,8), \quad g_9 = X_2 \times 1.8 \times \frac{\zeta}{X_3} - 1$$

$$g_{10} = 1.05 - X_2 \times 1.8 \times \frac{\zeta}{X_3}, \quad g_{11} = 1 - \frac{Z_{LR}}{Z_{LA}}, \quad g_{12} = \frac{1}{F_B} - 1$$

$$g_{13} = \frac{1}{F_D} - 1, \quad g_{14} = \frac{Z_{AB}}{Z_{RB}} - 1, \quad g_{15} = \frac{Z_{AD}}{Z_{RD}} - 1$$

ζ : Higher tensile steel factor

$S_{\min}, S_{\max}$: Minimum and maximum spacing of longitudinal members, respectively

Z_{LR}, Z_{LA} : Rule required and actual deck longitudinal sectional modulus, respectively

F_B, F_D : Local scantling reduction factor for hull members below and above N.A., respectively

Z_{AB}, Z_{AD} : Actual midship sectional modulus at bottom and deck, respectively

Z_{RB}, Z_{RD} : Required minimum midship sectional modulus at bottom and deck, respectively

Table 3: Simulation results between actual ship and optimization methods

Variable	Actual ship	RμGA	NMS	HJP
X_1 [*mm*]	925.0	950.0	975.0	920.0
X_2 [*mm*]	26.0	34.0	35.5	33.5
X_3 [*mm*]	400.0	590.0	635.0	595.0
X_4 [*mm*]	849.0	849.0	849.0	849.0
X_5 [*mm*]	750.0	750.0	750.0	750.0
X_6 [*mm*]	850.0	765.0	780.0	800.0
X_7 [*mm*]	868.0	765.0	765.0	785.0
X_8 [*mm*]	868.0	765.0	775.0	780.0
Total area [cm^2]	30,529	29,678	30,028	30,049
Weight [*ton/m*]	23.965	23.340	23.572	23.604
Ratio [%]	100.0	97.392	98.360	98.494

From the simulation results of Table 3, the midship section design by using RμGA has a weight efficiency of 2.608% to the design of the actual ship. The probability of obtaining the global minimum for RμGA is measured by running 10,000 independent trial iterations. The CPU running times are about 15.2 seconds. Figure 5 shows the optimum midship

section design of 2000 TEU hatchcoverless container ship. Figure 6 shows the evolution process as the best-so-far fitness of the objective function

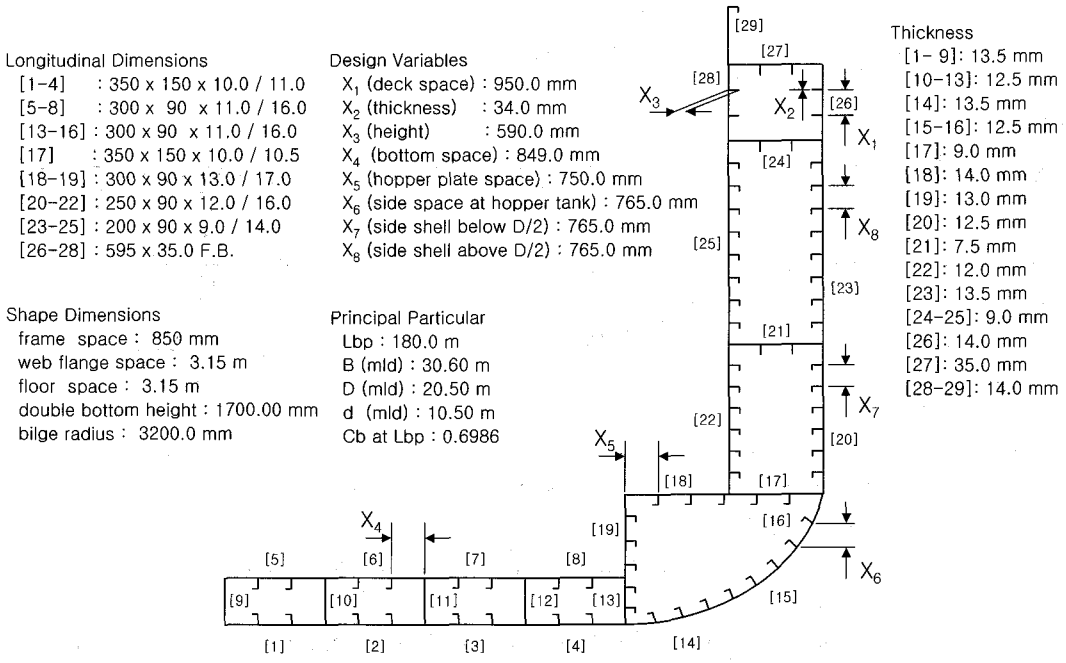

Figure 5: Optimum midship section design of 2000 TEU hatchcoverless container ship

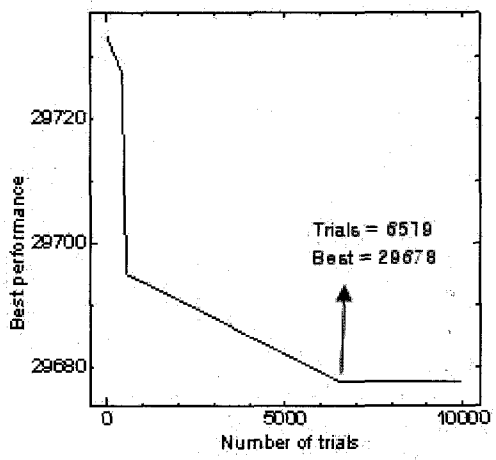

Figure 6: Evolution process (container ship)

5 Concluding remarks

A new approach, referred to as Real-coded Micro-Genetic Algorithm (RμGA), to solve continuous and/or discrete nonlinear optimisation problems is proposed and developed with the help of multiple genetic operators. RμGA approach is an abstraction of natural genetics and theoretical physics and is aimed to search the optimum solution space in

global optimisation problems. Therefore, RμGA can help in avoiding the premature convergence and search for better global solution, because of its wide spread applicability, global perspective and inherent parallelism.

From the simulation results of Tables 1-3 and Figures 2, 4 and 6, it was shown that RμGA implementation converged to global optimum solution with a marvellous explorability than the other optimisation methods in three different engineering design problems. Therefore, Real-coded Micro-Genetic Algorithm can be suggested as an useful tool for solving continuous and/or discrete nonlinear global optimisation problems.

Acknowledgements

The authors wish to acknowledge the financial support of NURI (New University for Regional Innovation). This support is gratefully acknowledged.

References

Choi, K.S. 1993. Introduction of Hanla 2000TEU open top container ship. J. of Society of Naval Architects of Korea, **30**, **1**, 6-12.

Davis, L. 1989. Adapting operator probabilities in genetic algorithms, Proc. 3rd International Conference on Genetic Algorithms. J. David Schaffer (Ed.), Morgan Kaufmann Publishers, 61-69.

Deb, K. 1991. Optimal Design of a Welded Beam via Genetic Algorithms. AIAA Journal, **29**, **11**, 2013-2015.

Deb, K. and Goyal, M. 1997. Optimizing engineering designs using a combined genetic search, In Thomas Back (Ed.). Proc. 7th International Conference on Genetic Algorithms, 521-528.

Herrera, F., M. Lozano and J.L. Verdegay. 1998. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artificial Intelligence Review, **12**, **4**, 265-319.

Kannan, B.K. and S.N. Kramer. 1994. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J. Mechanical Design, **116**, **2**, 405-411.

Kim, Y., K. Gotoh and M. Toyosada. 2004. Global cutting-path optimization considering the minimum heat effect with microgenetic algorithms. J. Marine Science and Technology, **9**, **2**, 70-79.

Kim, Y., K. Gotoh, K.S. Kim and M. Toyosada. 2005. Optimum grillage structure design under a worst point load using real-coded micro-genetic algorithm. Proc. 15th International Offshore and Polar Engineering Conference, Seoul, Korea, 730-736.

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi. 1983. Optimization by simulated annealing, science, **220**, **4598**, 671-680.

Koziel, S. and Z. Michalewicz. 1999. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimisation. Evolutionary Computation, **7**, **1**, 19-44.

Lloyd's Register of shipping; Rules and Regulations for the Classification of Ships. 2002. Part 3 and 4.

Michalewicz, Z. 1994. Genetic Algorithms + Data Structures = Evolution Programs. extended edition, Springer-Verlag, Berlin.

Michalewicz, Z. and C.Z. Janikow. 1991. Handling constraints in genetic algorithms, Proc. 4th International Conference on Genetic Algorithms, 151-157.

Ragsdell, K.M. and D.T. Phillips. 1976. Optimal Design of a Class of Welded Structures Using Geometric Programming. ASME: J. of Engineering for Industry, Ser. B, **98**, **3**, 1021-1025.

Sandgren, E. 1988. Nonlinear integer and discrete programming in mechanical design. Proc. ASME Design Technology Conference, Kissimee, FL, 95-105.

Wright, A.H. 1991. Genetic algorithms for real parameter optimisation. Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, 205-218.