Discussion on Distributed Genetic Algorithms for Designing Truss Structures

Yusuke Tanimura
tanisuke@mikilab.doshisha.ac.jp

Tomoyuki Hiroyasu
tomo@is.doshisha.ac.jp

Mitsunori Miki
mmiki@mail.doshisha.ac.jp

Doshisha University
Kyotanabe Kyoto, Japan

Abstract

Distributed Genetic Algorithm (DGA) is one of optimization methods and very suitable to parallel computers. Therefore, DGA is a very effective tool in high performance computing. In this paper, we discussed DGAs for designing truss structures. Most of the real world problems in optimization problems have constraints. Since DGA is an algorithm for problems without constraints, the fitness function of DGA should be modified to handling constraints in problems with constraints. In this paper, we proposed a new method of handling constraints for DGAs. At the same time, the extended model of DGA called Environment Distributed GA (EDGA) is introduced. In EDGA, each island has the different environment and treats the different constraint in this paper. This setting keeps the diversity of the solutions and leads the good results. The proposed method is simulated for designing truss structure that is consisted of 640 design variables. Through the simulations, the followings were made clarified. The conventional handling method of the constraints of GAs cannot derive the solutions in GAs, since the feasible domain is very narrow. The proposed method operates effectively for designing truss structure. This tendency is applicable to also other problems.

1 Introduction

These days, computers have made rapid progress in hardware and software. Especially, in the area of HPC, computational ability increased explosively. Most of super computers are massive parallel computers. Therefore, if the algorithm of application is not suitable to parallel computers, the computer cannot show efficiency even when computers have high calculation ability. As a result, researches of applications are very important for parallel computers.

Problems to find design variables that make a value of objective function maximum or minimum are called optimization problems. To solve optimization problems automatically, iterations between optimizer that decides the next searching points and analyzer that determines the value of object function should be performed. These iterations may cause the high calculation cost. Therefore, optimization problem is one of the important applications in HPC. There are many algorithms of optimization tools for optimization problems. However, most of these algorithms are not parallel but sequential algorithms. These features lead the fact that conventional optimization tools cannot gain the high parallel efficiency. Efficiency of the super computer cannot be utilized sufficiently. Therefore, new optimization algorithms that have high parallelism should be needed.

Genetic Algorithm (GA) that simulates creatures' evolution and heredity is one of powerful optimization tools. Since GA can find solutions without sensitivity information, GA can apply not only to continuos problems but also to discrete problems. Many researchers applied GAs to solve real world problems and derived good results. For example, Li et al.

optimized the shape and location of piezoelectric materials for topology optimization of flextensional actuators using SLP and GA[1]. K.Deb and S.Gulati solved the truss structure problems by GA[2]. The optimum layout of truss structure was also derived by using GAs[3]. Like this way, the GAs are very powerful tools especially for finding optimum solutions in global.

Because GA is one of multi point searching algorithms, GA needs a lot of iterations of calculation. Usually in real world problems, it takes some times to derive the value of objective function. These problems prevent many users from using GAs for real world problems, so far. However, since GA has high parallelisms, GA is one of strong candidacies for the future optimization tools on parallel computers. There are several types of GAs in parallel. The models of GAs are roughly classified into three categories; those are a master-slave model [4], an island model (Distributed Genetic algorithm: DGA)[5] and a cellular GA model [6]. In this paper, we focused on DGAs.

One of the biggest problems for GAs is handling constraints of optimization problems. GA basically is an algorithm for problems without constraints. Therefore, for the problems with constraints, some mechanisms of handling constraints are needed in DGAs. There are several studies of handling constraints in GAs. The simple method of handling constraints is a penalty term is added to the objective function for the degree of violation of constraints [7,8]. This method is also classified into two categories; the constraints are static and dynamic. The death penalty method is also simple. When a searching point violates the constraints, this point is eliminated and new searching point is derived until the constraints are satisfied. Michalewicz and Janikow presented the GENOCOP which uses projection operators that map feasible points back to feasible boundaries [9]. This method only applies to linear constraints problems. In the method of Hajela and Yoo, the generated solutions by GA operators that have guarantee of existence in feasible region [10]. Recently, there are many methods of GA with constraints using multi objective techniques. Surry, et al. presented the COMOGA where all members of the population are ranked on the basis of the constraint violations [11]. There are also several new methods of GAs that can handle constraints. Wright et al. developed the method where the searching points that violated constraints can survive in some probability [12]. Koziel and Michalewicz also developed the homomorphous mapping approach [13]. Tahk and Sun used the Lagrangian methods into GA to handle constraints [14].

These methods are powerful for optimization problems with constraints. However, these methods cannot solve some problems. Especially, when there are many design variables, most of these methods cannot solve the problems. One of the reasons not to work well for the problems with the huge number of design variables is that the feasible domain of the problems is very narrow compared to the design field. Therefore, most of the solution of the GAs does not satisfy the solutions. In this case, when there are no searching points in the initial state, GAs should have the mechanism to find the searching points that satisfy constraints.

In this paper, we propose two novel ideas for solving constraint problems using GA. At first, we introduce a new penalty function that can deal with the problems where all of the initial individuals of GA are in infeasible region. In the proposed method, the violated constraints conditions are tried to be modified to satisfy the constraints first. Then, after all the constraints are satisfied, the value of objective function is reduced. Secondly, we apply Environment Distributed Genetic Algorithms (EDGAs) into optimization problems with constraints. EDGAs is one of distributed genetic algorithms and has high parallel efficiency. At the same time, each islands of EDGA treats the different constraint and this mechanism leads to high searching ability.

These methods are applied to designing a truss structure. Truss structures are often seen in mechanical, civil and building structures. The designing truss structures are not difficult problems and can be solved by other methods effectively. However, this designing problem can easily measure the searching ability of GA. When the designing truss structures are solved by proposed method, the proposed method may apply to other types of designing structures. In the simulations of this paepr, there are 640 design variables. Through the simulations, the effectiveness and the characteristics of the proposed method are discussed.

2 Genetic Algorithms and Distributed Genetic Algorithms

2.1 Genetic Algorithms

Genetic algorithms are stochastic search algorithms based on principles of natural selection and recombination [4]. In a simple GA (SGA), the design variables are converted to the bit strings. The searching is performed in the string space. The operation of convert from real space to bit string space is called a coding. The reverse operation is called a decoding. This coding and decoding operation lead the easiness to treat discrete problems. GA is one of multi point searching methods. A set of these searching points is called a population and each searching point is called an individual. In GA, there are several operations. The population is evaluated and the best solutions are selected. These operations are called evaluation and selection respectively. After the selection, two individuals are chosen randomly and these individuals are reproduced into new individuals. Old individuals are called parents and new individuals are called children. In SGA, parents are replaced with children. This reproduce operation is called crossover. In the cross over, children inherit some bits from the parents. After the crossover, in some probabilities, the arbitrary bit is flapped. This operation is called mutation. These operations, evaluation, selection, crossover and mutation, are continued to find an optimum. One routine of these operation is called a generation. Over a number of generations, good traits dominate the population, resulting in an increase in the quality of the solutions.

2.2 Distributed Genetic Algorithms

In Distributed Genetic Algorithm (DGA), a large population is divided into smaller subpopulations, and a traditional GA is executed on each subpopulation separately. This subpopulation is called an island. Some individuals are selected from each subpopulation and migrated to different subpopulations periodically, as shown in Figure 1.

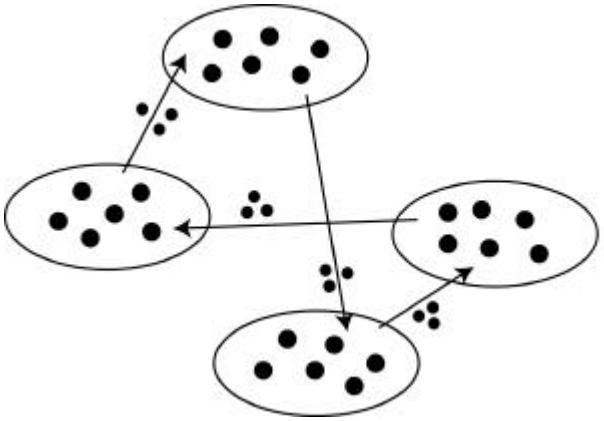

Figure 1:Distributed Genetic Algorithms

Two parameters are introduced in the migration process; migration interval which is the number of generations between each migration, and migration rate which is the percentage of individuals selected for migration from each subpopulation at the time of migration. Each subpopulation can be assigned to each processor of a parallel computer, and inter-processor communication occurs only at the migration. The migration topology adopted here is a ring with random destinations where each subpopulation has one destination and the destinations are determined randomly at every migration period as shown in Figure 1. The emigrants are selected randomly in their subpopulation. Migration selects individuals from one subpopulation and sends them to another subpopulation, which may contain individuals with very different kinds of building blocks. After migration, the genetic algorithm will mix the immigrants with the rest of the individuals in the destination subpopulation. Therefore, in the DGAs, since one population is divided into subpopulations, the number of each population is small. This leads to the quick convergence. However, DGAs have high diversity of solutions and can search in the global area. As a result, DGAs have high searching ability compared to the conventional GAs.

2.3 Genetic Algorithms with Constraints

2.3.1 Modification of Fitness Function for Problems with Constraints

As it is mentioned, Genetic Algorithm (GA) is an algorithm for optimization problems without constraints. Therefore, for solving optimization problems with constraints, fitness functions or design variables should be modified.

There are several methods dealing with constraints in GAs [5]. Most typical method is shown in the following equation.

$$\text{Equation (1): } eval(x) \;?\; \begin{cases} f(x), & \text{if } x \text{ is in feasible domain} \\ f(x) + penalty(x), & \text{otherwise} \end{cases}$$

The fitness function consists of value of objective function and penalty. When a solution does not satisfy the constraints, the value of fitness function becomes worse and the solution is eliminated. However, when the feasible domain is very narrow compared to the design field, most of the solutions that are generated in GA routines are out of feasible domain. Therefore, the effective searching cannot perform.

We propose a following fitness function,

$$\text{Equation (2): } eval(x) \;?\; \begin{cases} f(x), & \text{if } x \text{ is in feasible domain} \\ constant + Max(constraint(x)_i), & \text{otherwise} \end{cases}$$

In this fitness function, when there is no individuals that satisfy constraints in a population, the value of violated constraints are reduced at first. After some generations, the solutions that satisfy the constraints are found. After founding the solutions that satisfy the constraints, the value of the objective function becomes minimized.

2.3.2 Environment Distributed Genetic Algorithms

Usually in real world problems, there are several kinds of constraints. In this case, the Environment Distributed Genetic Algorithms (EDGA) [6] can be used. In the convention

DGAs, the environment of each island such as parameters, population size and so on is the same. On the other hand, the environment of each island in EDGA is different. In this paper, the EDGA is used for the problems with several types of constraints. In this case, the fitness function of each island is different. At first, the islands are separated into sub groups. The number of the groups is the same as the number of constraints.
In each sub group, the following fitness function is set,

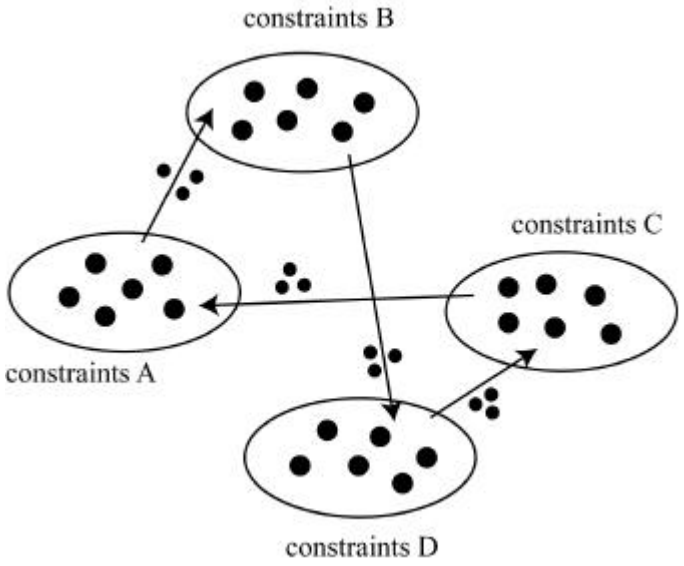

Figure 2: Environment Distributed Genetic Algorithms

$$\text{In i th island's } eval(x) \text{ ?} \quad \begin{cases} f(x), & \text{if } x \text{ is in feasible domain} \\ constant + constraint(x)_i, & \text{otherwise} \end{cases}$$

In the search of GA, the diversity of the solutions is very important. Since each island is in charge of each constraint, the diversity of the solutions is maintained. Therefore, by applying EDGA to the problems with constraints, the quick convergence and the good solution are expected.

3 Numerical Examples

3.1 Designing a Truss Structure

In this section, the proposed method is discussed through the numerical example. Designing a truss structure is chosen as a numerical example in this paper. A model of truss structure is shown in Figure 3 and the parameters of the truss structure are shown in Table 1. This is 128 stage truss and a cross section is circle. There is a force at the top of the truss.
Objective function is the total volume of this structure. DGA makes the value of objective function minimize. Design variables are areas of each component of truss structure. Design variables are not continuos but 8 types of discrete values. Therefore, this truss cannot be designed by optimization methods using sensitivity information. The displacement of node where the force is applied and stress of each member should be within in the certain values

and these are the constraints. This problem is a simple design problem. However, it can be said that the tendency of the result is the same as that of the other designing structures.

The simulation is performed on the PC cluster. The spec of PC cluster is summarized in Table 2.

Minimize total volume (X) $X \in \{1,2,\ldots,640\}$

subject to displacement at point $A(X) \leq 0.003$ [m]

stress of each element $(X) \leq 4000$ [Pa]

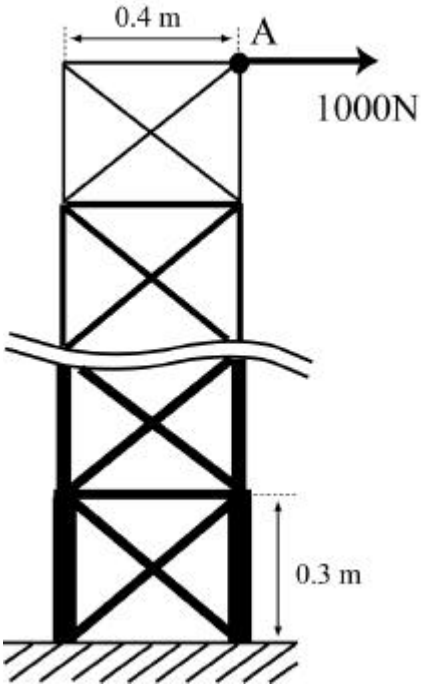

Figure 3: A Model of truss structure

Table 1: Parameters of truss structure

Young's modulus	1.0×10^9 [Pa]
Cross section	Circle
Diameter of cross section	1, 5, 10, 20, 40, 60, 80, 100 [mm]
Number of node	258
Number of elements	640

Table 2: Computing environments

Number of processors	16
Processor type	Pentium II 400 MHz (Deschutes)
Memory per processor	128 MB
OS	Debian GNU/Linux 2.2 (kernel 2.2.17)
C compiler	gcc version 2.95.2 20000220 (Debian GNU/Linux)
Basic Communication Layer	100 Mbps Ethernet
Message Passing Library	MPICH 1.2.1

Table 3: Parameters of GAs

Population size	256
Crossover method	One point crossover
Crossover rate	0.8
Mutation rate	0.001
Max generation	500
Migration topology	Ring
Migration rate	0.2
Migration interval	5 generation

3.2 Why GAs are not Good at Finding the Solutions in the Problems with Huge Number of Design Variables?

It is said that GAs are good at finding solutions in global area. Actually, GA can find good solutions in test functions. However, very few users use GAs for problems that have huge number of design variables. Why GAs are not good at finding the solutions in the problems with huge number of design variables? One of the reasons is that the feasible domain is very narrow. Constraints make the feasible domain very narrow.

As it is described before, the equation (1) is a typical fitness function to dealing constraints. In Figure 4, the transition of the solutions when the equation (1) is used is shown. In this case, only the stress is considered as a constraint and the solution is derived by SGA.

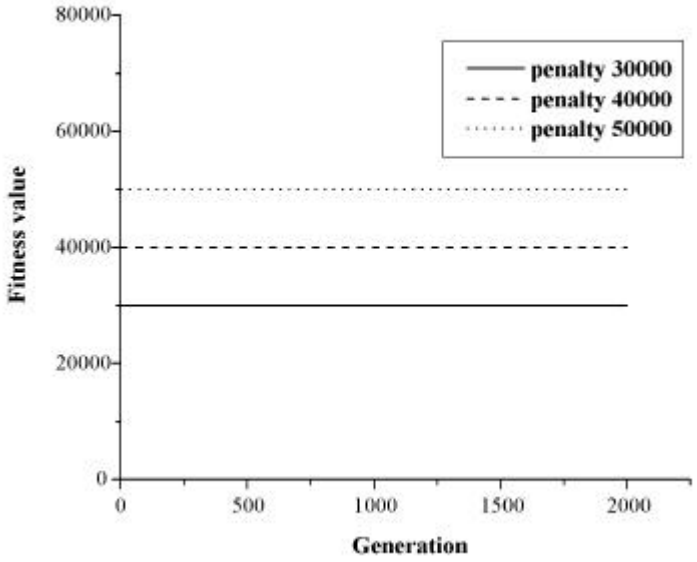

Figure 4: Fitness transition (Equation 1)

The values of penalty are much bigger than the value of objective function. From this figure, the value of fitness is not changed with along to the generations. This result means that any individual of GA cannot satisfy the constraint. Therefore, the solution cannot be derived with any parameters of penalty when equation (1) is used. These results are the almost same as the results by DGA.

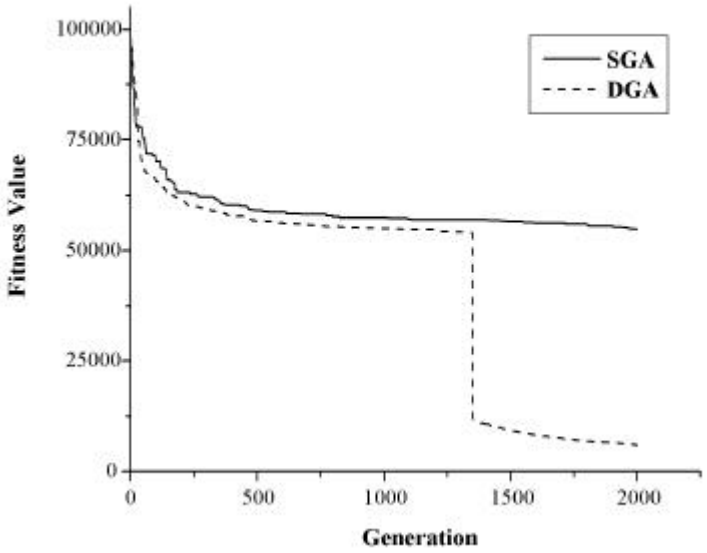

Figure 5: Fitness transition (Equation 2)

In Figure 5, the transition of the solution of SGA and DGA when the proposed method (equation 2) is used. The penalty 50000 is used in this case. This value is much bigger than that of objective function and constraints. DGA reduced fitness largely at around 1300 generation. At this point, DGA found the point that satisfies the constraints and fitness function was shifted. SGA also reduced the fitness. That means SGA was trying to find the

point that satisfies the constraint but SGA could not. From these results, it is concluded that the proposed method is very effective for designing truss structure in DGA.

3.3 Searching Ability of Distributed Genetic Algorithms

Since the population is divided into sub populations in DGAs, the number of each sub population is small. This feature leads to the quick convergence. At the same time, because of the migration mechanism, the diversity of the solutions is maintained during the searching. Because of these characteristics of DGAs, DGA has a high searching ability compared to the conventional GA.

In Figure 6, the fitness transition of Simple GAs (SGAs) and Distributed GAs(DGAs) is shown. In this case, both the displacement and the stress are constraints. Therefore, the problem becomes more complex compared to the problem where only the stress is constraint. In this simulation, the penalty is also 50000.

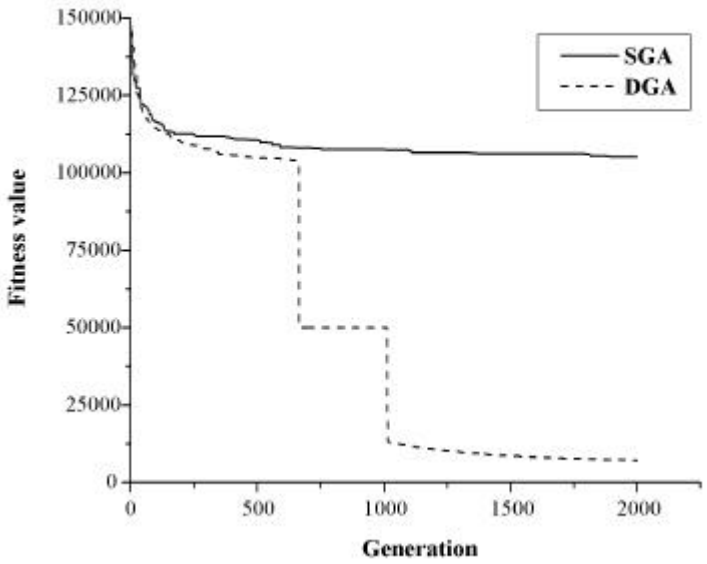

Figure 6: Comparison SGAs and DGAs

SGAs cannot find the point that satisfies the constraints. On the other hand, DGAs can reduce the fitness. DGA reduced the fitness largely at around 700 and 1000 generations. These are the points that DGA found the points that satisfy the two types of constraints respectively. From this result, it can be concluded that DGAs has higher searching ability compared to SGAs. This result can be derived not only in designing truss structure but also in numerical test functions.

3.4 Effectiveness of Environment Distributed Genetic Algorithms

In 2.3.2, Environment Distributed Genetic Algorithm (EDGA) is introduced. In this section, the effectiveness of EDGA is discussed.

In Figure 7, the fitness transition of DGA and EDGA is shown. In this case, both the displacement and the stress are constraints. The penalty is also 50000. 8 processors (8 islands) are used in this simulation. The followings are the fitness function of each island.

Island 0 – 3

$$eval(x) \leftarrow \begin{cases} f(x), & \text{if } x \text{ is in feasible domain} \\ constant + Max(constraint_1(x)_i) + Max(constraint_2(x)_i), & \text{otherwise} \end{cases}$$

Island 4,5

$$eval(x) \leftarrow \begin{cases} constant + Max(constraint_2(x)_i), & \text{if } x \text{ satisfy } constraint_1 \\ constant \leftarrow 2 + Max(constraint_1(x)_i), & \text{otherwise} \end{cases}$$

Island 6,7

$$eval(x) \leftarrow \begin{cases} constant + Max(constraint_1(x)_i), & \text{if } x \text{ satisfy } constraint_2 \\ constant \leftarrow 2 + Max(constraint_2(x)_i), & \text{otherwise} \end{cases}$$

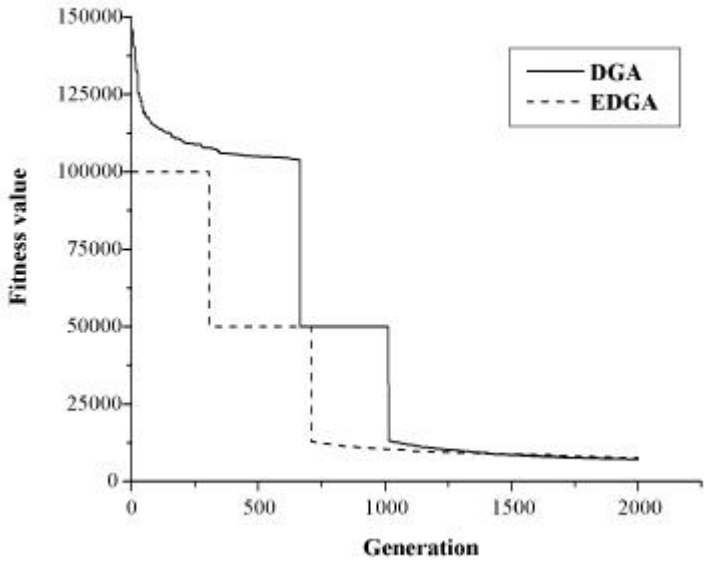

Figure 7: Comparison DGAs and EDGAs

From this figure, it is obvious that EDGA can find solution quickly compared to the DGA. EDGA has the mechanism to keep the high diversity. This mechanism derived the good result.

3.5 Speed Up

In DGAs, each sub population is allotted to one processor. When the number of processors is different, the model of DGA is different. Since network communication has just occurred at the time of the migration, network cost is very small in DGAs. Therefore, DGA is a very good parallel model. In Figure 8, the speed up is shown with respect to the number of processors. The derived speed up is almost linear or more than that. There are two reasons for the super linear. One of the reasons is that the necessary calculation cost is reduced, since model is changed. In GAs, there are several operations where individuals are sorted. Therefore the smaller population size leads to the smaller calculation cost. The other reason is that this model needs not to communicate frequently. Therefore, the increase of parallel efficiency of DGAs have parallel factor and distributed factor.

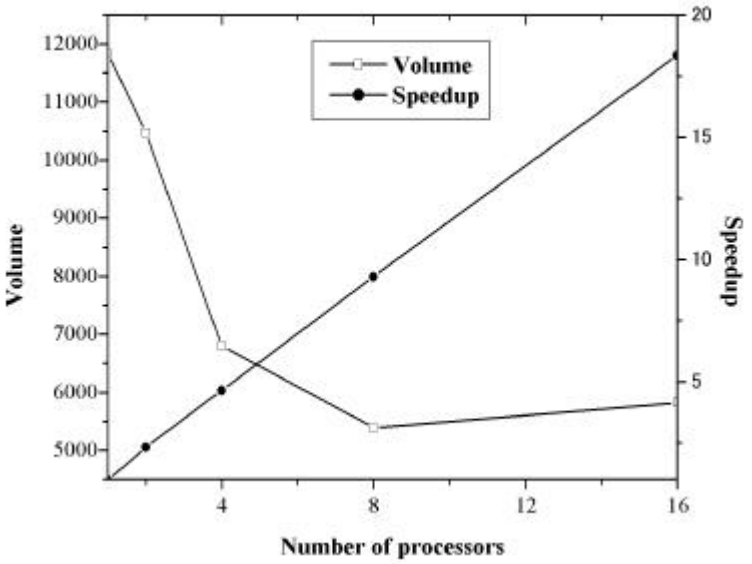

Figure 8: Speed Up

4 Conclusions

In this paper, we discussed Distributed Genetic Algorithms (DGAs) for designing truss structure. The designing truss structure is chosen as one example of a real world problem.

Especially, we discuss handling constraints. Usually, a feasible domain of optimization problems that have the huge number of design variables and constraints is very narrow. Therefore, the most genes of DGAs become dead genes that do not satisfy the constraints. In this case, DGAs cannot search solutions effectively. The mechanism not to generate dead genes should be important. In this paper, we proposed a new setting of a fitness function of DGAs. In the proposed setting, DGA reduces the value of the violated constraints at first and find the solutions that satisfy the constraints. After finding the solutions that satisfy the constraints, the value of the objective function is reduced. By this setting, DGA can find the optimum solution. At the same time, Environment Distributed Genetic Algorithm (EDGA) is also introduced to constraint problems. In EDGA, the different constraint is treated in each island and this mechanism leads to high searching ability. Through the designing truss structures on PC cluster systems, the proposed setting and EDGA are discussed. From the results of the simulations, it is clarified that the proposed method leads the good solution. The simulation was performed on the PC cluster that is consisted of 16 nodes and derives the good speed up. Therefore, we can conclude that EDGA with proposed handling method of constraints is a good method for designing huge structures.

References

[1] Ying Li, Xuemei Xin, Noboru Kikuchi and Kazuhiro Saitou, Optimal shape and location of piezoelectric materials for topology optimization of flextensional actuators, Proc. of the genetic and evolutionary computation conference, 2001, pp.1085-1090

[2] K.Deb, S.Gulati, Design of truss-structures for minimum weight using genetic algorithms, Finite Elements in Analysis and Design, 2001, Vol 37, Iss 5, pp.447-465

[3] O Hasancebi, F Erbatur, Layout optimization of trusses using improved GA methodologies, Acta Mechanica, 2001, Vol 146, Iss 1-2, pp.87-107

[4] D.Levine, "User's guide to the PGAPack parallel genetic algorithm library", T.R.ANL-95/18, Algonne National Laboratory, 1996

[5] Reiko Tanese, "Parallel Genetic Algorithms for A Hypercube", Proceedings. of 2nd ICGA, 1987

[6] Martina Gorges-Schleuter, "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy", Proceedings of 3rd ICGA, 1989

[7] A.Homaifar, C.X.Qi and S.H.Lai, "Constrained Optimization Via Genetic Algorithms", Simulation 62(4), 1994, pp.252-254

[8] J.A.Joines, C.R.Hourck, "On the Use of Non-Stationary Penalty Functions to Solve Non-linear Constrained Optimization Problems with GA's", IEEE Conference on Evolutionary Computation 2, 1994, pp.579-584

[9] Z.Michalewicz, C.Z.Janikow, "Handling Constraints in Genetic Algorithms", Proceedings of the International conference on Genetic Algorithms 4, pp.151-157

[10] P.Hajela, J.Yoom, "Constraint Handling in Genetic Search – A Comparative Study", AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference – Collection of Technical Papers 4, 1995, pp.2176-2186

[11] P.D.Surry, N.J.Radcliffe, "The COMOGA Method: Constrained Optimization by Multi-Objective Genetic Algorithms", Control and Cybernetics 26(3), 1997, pp391-412

[12] J.A.Wright, R.Farmani, "Genetic Algorithms: A fitness formulation for constrained minimization", Proceedings of the Genetic and Evolutionary Computation Conference, 2001, pp.725-732

[13] S.Koziel, Z.Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization", Evolutionary computation 7(1), 1999, pp. 19-44

[14] M.-J,Tahk, B.-C.Sun, "Coevolutionary Augmented Lagragian Methods for Constrained Optimization", Evolutionary computation 4(2), 2000

[15] D.E. Goldberg, "Genetic algorithms in search", optimization and machine learning, Addison-Wesley, 1989

[16] Zbigniew Michalewicz, "Genetic Algorithms + Data Structure = Evolution Programs", Springer-Verlag, 1992

[17] Mika Kaneko, Mitsunori Miki, Tomoyuki Hiroyasu, "A Parallel Genetic Algorithm with Distributed Environment Scheme", The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, 2000

[18] Enrique Alba, Jose M.Troya, "A Surbey of Parallel Distributed Genetic Algorithms", Complexity Vol.4 No.4, John Wiley & Sons, Inc., 1999

[19] Macro Dorigo, Vittorio Maniezzo, "Parallel genetic algorithms: Introduction and overview of current research", Parallel Genetic Algorithms: Theory and Applications, IOS Press, 1993

[Conference] HPC Asia 2001 is the 5th International Conference and Exhibition on High Performance Computing in the Asia-Pacific Region

[Place] Royal Pines Resort, Gold Coast, Queensland Australia

[Date] September 24-28, 2001

[Section] Algorithms