Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computers & Industrial Engineering 48 (2005) 799–809
computers & industrial engineering
www.elsevier.com/locate/dsw

# Hybrid genetic algorithm for multi-time period production/distribution planning

Mitsuo Gen[a,*], Admi Syarif[b,c]

[a]*Graduate School of Information Production and System, Waseda University, Kitakyushu 808-0135, Japan*
[b]*Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology, Ashikaga 326-8558, Japan*
[c]*Department of Mathematics, Lampung University, Bandar Lampung 35145, Indonesia*

**Abstract**

In this paper, we deal with a production/distribution problem to determine an efficient integration of production, distribution and inventory system so that products are produced and distributed at the right quantities, to the right customers, and at the right time, in order to minimize system wide costs while satisfying all demand required. This problem can be viewed as an optimization model that integrates facility location decisions, distribution costs, and inventory management for multi-products and multi-time periods. To solve the problem, we propose a new technique called spanning tree-based genetic algorithm (hst-GA). In order to improve its efficiency, the proposed method is hybridized with the fuzzy logic controller (FLC) concept for auto-tuning the GA parameters. The proposed method is compared with traditional spanning tree-based genetic algorithm approach. This comparison shows that the proposed method gives better results.


*Keywords:* Production/distribution problem; Logistics; Optimization; Genetic algorithms; Prüfer number

## 1. Introduction

To be competitive in the global manufacturing environment, the strategic logistic planning becomes one of important factors. The problem of effectively designing logistic system involves the design of production/distribution and inventory. There is a large number of literature dealing with various variation production/distribution problems. In most cases, they consider an overall production strategy, inventory strategy and flow of products through a facility over in a single period of time in order to minimize cost or maximize profit. An example of a good formulation and discussion for location

* Corresponding author. Fax: +81 93 692 5021.
*E-mail addresses:* gen@waseda.jp (M. Gen), admi@genlab.ashitech.ac.jp (A. Syarif).

0360-8352/$ - see front matter © 2004 Published by Elsevier Ltd.
doi:10.1016/j.cie.2004.12.012

allocation planning for multi-commodity problem is given by Pirkul and Jayaraman (1998). In real-world applications, however, this planning problem exists because there are limited production resources that cannot be stored from period to period. This gives rise to a good production/distribution and inventory planning model with multi-time period (mt-PDI) (Graves, 1999). In this model, the objective is to design an efficient production/distribution and inventory strategy so that products are produced and distributed at the right quantities, to the right customers, and at the right time, in order to minimize system wide costs while satisfying all demands required. This problem can be viewed as an optimization model that integrates facility location decisions, transportation and distribution costs, and inventory management for multi-products and multi-time periods. In this problem, the author considers multiple products with independent demand, multiple shared resources and multi-time periods. Fig. 1 illustrates this problem.

In order to solve the multi-time period production/distribution and inventory problem (mt-PDI), this paper develops a technique called hybridized spanning tree-based genetic algorithm (hst-GA). This method differs from other GA methods mainly several aspects as follows: we utilize the Prüfer number (Prüfer, 1918) encoding based on a spanning tree which is adopted as it is capable of equally and uniquely representing all possible trees. Though there are several arguments for this coding technique for constrained network problems, in our pervious work we have shown the applicability and the efficiency of the Prüfer number representation for various network problems (Gen & Cheng, 1997; Gen, Cheng, & Oren, 2001). Further, to improve the performance of the algorithm, here, we also adopt the automatic fine tuning for the crossover ratio and mutation ratio using fuzzy logic controller (FLC) (Wang, Wang, & Hu, 1997). The effectiveness and efficiency of the proposed method are evaluated by comparing several numerical experiment results of the proposed method with those of traditional method.

The rest of this paper is organized as follows: the mathematical formulation of this problem is given in Section 2. In Section 3, we give a brief discussion of basic concepts and searching procedures for the proposed method (hst-GA). Those include the representation of chromosome, genetic operations, evaluation method, selection method and the FLC concept to make auto-tuning of the GA parameters. In Section 4, numerical experiments and comparison with the results of traditional algorithm are presented to demonstrate the efficiency of the proposed method. Finally, some concluding remarks are given in Section 5.

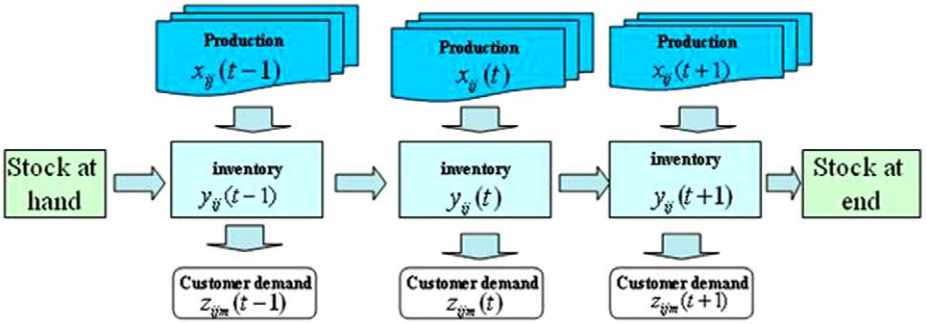


Fig. 1. The illustration of mt-PDI problem.

## 2. Mathematical model

Now we present a comprehensive mathematical model that considers real-world factors and constraints of the problem. Firstly, the notations used in the model are defined. The mathematical model with its objective function and constraints will be expressed later.

*Indices*

$t$ index of time period ($t=1,2,\ldots,T$)
$i$ index of product ($i=1,2,\ldots,I$)
$j$ index of plant ($j=1,2,\ldots,J$)
$k$ index of resource ($k=1,2,\ldots,K$)
$m$ index of customer ($m=1,2,\ldots,M$)

*Parameters*

$a_{ijk}$ amount of resource $k$ required to produce one unit of product $i$ at plant $j$
$b_{jk}(t)$ amount of resource $k$ available at plant $j$ in period $t$
$d_{im}(t)$ demand for product $i$ by customer $m$ in period $t$
$p_{ij}(t)$ unit cost of production for product $i$ at plant $j$ in period $t$
$q_{ij}(t)$ unit inventory holding cost for product $i$ at plant $j$ in period $t$
$c_{ijm}(t)$ shipping cost of product $i$ from plant $j$ to customer $m$ in period $t$

*Variables*

$x_{ij}(t)$ amount of product $i$ produced at plant $j$ in period $t$
$y_{ij}(t)$ inventory product $i$ at plant $j$ in period $t$
$z_{ijm}(t)$ amount of product $i$ shipped from plant $j$ to customer $m$ in period $t$

In this problem, we want to determine the production number for each product in each plant, inventory strategy and distribution network design to satisfy the resource capacity and customer demand with minimum cost. It can be formulated as follows:

$$\min \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij}(t)x_{ij}(t) + \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J} q_{ij}(t)y_{ij}(t) + \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M} c_{ijm}(t)z_{ijm}(t) \tag{1}$$

s.t.

$$y_{ij}(t-1) + x_{ij}(t) - \sum_{m=1}^{M} z_{ijm}(t) = y_{ij}(t) \quad \forall\, i,j,t \tag{2}$$

$$\sum_{j=1}^{J} z_{ijm}(t) = d_{im}(t), \quad \forall\, i, m, t \tag{3}$$

$$\sum_{i=1}^{I} a_{ijk} x_{ij}(t) \leq b_{jk}(t), \quad \forall\, j, k, t \tag{4}$$

$$x_{ij}(t), y_{ij}(t), z_{ijm}(t) \geq 0 \tag{5}$$

In the above model, the objective function captures production and inventory holding costs, which depend on the plant, plus transportation or distribution cost for shipment of product from plant to the customer. The constraint (2) is the inventory balance constraint that assures the supply of an item at each plant is either held in inventory or shipped to a customer to meet demand. Constraint (3) ensures that the shipments satisfy the demand of each customer for each period. Constraint (4) is a set of resource constraints. Production in each period is limited by the availability of a set of shared resources. Typical resources are various types of labor, process and material handling equipment and transportation modes.

## 3. Design of the algorithm

The genetic algorithm (GA) that solves problems using the Darwinian concept was first introduced by Holland (1975). GA is very useful when a large search space with little knowledge of how to solve the problem is presented. It belongs to the class of heuristic optimization techniques, which include simulated annealing, Tabu search, and evolutionary strategies. It has been with great success in providing optimal or near optimal solution for many diverse and difficult problems. It is noted by Davis (1991) that GA consists of five components as follows:

1. Chromosome representation;
2. A means of creating the initial population;
3. A measure function or criterion to be used to evaluate the fitness of chromosome;
4. A genetic operator that can create the next generation;
5. A way to set up the GA parameters, e.g. population size, crossover probability, mutation probability, etc.

### *3.1. Spanning tree-based genetic algorithm*

To solve the problem using GA, the method of representing the candidate solution in coding space is very important. Usually different problems have different data structures or genetic representations. For network problems that their solution has spanning tree topologies, spanning tree-based representation would be appropriate. Basically, there are three ways of encoding a spanning tree: (1) edge encoding, (2) vertex encoding and (3) edge-and-vertex encoding (Lo & Chang, 2000).

One of the classical theorems in graphical enumeration is Cayley's theorem (Dossey, Otto, Spence, & Eynden, 1993). Prüfer (1918) gave the simple proof of Calley's formula by establishing a one-to-one correspondence between the set of spanning trees and $N-2$ digit integer number with each integer between 1 and $N$ inclusive. The $N-2$ digit number later is known as Prüfer number. This Prüfer number

encoding procedure belongs to the class of vertex encoding. The node that appears $d$ times in the Prüfer number will have $d+1$ arc connected.

*3.1.1. Prüfer number representation*

For mt-PDI problem, each chromosome is represented by using $T$ Prüfer numbers. Each Prüfer number is performed from randomly generated $J+M-2$ digit in the range $[1, J+M]$. It represents the network graph solution of the problem with $J$ plants and $M$ customers in each time period. Before decoding a Prüfer number, we should check its feasibility. The Prüfer number $P(T)$ is said to be feasible if the total number of arcs connected to the plant nodes is equal to the total number of arcs connected to the customer nodes. Let $L_i$, $i \in \{1,2,\ldots,J+M\}$ is the appearance number of node $i$ in Prüfer number $P(T)$. Thus a feasible Prüfer number should satisfy the following criteria:

$$\sum_{i=1}^{m} L_i + 1 = \sum_{i=m+1}^{m+n} L_i + 1 \tag{6}$$

The Prüfer number that does not satisfy the above criteria is called infeasible Prüfer number. To handle with an infeasible Prüfer number, Gen and Cheng (2000) reject the chromosome and generate a new chromosome that satisfies the criteria. However, this technique will take a large computational time especially when the difference of the suppliers/plants and destination/customers numbers is very large. It only works for relatively small size problems. In our previous work, we introduced a repairing procedure for an infeasible Prüfer number as follows (Syarif & Gen, 2003):

Step 1: Generate the Prüfer number.
Step 2: Determine the number of appearance for each node in the Prüfer number.
Step 3: If the criterion of feasibility is satisfied then stop. Otherwise go to step 4.
Step 4: If the total number of arcs connected to the plant nodes is greater than the total number of arcs connected to the customer nodes, then select one digit in the Prüfer number which belongs to the plant node and replace it with the customer node. Otherwise select one digit in the Prüfer number which belongs to the customer node and replace it with the plant node. Go to step 2.

After checking its feasibility, we can determine a unique shipment strategy to satisfy the customer demands from each plant for each product by decoding the Prüfer number. For each time period $t$ and specific product $i$, if the number of product demand to plant is larger than its inventory then that product should be produced in that plant.

Procedure: convert Prüfer number to transportation tree

Step 1: Let $P(T)$ be the original Prüfer number and let $\bar{P}(T)$ be the set of all nodes that are not included in $P(T)$.
Step 2: Repeat the process (2.1)–(2.5) until no digits are left in $P(T)$.
(2.1) Let $j$ be the lowest numbered eligible node in $\bar{P}(T)$. Let $m$ be the leftmost digit in $P(T)$.
(2.2) If $j$ and $m$ are not in the same set $O$ or $D$ ($O$ and $D$ are the set of plant nodes and customer nodes respectively) then add the edge $(j, m)$ to tree $T$. Otherwise, select the next digit $n$ from $P(T)$ that is not included in the same set with $j$, exchange $m$ with $n$, add the edge $(i, k)$ to the tree $T$.

(2.3) Remove $m$ from $P(T)$ and $j$ from $\bar{P}(T)$. If $m$ does not occur anywhere in the remaining part of $P(T)$, put it into $\bar{P}(T)$.
(2.4) Determine the amount of product $i$ shipped from plant $j$ to customer $m$.
(2.5) Update the availability of resource capacity, inventory and demand.

Step 3: If no digits remain in $P(T)$ then there are exactly two nodes, $j$ and $m$, still eligible in $\bar{P}(T)$ for consideration. Add edge $(j, m)$ to tree $T$ and form a tree with $J+M-1$ edges.

Step 4: If all customer demands are satisfied then stop. Otherwise, assign that customer to one of the plant and return to Step 2.4.

After decoding all Prüfer numbers, we can determine the shipment strategy for each product in each time period and the necessity of production for each product in each plant for each time period. In this sense, if there exists remaining resources in the plants, they can be used for manufacturing additional quantities of the products and store them as inventory. The selection of the products to be manufactured here is done randomly.

If there are still available resources in the plants, use them for producing products and store as the inventory for the next time period in those plants.

### *3.1.2. Initialization*

As the initial population, we generate population size (*pop_size*) chromosomes. Each individual chromosome in the initial population is a candidate solution to the problem. We use the complete random method to generate the initial population.

### *3.1.3. Genetic operators*

*Crossover*. The crossover operation is done by exchanging the genes between two chromosomes. For simplicity, we used the one-cut-point crossover operation. This crossover operation is done by selecting a cut point randomly and exchanging the right part between two parents. For avoiding an infeasible chromosome (a Prüfer number) that may be generated after the crossover operator, all offsprings are checked for their feasibility before being decoded into a corresponding spanning tree.

*Mutation*. The inversion mutation is used. The inversion mutation is done by selecting two positions within a chromosome at random and then inverts the substring between these two positions. Since the number of appearance for each node in the Prüfer number after this operation is the same, the offspring generated by this operation is always guaranteed to be feasible. Fig. 2 illustrates the inversion mutation process.

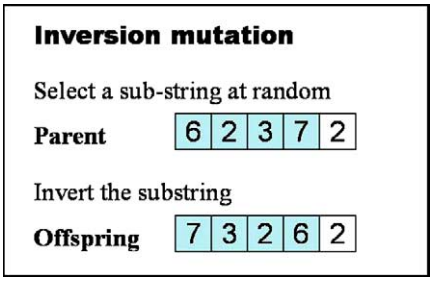


Fig. 2. The illustration of mutation operation.

*3.1.4. Evaluation*

The evaluation is done to measure the performance of the chromosome. Here we use the overall production, distribution and inventory cost value as the fitness function of the chromosome. It is done by the following steps:

Procedure: evaluation

Step 1: For each time period, decode Prüfer number into spanning tree;
Step 2: Determine production and distribution strategy based on the demand of customer in the spanning tree network;
Step 3: Use the available resource to produce one of product and store it in the inventory;
Step 4: Determine over all production cost, inventory cost and transportation cost.

*3.1.5. Selection*

The purpose of the selection method is to determine the chromosome that survives for the next generation. There have been many variants of selection method in the literature (Gen & Cheng, 1997). In traditional GA, a chromosome is selected for the next generation with certain probability. The best chromosome in the current generation may not be selected to the next generation. This causes a difficulty in reaching the optimal/near optimal solution. In other words it would take more computational time to get a good quality solution. In this paper, we use the mixed strategy with $\mu+\lambda$-selection and roulette wheel selection to enforce the best chromosomes into the next generation. This strategy selects $\mu$ best chromosomes from $\mu$ parents and $\lambda$ offspring. If there are no $\mu$ different chromosomes available, then the vacant pool of population is filled up with roulette wheel selection.

*3.2. Fuzzy logic controller*

When using GA, one of the important factors is a balance between exploitation and exploration in the search space. To provide this balance, determination of the design strategy for GA parameters such as population size, maximum generation, crossover probability and mutation probability is one of the critical issues. Due to its difficulty, several authors used fuzzy logic controller (FLC) for automatically tuning the GA parameters. FLC is acceptable because only incomplete knowledge and imprecise information are available for identification of the relation between the GA parameters and the behavior of GA. FLC provides an algorithm that can convert linguistic control strategy based on expert knowledge into an automatic control strategy. Lee and Takagi (1993) used the phenotypic diversity measure (PDM) for the best, average and worst of fitness to automatically tune crossover probability $p_C$ and mutation probability $p_M$ (Lee & Takagi, 1993). Zeng and Rabenasolo (1997) designed FLC for adjusting $p_C$, $p_M$ and the position for crossover operation by approximating the relationship between GA parameters and several measures in populations. It has been shown in previous researches that adapting such GA parameters automatically can improve the quality of solution (Syarif & Gen, 2003).

In this paper, we use the FLC concept for automatically tuning the $p_C$ and $p_M$ based on the changes in the average fitness of the population (Wang et al., 1997). Let $\Delta f(t)$ be the different of average fitness function between the $t$th and $t-1$st generation, the crossover and mutation ratios for the next generation in general is done using the following IF–THEN concept:

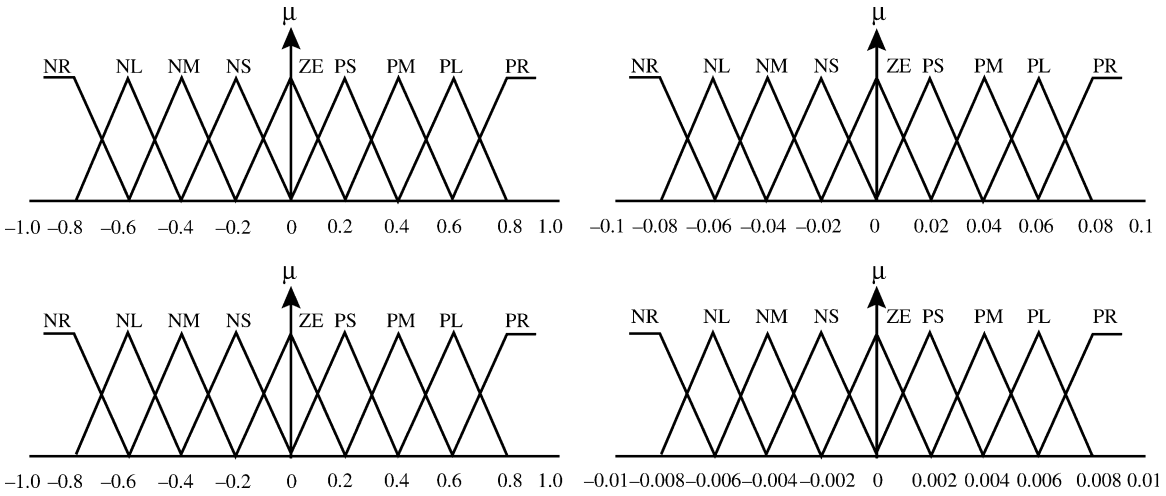


Fig. 3. The membership function for input–output FLC variables. NR, negative larger; NL, negative large; NM, negative medium; NS, negative small; ZE, zero; PS, positive small; PM, positive medium; PL, positive large; PR, positive larger.

- If $|\Delta f(t)-\Delta f(t-1)|<\varepsilon$, ($\varepsilon$ is a small positive number near to zero), then rapidly increase the $p_C$ and $p_M$ for the next generation.
- If $\Delta f(t)-\Delta f(t-1)<\varepsilon$ then decrease the $p_C$ and $p_M$ for the next generation.
- If $\Delta f(t)-\Delta f(t-1)>\varepsilon$ then $p_C$ and $p_M$ for the next generation.

In this approach, the inputs of FLC are $\Delta f(t)$ and $\Delta f(t-1)$. The outputs of the FLC are $\Delta c(t)$ and $\Delta m(t)$ (the change of crossover ratio and mutation ratio, respectively). The membership functions of fuzzy all input and output linguistic variables are illustrated in Fig. 3.

Based on a number of experimental data and domain expert opinion, the input values are respectively normalized into integer values in the range $[-4.0, 4.0]$ according to their corresponding maximum/minimum values. For simplicity, we used the fuzzy decision table as given in Table 1.

Table 1
Control action for crossover and mutation ratio

| $Z(i,j)$ | | $i$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | −4 | −3 | −2 | −1 | 0 | 1 | 2 | 3 | 4 |
| $j$ | −4 | −4 | −3 | −3 | −2 | −2 | −1 | −1 | 0 | 0 |
| | −3 | −3 | −3 | −2 | −2 | −1 | −1 | 0 | 0 | 1 |
| | −2 | −3 | −2 | −2 | −1 | −1 | 0 | 0 | 1 | 1 |
| | −1 | −2 | −2 | −1 | −1 | 0 | 0 | 1 | 1 | 2 |
| | 0 | −2 | −1 | −1 | 0 | 2 | 1 | 1 | 2 | 2 |
| | 1 | −1 | −1 | 0 | 0 | 1 | 1 | 2 | 2 | 3 |
| | 2 | −1 | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 |
| | 3 | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 |
| | 4 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 |

After assigning the input values to the corresponding indexes $i$ and $j$, we can determine the scaling value $Z(i,j)$. Then, the changes on crossover and mutation ratios are determined as follows:

$$\Delta c(t) = \alpha Z(i,j) \tag{7}$$

$$\Delta m(t) = \beta Z(i,j) \tag{8}$$

where $\alpha$ and $\beta$ are given values to regulate an increasing and decreasing range for the crossover ration and mutation ratio (e.g. $\alpha = 0.02$ and $\beta = 0.002$). The values of crossover ratio and mutation ratio for the next generation are calculated as follows

$$p_C(t+1) = p_C(t) + \Delta c(t) \tag{9}$$

$$p_M(t+1) = p_M(t) + \Delta m(t) \tag{10}$$

where $p_C(t)$ and $p_M(t)$ are crossover ratio and mutation ratio at generation $t$, respectively.

### 3.3. Overall procedure

Let $P(t)$ be a population of chromosomes for iteration $t$, $C(t)$ be the generated offspring at iteration $t$. The overall procedure of hst-GA is summarized as follows:

Procedure: hst-GA

**begin**
$t = 0$;
initialization $P(0)$;
evaluate the chromosome;
**while** (**not** termination condition) **do**
Crossover;
Mutation;
evaluate the offspring $C(t)$;
determine $P(t)$ selection;
determine the GA parameters using FLC;
$t = t + 1$;
**end**
**end**

## 4. Numerical experiments

In order to see the effectiveness of the proposed method, the hst-GA is implemented in visual C language and run on PC Pentium 700. We also run the numerical experiments using traditional st-GA on the same computer. We examine a set of test problems with different size as given in Table 2. As the initial value for the GA parameters, we set $p_C = 0.4$ and $p_M = 0.2$.

For each test problem, we have two computational experiments by giving different population size *pop_size* and maximum generation *max_gen*. Each computational experiment is run ten times. Table 3

Table 2
Design of test problems

| Problem | *T* | *I* | *J* | *K* | *M* |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 3 | 2 | 7 |
| 2 | 4 | 3 | 4 | 3 | 10 |
| 3 | 6 | 2 | 7 | 4 | 21 |

Table 3
Computational results

| Problem | *pop_size* | *max_gen* | st-GA[a] | | | hst-GA | | |
|---|---|---|---|---|---|---|---|---|
| | | | Best | Mean | Worst | Best | Mean | Worst |
| 1 | 10 | 1000 | 3372 | 3394.2 | 3412 | 3371 | 3372.2 | 3395 |
| | 20 | 1500 | 3371 | 3383.5 | 3407 | 3371 | 3374.7 | 3387 |
| 2 | 20 | 3000 | 16,732 | 16,823.7 | 17,037 | 16,687 | 16,783.2 | 17,012 |
| | 25 | 4000 | 16,692 | 16,713.2 | 16,773 | 16,687 | 16,703.1 | 16,721 |
| 3 | 50 | 4000 | 41,375 | 41,626.3 | 42,171 | 42,262 | 41,373.7 | 41,983 |
| | 75 | 5000 | 41,368 | 41,481.6 | 41,873 | 41,253 | 41,351.2 | 41,537 |

[a] Proposed method without FLC.

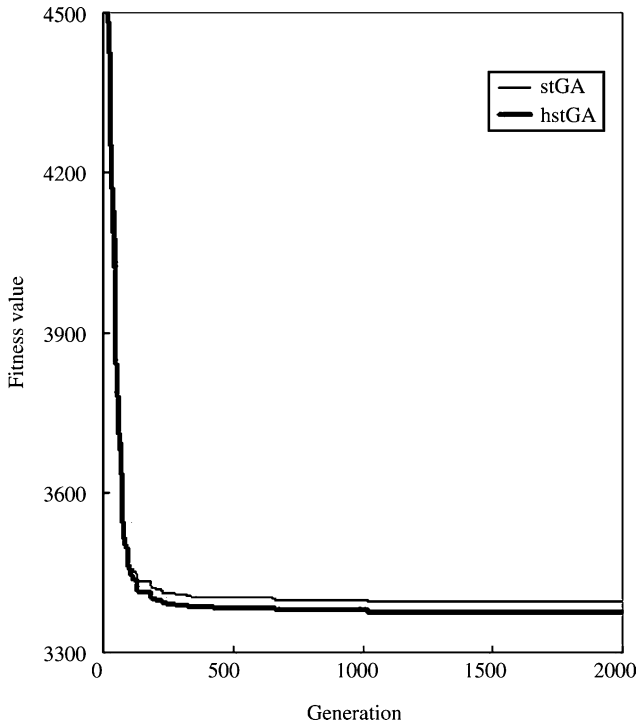


Fig. 4. The average fitness value in the generation.

shows a comparison of the computational results. These results show that the proposed method outperforms the traditional st-GA. The convergence behavior of the methods in the generations for test problem 1 is shown in Fig. 4.

## 5. Conclusion

In this paper, we proposed a new approach called spanning tree-based hybrid genetic algorithm hst-GA to solve the multi-time period production/distribution and inventory problem (mt-PDI). In order to improve the efficiency of GA, FLC was hybridized to the evolutionary process for making auto-tuning of the GA parameters. We carried out the numerical experiments and compared the results with those of the traditional spanning tree-based genetic algorithm. We observe that the hst-GA produces better results than the st-GA.

## Acknowledgements

This research was supported by the Grant-in-Aid for Scientific Research (No. 10044173) by the Ministry of Education, Science and Culture, the Japanese Government. The work of the second author was also supported by Development Undergraduate Education Project, The University of Lampung, Indonesia (Indonesian Government and World Bank Project).

## References

Davis, L. (1991). *Handbook of genetic algorithms*. Van Nostrand Reinhold.

Dossey, J., Otto, A., Spence, L., & Eynden, C. (1993). *Discrete mathematics*. Harper Collins.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: Wiley.

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*. New York: Wiley.

Gen, M., Cheng, R., & Oren, S. (2001). Network design techniques using adapted genetic algorithms. *Advances in Engineering Software*, *32*(9), 731–744.

Graves, S. C. (1999). Manufacturing, planning and control. *MIT working paper*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Lee, M., & Takagi, H. (1993). Dynamic control of genetic algorithm using fuzzy logic techniques. *Proceedings of the fifth international conference on genetic algorithm* (pp. 76–83).

Lo, C. C., & Chang, W. H. (2000). A multi-objective hybrid genetic algorithm for the capacitated multipoint network design problem. *IEEE Transaction on System, Man and Cybernetics*, *30*(3), 461–470.

Pirkul, H., & Jayaraman, V. (1998). Multi-commodity, multi-plant, capacitated location allocation problem: formulation and efficient heuristic solution. *Computers and Operations Research*, *25*(10), 869–878.

Prüfer, H. (1918). Neuer bewis eines sizes uber permutationen. *Archive für Mathematik und Physik*, *27*, 742–744.

Syarif, A., & Gen, M. (2003). Solving exclusionary side constrained transportation problem by using a hybrid spanning tree-based genetic algorithm. *Journal of Intelligent Manufacturing*, *14*(3/4), 389–399.

Wang, P. T., Wang, G. S., & Hu, Z. G. (1997). Speeding up the search process of genetic algorithm by fuzzy logic. *Proceedings of the fifth European congress on intelligent techniques and soft computing* (pp. 665–671).

Zeng, X., & Rabenasolo, B. (1997). A fuzzy logic based design for adaptive genetic algorithms. *Proceedings of the fifth European congress on intelligent techniques and soft computing* (pp. 660–664).