Journal of the Operations Research
Society of Japan
Vol. 40, No. 1, March 1997



# EVOLUTION ALGORITHM FOR OPTIMAL CAPACITY EXPANSION

Mitsuo Gen
*Ashikaga Institute of Technology*

Baoding Liu
*Tsinghua University*

(Received July 20, 1994; Revised June 26, 1995)

*Abstract* This paper presents an evolution algorithm for optimal capacity expansion problem which is related to determining when and how much capacity should be added under varying types of demand and cost. To avoid premature convergence and stalling of the evolutionary process, we employ an exponential-fitness scaling scheme. To improve the chromosomes, we introduce hetero-dimensional mutation which generates a new dimension and produces a feasible solution, and homo-dimensional mutation which mutates the chromosomes in the negative of gradient (subgradient) direction. Some numerical examples are provided to illustrate the effectiveness of the proposed evolution algorithm.

## 1. Introduction

Capacity expansion problems have been studied for over thirty years and applied in a wide variety of areas, for example, heavy industries, communication networks, electrical power services and water resource systems. The capacity expansion problem is to determine *when* and *how much* capacity should be added under varying types of demand and cost. Usually, there exist two types of demand: deterministic and stochastic. The optimal capacity expansion with deterministic demand has been reported by Manne[13], Erlenkotter[2][3], Freidenfelds[6] and summarized by Luss[12]. Stochastic case has been considered by Freidenfelds[5], Buzacot and Chaouch[1] and Erlenkotter *et al.*[4], *etc.* In this paper we will restrict our attention to deterministic case with continuous time.

Evolution algorithms are a type of stochastic search methods and have been used widely for numerical optimization problem(Goldberg[10], Michalewcz[14], Gen and Cheng[16]), such as optimal control problems, transportation problems, traveling salesman problems, drawing graphs, scheduling and machine learning. Evolution algorithms are also studied by several researchers and summarized by Fogel[7]. As a research work related closely with this paper, Gen and Liu[9] present an evolution algorithm for continuous time production plan problem which is concerned with determining a rate of production under varying types of demand and cost.

In this paper we present an evolution algorithm for solving optimal capacity expansion problem. To avoid premature convergence and stalling of the evolutionary process which usually occurs in traditional genetic methods, we employ an exponential-fitness scaling scheme proposed by Gen *et al.*[8]. To improve the chromosomes, we introduce hetero-dimensional mutation which generates a new dimension and produces a feasible solution, and homo-dimensional mutation which mutates the chromosomes in the negative of gradient (subgradient) direction. Finally, we illustrate the effectiveness of the proposed evolution algorithm by some numerical examples.

## 2. Optimal Capacity Expansion

The criterion used here for determining the best capacity policy is to minimize the discounted costs associated with the expansion process, including costs for expansions, shortages, congestion, idle capacity, maintenance and inventory, taking into physical constraints.

In this paper, we consider the continuous time case as shown in Figure 1.

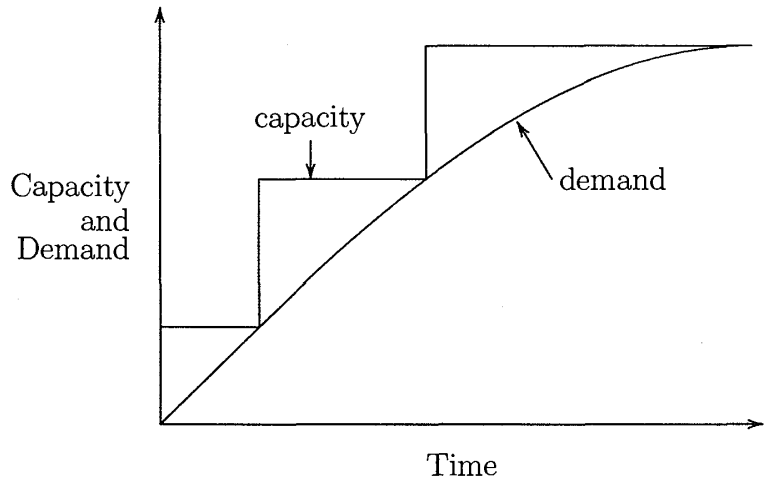


Figure 1: Capacity Expansion Problem

Figure 1 illustrates the curve of demand $d(t)$ and expansion of capacity over time under the following simplifying assumptions:

(i) the equipment has an infinite economic life;

(ii) whenever the demand catches up with the existing capacity, $x$ units of new capacity are installed; and

(iii) the planning horizon is transacted after an arbitrary finite time $T$.

The installation cost that results from a single capacity increment of size $x$ at time $t$ is assumed to be given by a function $c(x,t)$ which is increasing in $x$ and decreasing in $t$.

Our problem is related to determining an integer $n$ and $0 = t_1 < t_2 < \cdots < t_n < T$ such that the total installation cost is minimized, i.e.,

$$\min_n \min_{0=t_1<t_2<\cdots<t_n<T} \sum_{i=1}^{n} c(d(t_{i+1}) - d(t_i), t_i) \tag{2.1}$$

No computing method for the optimal capacity expansion problem (2.1) have been reported in literature. From the basic equation (2.1), we see that this problem is very different from the classical optimization because both of the dimension $n$ and decision vector $\mathbf{t} = (t_1, t_2, \cdots, t_n)$ are variables, while the classical cases have a fixed dimension. In fact, it is very difficult to apply a traditional approach to such a problem. This is our basic motive to develop an evolution algorithm.

## 3. Evolution Algorithm

In this section we present an evolution algorithm to solve the optimal capacity expansion problem. We will discuss representation structure, initialization process, evaluation function, selection, mutation and crossover operations.

### 3.1. Representation Structure

We use an integer $n$ and a vector $V = (x_1, x_2, \cdots, x_n)$, $\epsilon \leq x_i \leq 1$, as a chromosome to represent a solution $(t_1, t_2, \cdots, t_n)$ to the optimal capacity expansion problem, where $\epsilon$ is an appreciate small positive number. The mapping from a chromosome $(x_1, x_2, \cdots, x_n)$ into a solution $(t_1, t_2, \cdots, t_n)$ can be written as follows:

$$\begin{aligned} t_1 &= 0 \\ t_i &= \frac{x_1+x_2+\cdots+x_{i-1}}{x_1+x_2+\cdots+x_n} \cdot T, \qquad i = 2, 3, \cdots, n \end{aligned} \tag{3.1}$$

which ensures that the solution $(t_1, t_2, \cdots, t_n)$ is always feasible.

### 3.2. Initialization Process

We define an integer *pop_size* as the number of chromosomes. *pop_size* chromosomes will be randomly initialized by the following way:

To determine a chromosome, we first determine its dimension, *i.e.*, the number of capacity expansions. Usually, the range for possible $n$ can be given by some priori knowledge. So we can generate a random integer $n$ within that range, then $n$ random numbers $x_1$, $x_2$, $\cdots$, $x_n$ on the interval $[\epsilon, 1]$ to obtain an initial chromosome $(x_1, x_2, \cdots, x_n)$. Repeat this process *pop_size* times and produce *pop_size* initial feasible chromosomes.

### 3.3. Evaluation Function

Let $u$ denote the original fitness, *i.e.*, the objective value. Usually, the original fitness proportionate-reproduction scheme frequently causes two significant difficulties: premature convergence termination at early generations, and stalling at late generations. To overcome these two problems, Goldberg[10] suggests a linear-fitness scaling scheme, $u' = au + b$. Lee and Johnson[11] suggests a modified linear-fitness scaling scheme, *i.e.*, for the lower-fitness strings, the minimum original fitness $u_{\min}$ maps to 0; for the higher-fitness strings, the maximum original fitness $u_{\max}$ maps to a parameter $f_{\text{multiple}}$ which is greater than 1; the average fitness $u_{\text{avg}}$ maps to 1. Michalewicz[14] suggests a power law scaling scheme, $u' = u^k$, where $k$ is a real number which is close to one.

Here we will employ an exponential-fitness scaling scheme suggested by Gen *et al.*[8]. We define three preference parameters $p_1$, $p_0$ and $p_2$ $(0 < p_1 < p_0 < p_2 < 1)$ which can determine three critical numbers $u_1$, $u_0$ and $u_2$ such that there are $(p_1 \cdot pop\_size)$, $(p_0 \cdot pop\_size)$ and $(p_2 \cdot pop\_size)$ chromosomes which are less than $u_1$, $u_0$ and $u_2$, respectively, in the set of the *pop_size* chromosomes at the current iteration.

For the optimal capacity expansion, the original fitness $u_1$ is set as $2 - e^{-1} \approx 1.63$, the original fitness $u_0$ is set as 1, and the original fitness $u_2$ is set as $e^{-1} \approx 0.37$. Then the relationship between the original fitness $u$ and the exponential fitness $u'$ is represented as follows:

$$u' = \begin{cases} 2 - \exp\left[-\frac{u-u_0}{u_1-u_0}\right], & u < u_0 \\ \exp\left[-\frac{u-u_0}{u_2-u_0}\right], & u \geq u_0 \end{cases} \tag{3.2}$$

which is shown in Figure 2.

Thus we can define the evaluation function as follows:

$$eval(V) = u' / \sum_{i=1}^{pop\_size} u'_i \tag{3.3}$$

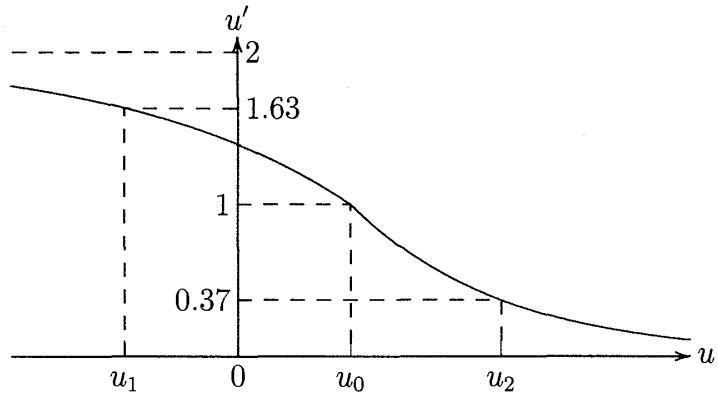


Figure 2: Exponential-fitness Function

where $V$ is a chromosome, $u'$ is the exponential fitness of $V$, and $u'_i$ is the exponential fitness of chromosome $V_i$, $i = 1, 2, \cdots, pop\_size$, respectively.

For the above mentioned evaluation function, the chromosome with higher fitness can have more chance to produce offspring.

### 3.4. Selection Process

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select a single chromosome for a new population as the following steps:

**Step 1:** Calculate the cumulative probability $q_i$ for each chromosome $V_i$,

$$\begin{aligned} q_0 &= 0 \\ q_i &= \sum_{j=1}^{i} eval(V_j), \qquad i = 1, 2, \cdots, pop\_size. \end{aligned} \tag{3.4}$$

**Step 2:** Generate a random real number $r$ in $[0, 1]$.

**Step 3:** Select the $i$-th chromosome $V_i$ $(1 \leq i \leq pop\_size)$ if $q_{i-1} < r \leq q_i$.

**Step 4:** Repeat Steps 2 and 3 *pop_size* times and obtain *pop_size* copies of chromosomes.

### 3.5. Crossover Operation

We define a parameter $P_c$ of an evolution system as the probability of crossover. This probability gives us the expected number $P_c \cdot pop\_size$ of chromosomes which undergo the crossover operation.

First we generate a random real number $r$ in $[0, 1]$; secondly, we select the given chromosome for crossover if $r < P_c$. Repeat this operation *pop_size* times and produce $P_c \cdot pop\_size$ parents, averagely.

For each pair of parents (vectors $V_1$ and $V_2$), if they have the same dimension, the crossover operator on $V_1$ and $V_2$ will produce two children $X$ and $Y$ as follows:

$$X = c_1 \cdot V_1 + c_2 \cdot V_2, \qquad Y = c_2 \cdot V_1 + c_1 \cdot V_2 \tag{3.5}$$

where $c_1$, $c_2 \geq 0$ and $c_1 + c_2 = 1$. Otherwise, skip the crossover operation.

Since the constraint set is convex, this arithmetical crossover operation ensures that both children are feasible if both parents are.

### 3.6. Mutation Operation

Two types of mutation operators are defined in this evolution algorithm. The first, *hetero-dimensional mutation*, generates a new dimension and then produces a feasible solution. The second, *homo-dimensional mutation*, mutates the chromosomes in the negative gradient (subgradient) direction.

It is well-known that the Taylor's expansion of a continuous differentiable function $f$ is

$$f(\mathbf{x}+\triangle\mathbf{x}) = f(\mathbf{x}) + \bigtriangledown f(\mathbf{x}+\theta\triangle\mathbf{x})^{\mathrm{T}}\triangle\mathbf{x}, \qquad \mathbf{x}\in \mathrm{R}^n \tag{3.6}$$

where $0 \le \theta \le 1$, $\bigtriangledown f(\mathbf{x})$ denotes the gradient of the function $f$ at the point $\mathbf{x}$, $\triangle\mathbf{x}$ is a small perturbation in $R^n$. So for the optimal capacity expansion, it may be better to choose the negative direction of gradient as the mutation direction.

Because of the complexity of the problem, we will calculate the $i$th component of the gradient (regardless of the existence) approximately by

$$\frac{f(x_1,\cdots,x_i+\triangle x_i,\cdots,x_n) - f(x_1,\cdots,x_i,\cdots,x_n)}{\triangle x_i} \tag{3.7}$$

where $\triangle x_i$ is a small real number.

We define two parameters $P_{m1}$ and $P_{m2}$ as the probabilities of hetero-dimensional mutation and homo-dimensional mutation. Then the two probabilities give us the expected numbers $P_{m1}\cdot pop\_size$ and $P_{m2}\cdot pop\_size$ of chromosomes which undergo the hetero-dimensional mutation and homo-dimensional mutation, respectively.

Generating a random real number $r$ in $[0,1]$, we select the given chromosome for hetero-dimensional mutation and homo-dimensional mutation if $r < P_{m1}$ and $P_{m1} < r < P_{m1}+P_{m2}$, respectively. Suppose that a parent denoted by a vector $V = (x_1, x_2, \cdots, x_n)$, is selected. For hetero-dimensional mutation, we generate a new dimension $n+1$ and then produce an $n$+1-dimensional feasible solution by the procedure like initialization. For homo-dimensional mutation, we choose a direction $\mathbf{d}$ which is a negative direction of the approximate gradient and a large positive number $M$ which ensures the operator is probabilistically complete, if $V + M\cdot\mathbf{d}$ is not feasible, then we set $M$ as a random number between 0 and $M$ until it is feasible, thus the offspring is

$$V' = V + M\cdot\mathbf{d} \tag{3.8}$$

Repeat this operation $pop\_size$ times and produce $(P_{m1}+P_{m2})\cdot pop\_size$ offspring, averagely.

### 3.7. Algorithm

Following selection, crossover and mutation, the new population is ready for its next evaluation. The evolution algorithm will terminate after a given number of cyclic repetitions of the above steps. We can summarize the evolution algorithm for the optimal capacity expansion problem as follows:

**Step 0: Set parameters**

*input max_gen; //numbers of generations*
*pop_size; //population size*
*$P_{m1}$; //probability of hetero-dimensional mutation*
*$P_{m2}$; //probability of homo-dimensional mutation*
*$P_c$; //probability of crossover*
*$p_1, p_0, p_2$; //preference parameters*

**Step 1: Initialization process**
  **for** $i = 1$ **to** *pop_size* **do**
    *produce a random integer* $n$;
    *produce* $n$ *random real numbers* $x_1, x_2, \cdots, x_n$ *on* $[\epsilon, 1]$;
    $V_i = (x_1, x_2, \cdots, x_n)$;
  **end**
**Step 2: Evaluation**
  **for** $i = 1$ **to** *pop_size* **do**
    *compute the objective* $u_i$ *for* $V_i$;
  **end**
  **for** $i = 1$ **to** *pop_size* **do**
    *compute the exponential-fitness* $u'_i$;
  **end**
  **for** $i = 1$ **to** *pop_size* **do**
    *compute the cumulative probabilities* $q_i = \sum_{j=1}^{i} u'_j / \sum_{j=1}^{pop\_size} u'_j$;
  **end**
**Step 3: Selection operation**
  **for** $i = 1$ **to** *pop_size* **do**
    **if**$(q_{i-1} < random() \leq q_i)$ **then**
      *select* $V_i$;
    **end**
  **end**
**Step 4: Crossover operation**
  **for** $i = 1$ **to** *pop_size/2* **do**
    **if**$(random() \leq P_c)$ **then**
      *int* $j$ = *random(pop_size);*
      *int* $k$ = *random(pop_size);*
      *perform the crossover on* $j$*-th and* $k$*-th chromosomes;*
    **end**
  **end**
**Step 5: Mutation operation**
  **for** $i = 1$ **to** *pop_size* **do**
    *r=random();*
    **if**$(r \leq P_{m1})$ **then**
      *hetero_dimensional_mutation();*
    **else**
      **if**$(P_{m1} < r \leq P_{m1} + P_{m2})$ **then**
        *homo_dimensional_mutation();*
      **end**
    **end**
  **end**
**Step 6: Termination test**
  **if**$(number\ of\ iterations < max\_gen)$ **then**
    *goto Step 2;*
  **else**
    *stop;*
  **end**

## 4. Numerical Experiments

The evolution algorithm has been implemented in C language. We will use it to solve the following numerical example which is a modification of one in Odanaka[15].

We take the demand pattern as

$$d(t) = b \cdot \sin(\frac{t}{2T}\pi), \qquad 0 \leq t \leq T \tag{4.1}$$

which will be satisfied by the production capacity and the cost function as

$$c(q,t) = \begin{cases} 0, & q = 0 \\ \exp[-at] \cdot (k + e \cdot q), & q > 0 \end{cases} \tag{4.2}$$

which results from a single capacity expansion of size $q$ at time $t$.

In this numerical example, we set the parameters as $b = 10$, $T = 100$, $a = 0.06$, $k = 10$, $e = 10$.

In our experiment, the population size is 50, the probability of crossover is 0.1, the probability of hetero-dimensional mutation is 0.1, the probability of homo-dimensional mutation is 0.4, the preference parameters $p_1$, $p_0$ and $p_2$ are defined as 0.1, 0.5 and 0.9, respectively.

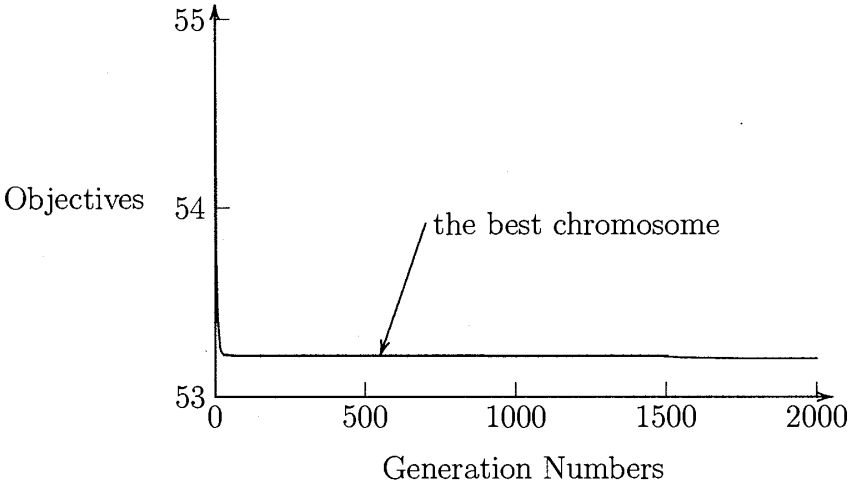


Figure 3: Progress of Evolution

Figure 3 shows that in 10 iterations the best solution of problem is 53.594822; in 100 iterations the best solution is 53.540710; in 500 iterations the best solution is 53.510918; in 1000 iterations the best solution is 53.510906; in 2000 iterations the best solution is 53.510899 with six time expansions, the policy is to add 1.995217 at the beginning, 1.959452 at time 12.787778, 1.870075 at time 25.883461, 1.714847 at time 39.583107, 1.459610 at time 54.371712, and 1.000799 at time 71.275131. The process of capacity expansion is shown by Figure 4 in which the step-like function represents the production capacity and the curve represents the demand. In performing the algorithm with 1000 generations, the CPU time is 76.3 seconds on NEC EWS4800/210II workstation.

The proposed evolution algorithm has been performed on a lot of optimal capacity expansion problems for different parameter values. For example, when we set the parameters

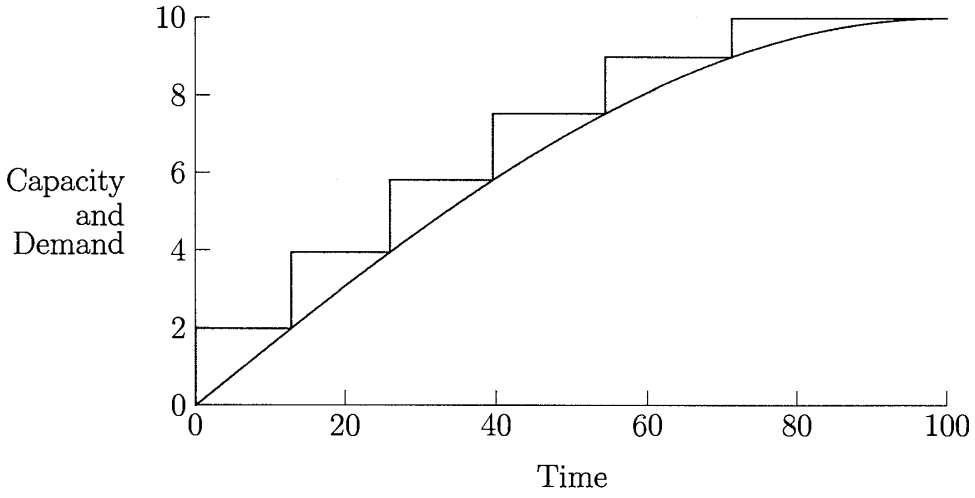


Figure 4: Process of Capacity Expansion

as $b = 15, T = 100, a = 0.05, k = 20, e = 15$, every run of our evolution algorithm with 2000 iterations shows that the optimal cost is 128.452 with six time expansions at

$$(t_1, t_2, t_3, t_4, t_5, t_6) = (0.000, 13.398, 27.103, 41.447, 56.850, 74.252).$$

If we set $b = 8, T = 100, a = 0.05, k = 20, e = 8$, every run of our evolution algorithm with 2000 iterations shows that the optimal cost is 59.293 with three time expansions at

$$(t_1, t_2, t_3) = (0.000, 23.516, 49.835).$$

## 5. Conclusion

In this paper we present an evolution algorithm for solving optimal capacity expansion problem. Usually, the fitness proportionate-reproduction scheme frequently causes two significant difficulties: premature convergence at early generations, and stalling at the late generations. We employed an exponential-fitness scaling scheme to overcome the two problems as above.

To improve the chromosomes, we introduce hetero-dimensional mutation which generates a new dimension and produces a feasible solution, and homo-dimensional mutation which mutates the chromosomes in the negative of gradient (subgradient) direction.

Finally, we discuss a numerical example which shows that the evolution algorithm is an effective algorithm to the optimal capacity expansion problem. For optimization problems, the time necessary for an algorithm to converge to the optimum depends on the number of decision variables, the time complexity of most algorithms can increase geometrically with the number of variables, however, the time complexity of the proposed evolution algorithm increases linearly as the dimension increases.

## Acknowledgement

This work was supported in part by a research grant from the Ministry of Educaton, Science and Culture, Japanese Government: Grant-in-Aid for Scientific Research Program(No. 07045032).

## References


[1] Buzacot J.A. and A.B. Chaouch, Capacity expansion with interrupted demand growth, *European Journal of Operational Reseach*, 34, 19-26, 1988.

[2] Erlenkotter D., Sequencing expansion projects, *Operations Research*, 21, 542-553, 1973.

[3] Erlenkotter D., Capacity expansion with imports and inventories, *Management Science*, 23, 694-702, 1977.

[4] Erlenkotter D., S. Sethi and N. Okada, Planning for surprise: water resources development under demand and supply uncertainty I., the general model, *Management Science*, 35, 149-163, 1989.

[5] Freidenfelds J., Capacity expansion when demand is a birth-death process, *Operations Research*, 28, 712-721, 1980.

[6] Freidenfelds J., *Capacity Expansion, Analysis of Simple Models with Applications*, Elsevier, New York, 1981.

[7] Fogel D.B., An introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks*, 5, 3-14, 1994.

[8] Gen M., B. Liu and K. Ida, Evolution program for deterministic and stochastic optimizations, in press, *European Journal of Operational Research*, 94, 618-625, 1996.

[9] Gen M. and B. Liu, Evolution program for production plan problem, *Engineering Design and Automation*, 1, 199-204, 1995.

[10] Goldberg D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[11] Lee J. and G.E. Johnson, Optimal Tolerance Allotment Using a Genetic Algorithm and Truncated Monte Carlo Simulation, *Computer-Aided Design*, 25, 601-611, 1993.

[12] Luss H., Operations research and capacity expansion problems: a survey, *Operations Research*, 30, 907-947, 1982.

[13] Manne A.S., *Investment for Capacity Expansion: Size, Location and Time Phasing*, MIT Press, Cambridge, 1967.

[14] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs, Second Edition,* Springer-Verlag, New York, 1994

[15] Odanaka, T., *Dynamic Management Decision and Stochastic Control Processes*, World Scientific, Singapore, 1990.

[16] Gen, M. and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, Inc., New York, 1997.



Mitsuo GEN
Department of Industrial and Systems Engineering
Graudate School of Engineering
Ashikaga Institute of Technology
Ashikaga 326, Japan
Email: gen@genlab.ashitech.ac.jp

Baoding LIU
Department of Applied Mathematics
Tsinghua University
Beijin, 100084 China