

A PARALLEL GENETIC ALGORITHM FOR GENERATION EXPANSION PLANNING

Yoshikazu Fukuyama, member
Fuji Electric Corporate R & D, Ltd.
No. 1, Fuji-machi, Hino-city, Tokyo, 191 Japan.

Hsaio-Dong Chiang, senior member
School of Electrical Engineering
Cornell University, Ithaca, NY 14853 USA

Abstract: This paper presents an application of parallel genetic algorithm to optimal long-range generation expansion planning. The problem is formulated as a combinatorial optimization problem that determines the number of newly introduced generation units of each technology during different time intervals. A new string representation method for the problem is presented. Binary and decimal coding for the string representation method are compared. The method is implemented on transputers, one of the practical multi-processors. The effectiveness of the proposed method is demonstrated on a typical generation expansion problem with four technologies, five intervals, and a various number of generation units. It is compared favorably with dynamic programming and conventional genetic algorithm. The results reveal the speed and effectiveness of the proposed method for solving this problem.

Key words: Generation Expansion Planning, Combinatorial Optimization, Parallel Genetic Algorithm, Parallel Computation, Multi-processors

INTRODUCTION

Generation expansion planning is an important planning activity for utility companies. Its basic objective is to determine the schedule for the construction of generation plants, the number and time of introduction for each new generation unit into production, and interconnecting links so that a reliable and economic supply for predicted load demands is ensured. The economic issue can be addressed by minimizing the expected sum of the investment and operation costs associated with each new generation unit under uncertain conditions. The reliability requirement is to ensure an adequate energy supply for predicted load demands under uncertain conditions. Sources of this uncertainty may come from future operating conditions (e.g. load variation and growth, equipment availability), social activities (e.g. environmental constraints, construction time) and economic activities (e.g. economic growth, fuel costs, interest rates and financial constraints).

There are basically two approaches for generation expansion planning; stochastic [1,2,4,5,9,10,11] and deterministic [3,6,7,8]. The stochastic approach takes uncertain factors into account in an explicit manner. The deterministic approach evaluates system

performance under different scenarios. The need to develop a fast optimization method for the deterministic approach is obvious because it involves a great number of simulations under many different scenarios. Levin et. al. [3] developed a deterministic screening method based on the Kuhn Tucker condition. This method is efficient but a unit capacity is treated as a continuous variable, instead of discrete unit sizes. Yang et. al. [6] and David et. al. [7][8] developed a deterministic multi-objective decision procedure based on dynamic programming (DP). They can treat discrete unit sizes. However, execution time may increase drastically as the dimensions of the problem, such as the number of technologies and generation units, increases. Therefore, a fast deterministic method has not yet been developed.

This paper develops a parallel genetic algorithm (PGA) to solve the problem of determining the optimal number of newly introduced generation units at each time interval under different scenarios. genetic algorithm (GA) is employed because of its ability and efficiency in finding global optimal solutions, and its high suitability for parallel computation [12]. There are two classes of PGA: coarse-grain [13-15] and fine-grain [16,17]. The former utilizes fewer processors with less communication needed than the latter. The PGA developed in this paper belongs to the class of coarse-grain PGA, achieving a trade-off between computational speed and hardware cost.

The contribution of this paper can be summarized as follows: We propose a parallel genetic-based method for optimal long-range generation expansion planning. Since the efficiency of a GA-based solution algorithm depends greatly on the coding scheme and the selection method used, the proposed PGA uses an effective coding scheme and selection method tailored to the problem. It can deal with discrete unit sizes of generation units and the execution time is almost proportional to the number of newly introduced generation units. Thus, the proposed PGA is effective for high-dimension generation expansion problems. This method can be a fast deterministic method under a certain scenario. It can be used as a basic tool for studying the generation expansion problem independently. Moreover, further integrated methods considering multi-objective decision procedure [6][7][8] and uncertainty [10] can be developed based on the proposed method. A transputer, which is one of the efficient multi-processors, is utilized to implement the proposed method. The processor can be implemented on the engineering workstations (EWS), which are widely used. Therefore, our results indicate the possibility of using multi-processors on EWS independently to solve the optimal long-range generation expansion planning problem in a practical field. The proposed PGA is applied to a typical test system consisting of 4 technologies and 5 time intervals. These simulation results compare favorably with those obtained by DP and conventional GA.

PROBLEM FORMULATION

Cost function

Cost function can be considered as linear combination of fix and variable costs through all time intervals as shown below.

$$\min f_c(x) = \sum_{i=1}^N \left[\sum_{j=1}^M \{a_j (X_{(i-1)j} + x_{ij}) + b_j w_{ij} (\phi_{i(j-1)}, \phi_{i(j-1)} + X_{ij})\} \right] \quad (1)$$

$$X_{ij} = \sum_{k=1}^j x_{kj} + X_{0j} \quad (i=1,2,\dots,N, j=1,2,\dots,M) \quad (2)$$

$$\phi_{ij} = \sum_{k=1}^j X_{ik} \quad (i=1,2,\dots,N, j=1,2,\dots,M) \quad (3)$$

$$w_{ij} (\phi_{i(j-1)}, \phi_{i(j-1)} + X_{ij}) = \int_{\phi_{i(j-1)}}^{\phi_{i(j-1)} + X_{ij}} L_i(u) du \quad (i=1,2,\dots,N, j=1,2,\dots,M) \quad (4)$$

where,

- N : the total number of time intervals,
- M : the total number of technologies,
- a_j : fix cost coefficient of generation technology j,
- b_j : variable cost coefficient of generation technology j,
- x_{ij} : introduced amount (MW) of generation technology j at interval i,
- X_{ij} : total introduced amount of generation technology j till interval i,
- w_{ij} : energy output (MWh) of generation technology j at interval i,
- ϕ_{ij} : the loading point of generation technology j at interval i,
- L_i : load duration curve at interval i.

Constraints

(1) maximum and minimum capacity of introduced unit

$$x_{j,\min} \leq x_{ij} \leq x_{j,\max} \quad (i=1,2,\dots,N, j=1,2,\dots,M) \quad (5)$$

where,

- $x_{j,\min}$: minimum capacity of technology j,
- $x_{j,\max}$: maximum capacity of technology j.

(2) supply and demand balance at each interval

$$\sum_{j=1}^M X_{ij} \geq P_{Di} + P_{Ri} \quad (i=1,2,\dots,N) \quad (6)$$

where,

- P_{Di} : peak load at interval i,
- P_{Ri} : reserve at interval i.

(3) generation mix at the current and final interval

$$\begin{aligned} X_{0j} &= X_{Cj} \quad (j=1,2,\dots,M) \\ X_{Nj} &= X_{Fj} \quad (j=1,2,\dots,M) \end{aligned} \quad (7)$$

where,

- X_{Cj} : total generation amount of generation technology j at the current interval,
- X_{Fj} : total generation amount of generation technology j at the final interval.

(4) cost coefficient constraints

$$b_j \leq b_{j+1} \quad (j=1,2,\dots,M) \quad (9)$$

OVERVIEW OF PARALLEL GENETIC ALGORITHM (PGA)

Conventional genetic algorithm (GA)

GA is one of the stochastic search algorithms based on the mechanics of natural genetics. A solution variable for the

problem is first represented using artificial chromosomes (strings). In other words, the problem is encoded to strings that GA can handle. A string represents one search point in the solution space. GA is a parallel search method because it uses a set (population) of strings (i.e. multiple search points). It modifies strings (searching points) using natural selection and genetic operators such as cross-over and mutation. After convergence, strings are decoded to the original solution variables and the final solutions are obtained. The procedure of GA can be formulated as follows. The details can be found in [12].

(1) Representation of the problem using strings,

The parameters of the original problem are represented using series of binary, decimal, or floating point number, so-called a string. Each string represents a single search point in the solution space.

(2) Generation of initial string population

Strings are generated randomly. However, it is sometimes effective to produce initial strings using the problem-dependent methods [12]. These methods produce sub-optimal initial searching points in the solution space. Therefore, fast convergence to the optimal solution can be expected.

(3) Evaluation and selection of each string

Strings are evaluated using the fitness function, which represents the tendency of fitness of strings to the problem. A good candidate for the fitness function is the objective function. Basically, the strings that have higher fitness values are selected with high probability. In other words, better searching points are selected according to their objective function values. GA uses roulette wheel selection [12]. However, there are improved methods such as stochastic sampling with or without replacement [12].

(4) String operation

GA performs cross-over and mutation. These operations produce new searching points from the current searching points. Natural selection and string operations are repeated until the strings are converged to the optimal or sub-optimal solution.

Parallel genetic algorithm(PGA)

PGA performs GA in parallel and it has two different versions: coarse-grain and fine-grain. In coarse-grain PGA, distributed sub-populations with some strings are mapped into several processes and strings are sometimes exchanged between sub-populations during optimization. On the other hand, in fine-grain PGA, each string is mapped into each process and string information is exchanged between each process. In the former case, a parallel program consists of a few compute-intensive processes with little communication demands. The latter case is characterized by the large number of processes with low computational requirements but high demands on communication in order to coordinate all processes. It is obvious that coarse-grain PGA is appropriate for practical applications considering the trade-off between computational speed and hardware cost.

In [13], a formulation of coarse-grain PGA was proposed. Several sub-populations were used for optimization based on the analogy of allopatric speciation instead of one large population. In [14], a comparison between the conventional GA and the coarse-grain PGA was investigated and it showed the robustness and fastness of the PGA algorithm as compared with the conventional GA. In the coarse-grain PGA, Every k generations (k is called epoch), some strings are migrated from the neighboring sub-population. Since communication is restricted among neighboring sub-population, independence of each sub-population with the small number of strings allows us to search in different solution spaces of the problem and provides a nearly linear speed-up [14][15]. GA itself is a parallel search method. In

addition, the coarse-grain PGA performs several GA procedures in parallel. Therefore, it can search various solution spaces of the problem efficiently. In [15], various coarse-grain and fine-grain PGAs were summarized and classified.

PROBLEM FORMULATION USING GA

String representation

String representation is an important factor for solving the problem using GA. The following string representation method is used in this paper.

- o The length of a string equals to the number of total newly introduced generation units.
- o Each string position represents the introduced time interval of each unit.

Fig. 1 is an example of strings that includes 2 nuclear, 5 coal, 5 LNG (liquefied natural gas), and 14 thermal units. It means that two of nuclear units should be introduced at interval 1 and 3. Decimal coding is used in the figure. When using binary coding, the number of digits for representing the final interval is necessary for each unit.

The length of each string corresponds to the total number of newly introduced units in the representation method. Therefore, even if the string operation of cross-over and mutation is performed, the boundary conditions of the problem shown in (7) and (8) are satisfied. Moreover, even if the time interval for introducing each unit is different, strings with the same introduced intervals of each technology can be produced. For example, in Fig. 1, 13* and 31* mean the same introducing procedure for the nuclear generation units. Here, * represents the rest of the strings except for the nuclear generation units. In other words, using the above representation method, the optimal string is not unique and several strings can represent the same optimal introduced number of generation units. Therefore, the probability of producing the optimal string is high.

Generation of initial string population

Initial strings in the population are generated randomly. When a string is generated, a string is accepted if it satisfies constraints (5) and (6). Otherwise, the string is discarded and a new string is generated again. According to our experiences, introduction of new generation unit has the tendency of spreading out over interval in the optimal solution. Therefore, random generation of initial string must be appropriate for the particular string representation.

Evaluation and selection of each string

(1) Evaluation of strings

The fitness value of each string is evaluated using the following equation which is related to (1).

$$f = \frac{1}{1 + f_c} \quad (10)$$

A certain string may be dominant in the initial stage of the optimization. In such a case, the whole population may converge to the string quickly (premature convergence). In order to avoid this problem, a linear scaling technique [12] is used in the proposed method. The scaling technique is used for maintaining appropriate conflict conditions among strings, because large variation of the strings is one of the most important characteristics in GA. Maximum fitness value can be bounded using the scaling technique for maintaining variety of strings. It modifies each fitness value using the following equation.

$$f' = \alpha f + \beta \quad (11)$$

where,

f' : modified fitness value

f : original fitness value

α : coefficient

β : coefficient

(2) Selection of each string

We briefly describe three selection methods.

i) Roulette wheel selection (RWS) [12]

This conventional method is stochastic in nature and the procedure can be expressed as follows:

- o Sum up the fitness values of each string ($\sum f_i$).

- o Reproduce each string according to the following probability.

$$pselect_i = f_i / \sum f_i \quad (12)$$

ii) Remainder stochastic sampling without replacement (RSWOR) [12]

The method is one of the improved methods of RWS and the procedure can be expressed as follows:

- o The expected number of strings (e_i) can be calculated using the following equation.

$$e_i = pselect_i \cdot n_s \quad (13)$$

where,

n_s : the number of strings

- o Each string is reproduced using the integer part of e_i .

- o Each string is reproduced more by using the fraction part of e_i as the probability for coin toss (Bernoulli trials). The procedure is repeated for each string until the number of strings reaches n_s . For example, if e_i equals 2.5, two strings are reproduced according to the integer part. The more reproduction probability for the string should be 0.5.

iii) Remainder stochastic sampling with replacement (RSWR) [12]

This method is also one of the improved methods of RWS and the procedure can be expressed as follows:

- o Calculate the expected number of strings (e_i) using (13).
- o Each string is reproduced using the integer part of e_i .
- o Each string is reproduced more by using the fraction part of e_i as the weight for RWS. Namely, the method uses RWS only for the fraction part of e_i . The procedure is repeated until the number of strings reaches n_s .

Compared with the above three methods, RWS uses only a stochastic way for selection of the strings, while other two methods use both deterministic and stochastic ways. Therefore, the latter two methods can select more precisely expected number of strings based on the fitness values of each string.

String operation

When cross-over and mutation are performed, strings that satisfy constraints (5) and (6) are generated. Namely, after string operation, if it does not satisfy the constraints, string operation is performed again.

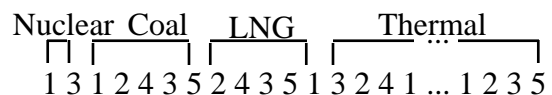


Fig. 1 An Example of strings using the representation method.

The cross-over used here is a simple one point cross-over [12]. The binary and decimal coding at the operation can be compared as follows: Figure 2 shows an example of the cross-over at the second string position using the binary coding. Here, an introduced interval is assumed to be represented with three digits. Therefore, the figure shows the introduced interval of only one generator. Numbers in parentheses in the figure show decimal number of each binary number. For example, the left-hand side two strings represent that the generation units should be introduced at intervals 5 and 2. In the example, the cross-over operation can change the introducing intervals from 5 and 2 to 6 and 1. Figure 3 shows an example of the cross-over also at the second string position using the decimal coding. It should be noted that the introduced intervals of five generation units are represented in the figure. Using the decimal coding, the operation can change only the combination of introduced intervals of total generation units and can not change the introduced intervals themselves. Therefore, the cross-over using the binary coding can obtain more various transitions for new strings than that using the decimal coding. However, more number of digits is necessary in the binary coding and it takes more execution time for optimization. Therefore, trade-off should be evaluated when a coding method is selected.

The interval number except the current interval is selected among maximum and minimum interval by the mutation using the decimal coding. Using the binary coding, one bit is changed from 1 to 0 and vice versa. The mutation both using the binary and decimal coding can change each introducing interval. It should be taken into account that the interval exceeding the maximum interval may be produced at the mutation using the binary coding according to the digit number. Namely, if the largest number which can be represented by a certain digits exceeds the maximum time interval of the problem, the operation may produce time intervals which are not allowable. On the other hand, the mutation using the decimal coding can always produce an allowable interval. This means that tendency of violation of the constraints by the string operation using the binary coding is larger than that using the decimal coding.

IMPLEMENTATION OF COARSE-GRAIN PGA

The procedure of the proposed parallel genetic algorithm for solving the generation expansion problem is summarized in Fig. 4 It consists of five procedures. Only the migration procedure is added to the conventional GA.

The proposed PGA has been implemented on transputer, that is one of the MIMD (multi instruction stream and multi data stream) processors. The concept of the coarse-grain PGA on transputer is shown in Fig. 5. In order to realize the coarse-grain PGA, the total population is distributed into several sub-populations. Each sub-population is allocated to each process and the conventional GA is performed using each sub-population on each process. Strings with highest fitness values are migrated

1 | 0 1 (5) | 1 0 (6)
 ->
 0 | 1 0 (2) | 0 1 (1)

Fig. 2 An example of the cross-over using the binary coding.

| 5 # ### | 2 *
 ->
 *** | 2 * *** | 5 #

Fig. 3 An example of the cross-over using the decimal coding.

from the neighboring processes at every epoch. The processes are installed on the actual transputer processors as the logical process. The logical processes can be installed in various ways on transputers easily. For example, four logical processes in Fig. 4 can be installed on one processor. They can be installed on 2 processors. In such a case, 2 processes of right hand side may be allocated to one processor and other 2 processes may be allocated to another processor. They can also be installed on 4 processors. In such a case, every processor has one process. If a processor has more than one process, each process shares the

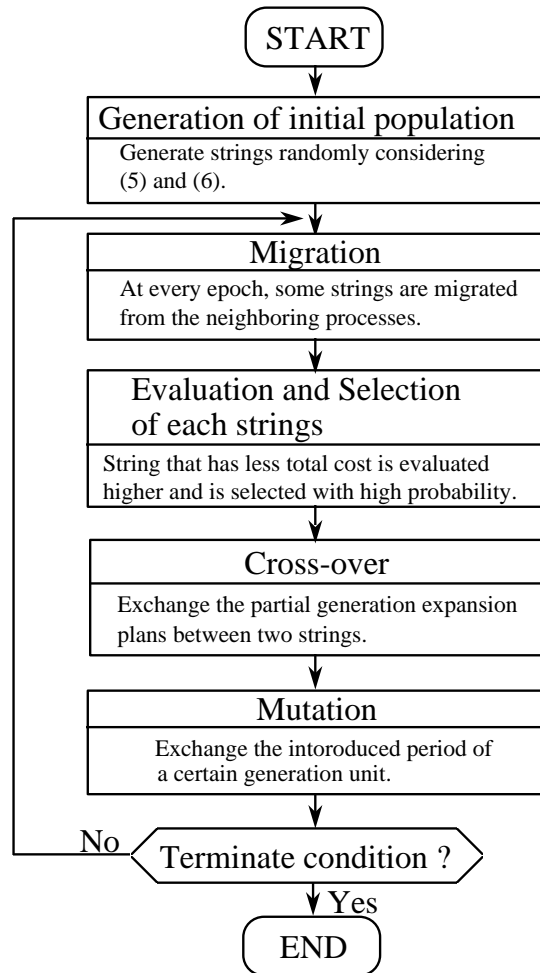


Fig. 4 The flow chart of the coarse-grain PGA for solving the generation expansion problem.

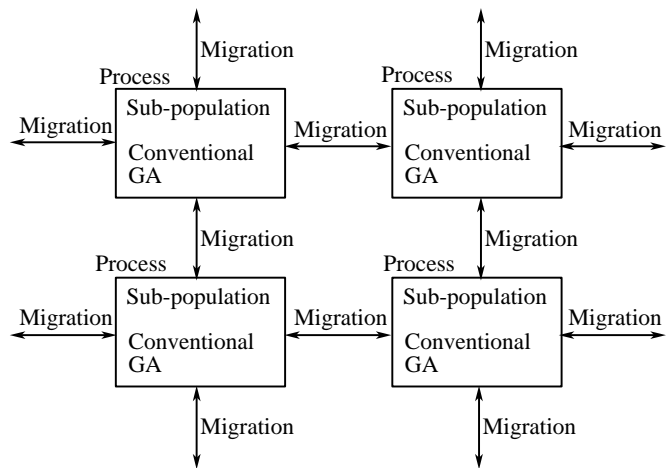


Fig. 5 Concept of coarse-grain PGA on transputer.

local memory and executes processes using the time-sharing technique. Therefore, the number of processors affects only execution time. This flexible feature is suitable for practical implementation of the proposed method as mentioned below.

NUMERICAL EXAMPLES

The proposed method has been applied to a test system consisting of 4 technologies and 15 years' intervals. It determines the number of newly introduced generation units at every three year (interval). Therefore, there are 5 intervals for optimization. The proposed method is compared with DP and the conventional GA.

Example 1

The proposed method has been applied to the generation expansion planning with 26 generation units to be introduced.

(1) Calculation parameter

Table 1 shows cost parameters and unit capacity of each technology. Table 2 shows the generation mix at the current and the final time interval (15 years after). The generation mix is given in advance. Load duration curves are shown in table 3. The curves are approximated with a second order function of loads. An Assumed scenario is the followings. Peak loads are assumed to increase 4 % per year and base loads are assumed to increase 2 % per year. The reserve is assumed to be 10 % of the peak load. Parameters for GA are as follows:

String representation: binary coding, decimal coding
 Selection method: RWS, RSWOR, RSWR
 cross-over probability: 0.5
 mutation probability: 0.01
 migration rate: 0.2
 epoch: 10 generations
 maximum generation: 100
 the number of strings in each sub-population: 4,8,16,32,64

(2) Calculation results

Table 4 shows the optimal number of newly introduced generation units at each time interval that can be obtained using DP, GA, and PGA. They give the same optimal result in this example.

Table 5 shows averages of the maximum fitness value among three selection methods using the conventional GA after 100 trials with different initial strings. The fitness value of the optimal solution shown in table 4 is assumed to be 100. The decimal coding and 64 strings for the population are used for all methods. The results show that RSWOR and RSWR are able to produce better results than the conventional RWS. However, the difference between the results of RSWOR and RSWR can not be observed for the problem. According to the results and some experiences reported in [12], we decided to use RSWR in the rest of our simulations.

Fig. 6 shows the frequency distribution of maximum fitness values after 100 trials with different initial strings when the fitness value of the optimal solution is assumed to be 100. PGA uses the decimal coding and total population size is set to be equal among all cases as follows:

GA using the binary coding: 64 strings,
 GA using the decimal coding: 64 strings,
 PGA with 2 processes: 32 strings per process,
 PGA with 4 processes: 16 strings per process,
 PGA with 8 processes: 8 strings per process,

PGA with 16 processes: 4 strings per process.

PGA shown in Fig. 4 is realized using logical processes on one to three transputers. Since logical processes are easily installed on actual transputer processors, same results can be obtained using the different number of processors. The only difference is the computation time according to the number of processors.

According to the results, the decimal coding generates better solutions than the binary coding. PGA with more processes can produce much better solutions. Moreover, using 16 processes, half

Table 1 Cost parameters and unit capacity of the test system [18].

Technology	Variable Cost [yen/kWh]	Fixed Cost [yen/kW]	Unit Capacity [MW]
Nuclear	5.0	65000.0	1000.0
Coal	8.0	45000.0	500.0
LNG	10.0	40000.0	300.0
Thermal	15.0	35000.0	200.0

Table 2 Generation mix at current and final interval. [18].

Tech.	Current interval		Final interval (15 year after)	
	Current number of units	Unit Capacity [MW]	Final number of units	Unit Capacity [MW]
Nuclear	4	4000.0	6	6000.0
Coal	6	3000.0	11	5500.0
LNG	4	1200.0	9	2700.0
Thermal	14	2800.0	28	5600.0

Table 3 Load duration curves at current and final interval [18].

Interval	Peak Load [MW]	Base Load [MW]	$L^{-1}(x) = (X-d)^2/c$	
			$c * 10^3$	$d * 10^4$
Current	10000.0	3500.0	0.4225	1.000
Final	18000.0	4710.5	1.7661	1.800

Table 4 Optimal number of newly introduced generation units at each time interval.

3 year interval	1	2	3	4	5
Nuclear	1	0	0	1	0
Coal	0	1	2	0	2
LNG	0	1	1	2	1
Thermal	2	4	2	2	4
Total generation [MW]	12400	14000	15700	17700	19800
Maximum load [MW]	12372	13916	15652	17604	19800

Table 5 Average of the maximum fitness value using various selection methods (100 trials).

Selection Method	RWS	RSWOR	RSWR
Average maximum fitness value	99.954	99.959	99.959

*) The fitness value of the optimal solution shown in table 4 is assumed to be 100.

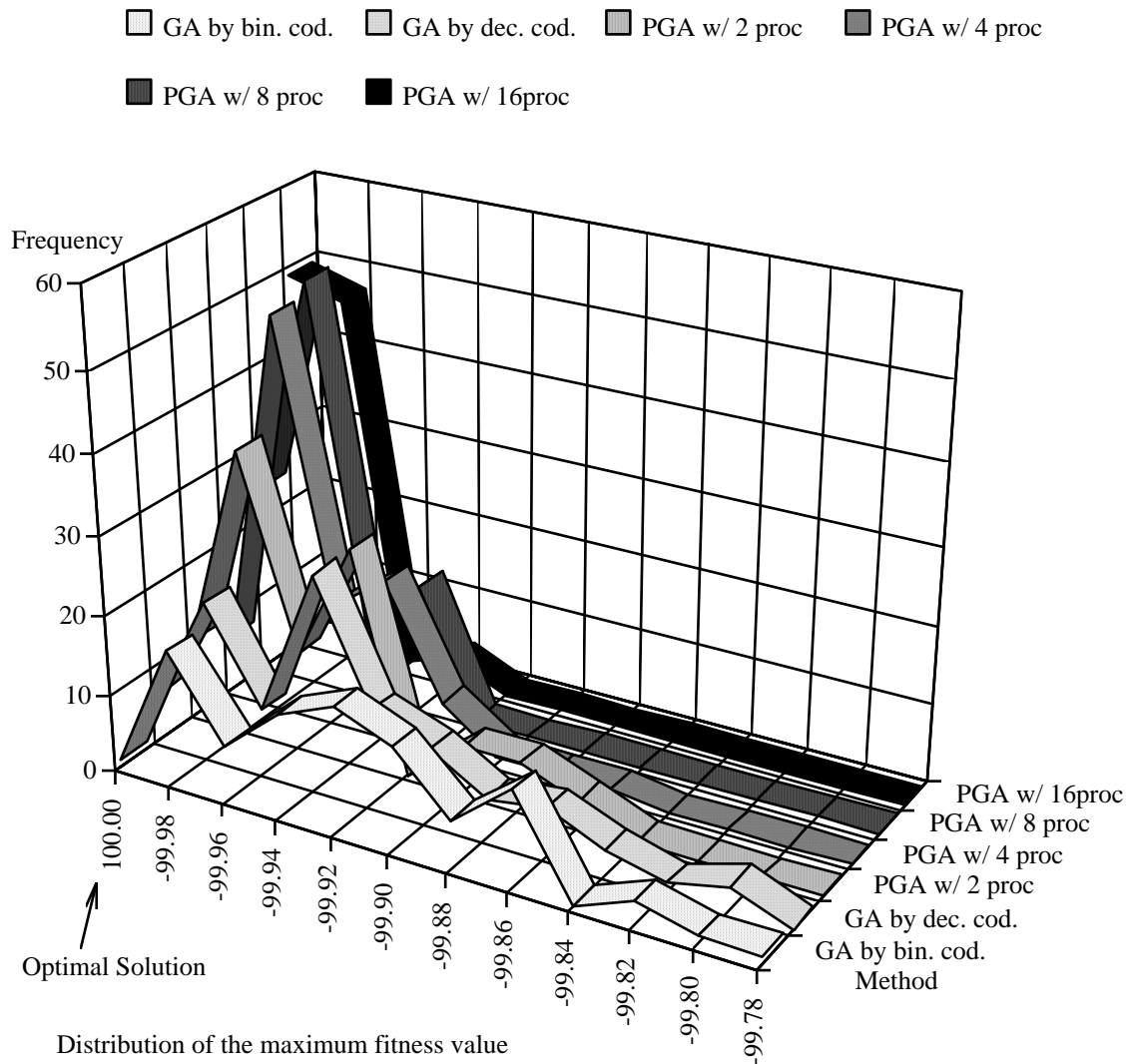


Fig. 6 Frequency distribution of maximum fitness values using various GA methods (100 trials).

Table 6 Observed and estimated average execution times.

Method	Time [sec]
Dynamic Programming	58.996
GA using the binary coding	456.733
GA using the decimal coding	46.880
PGA with 2 processors	23.098
PGA with 4 processors	14.026
PGA with 8 processors	6.635
PGA with 16 processors	3.206

of the total trials can produce the optimal solution. Table 6 shows the observed and estimated execution times using each method. Here, each process is assumed to be installed on one processor. The execution times of more than 4 processes are calculated using observed total process and communication times. The conventional DP can always produce the optimal solution but with long execution time. The GA using the decimal coding is 25 [%] faster than the DP. On the other hand, The proposed method can compute approximately 18 times faster than the conventional DP. It produces the optimal solution with high probability (about 50 %) using 16 processors. These features favor our proposed

PGA for solving the generation expansion problem under a certain scenario because it allows us to perform more number of simulations and higher probability to obtain the optimal solution under various scenarios. Moreover, flexible installation of logical processes on transputers allows us to consider a trade-off between computational speed and cost in their own environment and determine the number of processors.

Example 2

The proposed method has been applied to the generation expansion problems with a various number of newly introduced generation units. It uses the same parameters as example 1 and the difference is that the number of newly introduced generation units is multiplied by 1.5, 2.0, 2.5, 3.0, and 3.5. Fig. 7 shows the observed and estimated execution times by DP and PGA with 16 processors. Here, execution time for 100 generations is shown for PGA. The conventional DP always produces optimal results. On the other hands, optimal solutions are also obtained by the proposed PGA even when the number of introduced generation units increases. However, a probability to obtain optimal solutions decreases as the number of the generation units increases. It is only 7 % for the practical case with 91 generation units after 100 trials. Here, it should be noted that the results by the proposed method are the best results after only 100 generations. It takes over one week by DP to obtain optimal

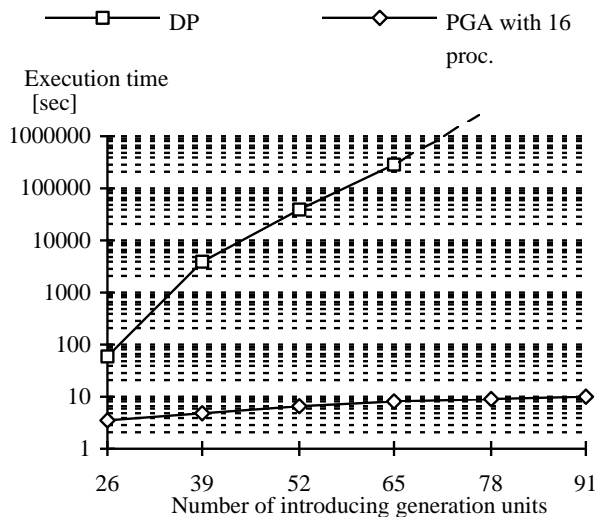


Fig. 7 Observed and estimated execution times for the number of introduced generation units.

results for the case with 91 generation units. Therefore, it may take several months to a year to obtain the results when considering various scenarios. On the contrary, the proposed method only takes around 10 seconds to get the results. It can obtain optimal results after 7 trials on average and it only takes around one minute. To sum up, the proposed method produces the results efficiently even when the conventional method takes a long time to get the results and it can not be applied practically. Moreover, Since the proposed method generates the results which always satisfy the constraints, the results are always feasible even if they are not optimal. Namely, the proposed method expands the range of practical generation expansion planning which can be solved by planning methods.

CONCLUSIONS

This paper presents a parallel genetic algorithm (PGA) for a generation expansion planning problem. The PGA belongs to the class of coarse-grain PGA in order to achieve the trade-off between computational speed and hardware cost. The PGA is implemented on transputer. The method is compared favorably with dynamic programming and the conventional genetic algorithm. Moreover, Binary and decimal coding, and several selection methods are compared. The method is applied to a typical test system with 4 technologies, 5 intervals, and various numbers of introduced generation units. The results show feasibility, and fast and accurate optimization ability of the proposed method. The results can be summarized as follows:

- o Coarse-grain PGA with decimal coding and RSWR are suitable for a long-range generation expansion planning problem.
- o The proposed method can search the solutions in the feasible region in parallel and efficiently. The execution time of the proposed method is almost proportional to the number of introduced generator units and it can produce the optimal results with high probability. Therefore, the proposed method can be a basic tool based on a deterministic approach for long-range generation expansion planning.
- o Transputer can be implemented on EWS, which are widely used. Therefore, the results indicate the possibility of using multi-processors on EWS independently for solving optimal long-range generation expansion planning in an actual use.

REFERENCES

- [1] G. J. Anders, "Generation Planning Model with Reliability Constraints", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-100, No. 12, December 1981.
- [2] M. C. Caramanis, et al., "The Introduction of Non-dispatchable Technologies as Decision Variables in Long-term Generation Expansion Models", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-101, No. 8, August 1982.
- [3] N. Levin and J. Zahavi, "Optimal Mix Algorithms with Existing Units", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-103, No. 5, May 1984.
- [4] W. D. Dapkus and T. R. Bowe, "Planning for New Electric Generation Technologies - A Stochastic Dynamic Programming Approach", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-103, No. 6, June 1984.
- [5] Y. M. Park, K. Y. Lee, and L. T. Youn, "New Analytical Approach for Long-term Generation Expansion Planning Based on Maximum Principle and Gaussian Distribution Function", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-104, No. 2, February 1985.
- [6] H. T. Yang and S. L. Chen, "Incorporating a Multi-Criteria Decision Procedure into the Combined Dynamic Programming / Production Simulation Algorithm for Generation Expansion Planning", *IEEE Trans. on Power Systems*, Vol. 4, No. 1, pp. 165-175, February 1989.
- [7] A. K. David and Z. Rong-da, "Integrating Expert Systems with Dynamic Programming in Generation Expansion Planning", *IEEE Trans. on Power Systems*, Vol. 4, No. 3, pp. 1095-1101, August 1989.
- [8] A. K. David and Z. Rong-da, "An Expert System with Fuzzy Sets for Optimal Planning", *IEEE Trans. on Power Systems*, Vol. 6, No. 1, pp. 59-65, February 1991.
- [9] B. Mo, et al., "Stochastic Generation Expansion Planning by Means of Stochastic Dynamic Programming", *IEEE Trans. on Power Systems*, Vol. 6, No. 2, pp. 662-668, May 1991.
- [10] B. G. Gorenstin, et al., "Power System Expansion Planning under Uncertainty", *IEEE Trans. on Power Systems*, Vol. 8, No. 1, pp. 129-136, February 1993.
- [11] T. Tanabe, et al., "Flexible Generation Mix under Multi Objectives and Uncertainties", *IEEE Trans. on Power Systems*, Vol. 8, No. 2, May 1993.
- [12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- [13] J. P. Cohoon, et al., "Punctual Equilibria: A Parallel Genetic Algorithm", *Proc. of second International Conference on Genetic Algorithms*, 1987.
- [14] R. Tanese, "Parallel Genetic Algorithm for a Hypercube", *Proc. of second International Conference on Genetic Algorithms*, 1987.
- [15] F. Hoffmeister, "Scalable Parallelism by Evolutionary Algorithms", in *Parallel Computing and Mathematical Optimization* edited by M. Grauer and D. B. Presmaer, Springer-Verlag 1990.
- [16] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization", *Proc. of third International Conference on Genetic Algorithms*, 1989.
- [17] M. G. Schleuter, "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy", *Proc. of third International Conference on Genetic Algorithms*, 1989.
- [18] K. Yasuda, et al., "Optimal Generation Expansion Planning based on the Dynamic Programming", *Bulletin of the Faculty of Engineering, Hokkaido University*, No. 142 1988 (in Japanese).

YOSHIKAZU FUKUYAMA received the B.S. and M.S. degrees in electrical engineering in 1985 and 1987, respectively, from Waseda University, Tokyo, Japan. He has been working at Fuji Electric Corporation Research and Development, Ltd. Japan.

He is currently a visiting scientist at Cornell University and working with Prof. Hsiao-Dong Chiang. His research interests include application of neural network, expert system, and genetic algorithms to electric power systems. He is a member of IEEE and IEE of Japan.

HSIAO-DONG CHIANG received the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley in 1986. He is currently an associate professor of electrical engineering at Cornell University. He was a recipient of the Engineering Research Initiation Award (1988) and of the Presidential Young Investigator Award (1989) both from the National Science Foundation. In 1990, he was selected by a Cornell Merrill Presidential Scholar as the faculty member who had the most positive influence on that student's education at Cornell. He was an Associate Editor of the IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications.

His research interests include power systems, nonlinear systems, optimization theory, and neural networks.

The procedure of the proposed parallel genetic algorithm for solving the generation expansion problem is summarized in Fig. 4. It consists of five procedures. Only the migration procedure is added to the conventional GA.

The proposed PGA has been implemented on transputer, that is one of the MIMD (multi instruction stream and multi data stream) processors. The concept of the coarse-grain PGA on transputer is shown in Fig. 5. In order to realize the coarse-grain PGA, the total population is distributed into several sub-populations. Each sub-population is allocated to each process and the conventional GA is performed using each sub-population on each process. Strings with highest fitness values are migrated from the neighboring processes at every epoch. The processes are installed on the actual transputer