A Production Planning Problem Solved by the Particle Swarm Optimization

Yin-Yann Chen

Abstract—**Several researchers have referred to the capacitated production lot-sizing allocation problems as NP-Hard. Consequently, it is more difficult to solve the capacitated production allocation problem considering practical characteristics, such as allocation problems among bottleneck machines, photo masks, and products with different re-entrant layers. In this paper, we proposed a novel variation of the particle swarm optimization (PSO) model, which is a binary PSO model with adaptable inertia weight and mutation mechanism. It is converted to be able to solve the model of binary decision variables. Moreover, it improves some weaknesses, including a propensity for obstruction near the optimal solution regions that hardly improve solution quality by fine tuning. In order to compare effectiveness, the traditional PSO, genetic algorithm, and the proposed PSO in this study are compared by the practical production planning problem in the TFT Array process. Based on the results of the experiments, it can be concluded that the proposed PSO is more effective than the other approaches in terms of superiority of solution and required CPU time.**

Index Terms—**TFT Array; production planning; allocation; particle swarm.**

I. Introduction

The capacitated lot sizing problem (CLSP) is to planning the lot sizes of multiple items over a planning horizon with the objective to minimize setup and inventory holding costs [1]. Berretta & Rodrigues [2] presented heuristic methods based on evolutionary algorithms in order to address the complex multi-stage CLSP (MSCLSP), including setup costs and setup times. In the work of Ozdamar & Bozyel [1], the CLSP is extended to include capacity consuming setups and overtime decisions. The objective function consists of minimizing inventory holding and overtime costs. The different approaches including the hierarchical production planning (HPP) approach, the iterative relaxation approach, the genetic algorithm (GA), and a simulated annealing (SA) approach, are proposed to compare among them. Song & Chan [3] proposed a dynamic programming algorithm to solve a single-stage CLSP (SSCLSP) with backlogging. The objective is to minimize the total cost of setup, stockholding, and backlogging to satisfy demands. Al-Fawzan [4] considered the problem of determining lot size and production sequence when production rate, setup cost, and unit processing cost are sequence-dependent. Using a standard CLSP model with backorder, a tabu search algorithm is proposed.

The capacitated lot sizing and loading problem (CLSLP) deals with the issue of determining the lot sizes of product families/end items, and loading them on parallel facilities to satisfy dynamic demand over a given planning horizon. Ozdamar & Birbil [5] dealt with the CLSLP that is a synthesis of three problems, namely, the CLSP with overtime decisions and setup times, minimizing total tardiness on unrelated parallel processors, and the class scheduling problem, each of which is NP in the feasibility and optimality problems. In addition, hybrid heuristics involving SA, tabu search (TS), and GA are developed to solve the CLSLP. In relation to this, Sambasivan & Yahya [6] developed a Lagrangian-based approach to solve a multi-plant, multi-item, multi-period CLSLP. A real problem in a company manufacturing steel rolled products is provided.

The capacitated plant location problem (CPLP) is for finding the subset of plants that will minimize the total fixed and transportation costs such that the demand of all customers can be satisfied [7]. In the CPLP, there are a set of potential locations for plants with fixed costs and capacities, and a set of customers, with demands for goods supplied from these plants. First, a choice is made of the subset of the plants to be opened, and second, the assignment of the customers to these plants is made. When each customer must be served only from a single plant, the problem is called CPLP with single source constraints (CPLPSS). When the capacities are unrestricted, the problem is known as the simple or uncapacitated plant location problem (SPLP).

Capacitated production lot-sizing allocation problems pose challenges due to its combinational nature [6]. When capacity constraints and setup costs are considered, this problem is NP-Hard. Bitran & Yanasse [8] showed that several cases of a single item can be solved with a polynomial-time algorithm, and that the problem turns to be NP-Hard when a second item with an independent setup is introduced. When we consider non-zero setup times, the feasibility decision problem becomes NP-Complete.

Due to the computational complexity of solving the capacitated lot-sizing allocation problem in an exact way, researchers have chosen to use heuristics. These heuristic methods on production planning issues include the GA [1][9][10][11], TS [4][12], SA [1], and ant colony optimization (ACO).

The PSO approach is an evolutionary computation technique developed by Kennedy & Eberhart [13]. It is a stochastic global optimization method which is based on the simulation of social behavior. Similar to the GA, the PSO

Manuscript received December 25, 2009. This work was supported in part by the National Science Council, Taiwan, R.O.C., under contract No. NSC 97-2221-E-150-080-MY2.

Yin-Yann Chen is with the Industrial Management Department, National Formosa University, Yunlin 632, Taiwan, R.O.C. (phone: +886-5-6315716; fax: +886-5-6311548; e-mail: yyc@nfu.edu.tw).

approach exploits a population of potential solutions to explore the search space.

Compared with the GA method, there are no operators involved in the PSO; instead, natural evolution is applied to extract a new generation of candidate solutions. In contrast with the mutation mechanism, PSO rests on the exchange of information between individuals (named particles), and of the population (named swarm). Each particle adjusts its flying trajectory according to its own previous best position and the best previous position obtained by all members of its neighborhood. Furthermore in PSO, the whole swarm is considered as the neighborhood. Thus, there occurs global sharing of information and particles profit from the discoveries and previous experience of all other members during the searching process.

Initially, assuming that the search space is D-dimensional, the i-th particle of the swarm is represented by a D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The best particle of the swarm, that is, the particle with the best objective function value, is denoted by gBest. The best previous position of the i-th particle in its own searching trajectory is recorded and represented as pBest.

The velocities and positions of the particles are manipulated according to the following equations (the superscript k denotes the iteration):

$$\begin{aligned} V_i^{k+1} = {} & w \times V_i^k \\ & + c_1 \times r_{i1}^k \times \left(pBest_i^k - X_i^k\right) \\ & + c_2 \times r_{i2}^k \times \left(gBest^k - X_i^k\right) \end{aligned} \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

where i = 1, 2, …, N, and N is the size of the population; w is the inertia weight which was developed to better control exploration and exploitation; c_1 and c_2 are two positive constants, called the cognitive and social parameters respectively; and r_{i1} and r_{i2} are random numbers uniformly distributed within the range [0, 1]. Eq.(1) is used to determine the i-th particle's new velocity, at each iteration, while Eq.(2) provides the new position of the i-th particle, adding its new velocity to its current position. The performance of each particle is measured according to a given fitness function, which is problem dependent. In the optimization problems in general, the fitness function is the objective function under consideration.

The role of w is considered important for the PSO's convergence behavior. It is employed to control the impact of the previous history of velocities on the current velocity. Therefore, the parameter w regulates the trade-off between the global (wide-ranging) and the local (nearby) exploration abilities of the whole swarm. Usually, a larger inertia weight facilitates exploration for searching new regions; while a small one tends to facilitate exploitation; that is, it fine-tunes the current search space. An appropriate value for the inertia weight w thereby provides the balance between the global and local exploration ability of the swarm, resulting in better quality of solutions. Experimental results show that it is preferable to initially set the inertia to a large value in order to induce global exploration of the search space, and then gradually decrease it to obtain refined solutions.

Therefore, it is even more difficult to solve the capacitated production allocation problem considering practical characteristics. In this paper, we involve the production planning problem in the TFT (thin film transistor) Array process. First, the TFT Array process is a re-entrant flow in which a similar sequence of processing step is repeated for five times. Further, the photolithography stage, the bottleneck in the TFT Array process, requires a second resource in addition to machines. Photo masks and scanners are both indispensable while executing exposure operation in the photolithography stage. Every product with a different re-entrant layer requires a unique mask, and consequently, each has a set of mask with five different pieces. Moreover, there is a tool eligibility issue between the masks and scanners. Masks are approved to be used in specific scanners for quality considerations. The TFT Array process is characterized as reentry and mask constraints, so it is regarded as a complicated scheduling problem and is more difficult to solve than the classical ones.

II. A Production Planning Problem

We formulate the programming model for the production planning problem in this paper. The symbols are defined as follows.

Indexes:

t = period index (day), t=1,2,…,T.

p = product index, p=1,2,..,P.

s = re-entrant layer index, s=1,2,…,S.

c = bottleneck machine index, c=1,2,…,C.

Parameters:

f_t = the fixed charge incurred whenever a product is produced in period t.

sc_p = the setup cost for product p.

h_{pt} = unit cost of inventory for product p in period t.

yd = the yield rate in the TFT Array process.

$unit_p$= the converted production unit for product p, transferring from a "*lot*" to large glass substrates (*sheets*). That is, the release production unit in the TFT Array process is a "cassette (or *lot*)" including about 20 glass substrates. Then, unit_p equals 20 for a certain product p.

d_{pt} = the processing quantity demanded for product p in period t.

v_{pt} = the capacity consumed for making a unit of product p in period t.

st_{pt} = the setup time for product p in period t.

b_t = available capacity for production in period t.

mu_c = the utilization rate for bottleneck machine c.

PT_{ps} = the processing time for the s-th re-entrant layer of product p.

e_{psc} = the matching constraints for re-entrant layers, products, and machines. If the s-th re-entrant layer of product p can be processed in machine c, e_{psc}=1; and otherwise, e_{psc}=0.

TD_{ps} = the processing quantity demanded for the s-th re-entrant layer of product p.

α = the penalty cost due to setup times, which is equivalent to the total machines allocated.

β= the penalty cost due to the discrepancy between the processing quantity demanded and the allocated production amounts.

Decision variables:

X_{pt} = production amounts of product p in period t.

I_{pt} = amounts of end of period inventory for product p in period t.

Y_{pt} = binary variable, Y_{pt}=1, if product p is produced in period t ; Y_{pt}=0, otherwise.

S_{pt} = binary variable, S_{pt}=1, if a setup is performed for product p in period t ; S_{pt}=0, otherwise.

A_{psc} = the total production amounts in machine c for the s-th re-entrant layer of product p.

Z_{psc} = binary variables, Z_{psc}=1, if the s-th re-entrant layer of product p is processed in machine c ; Z_{psc}=0, otherwise.

L_c = capacity loading in machine c. Here

$$L_c = \sum_p \sum_s \left(Z_{psc} \times A_{psc} \times PT_{ps} \right) \quad ,\forall c \cdot$$

AQ_{ps} = the total allocated production amounts for the s-th re-entrant layer of product p. That is,

$$AQ_{ps} = \sum_{c=1}^{C} A_{psc} \quad ,\forall p,s \cdot$$

VQ_{ps} = the discrepancy between the processing quantity demanded and the allocated production amounts for the s-th re-entrant layer of product p. Here,

$$VQ_{ps} = TD_{ps} - AQ_{ps} \quad ,\forall p,s \,.$$

After the declarations of indexes, parameters, and decision variables, the programming constraints for production planning in the TFT Array process are described as follows:

In constraint (3) is shown the balance equations for the inventory of products.

$$I_{pt} = I_{p,t-1} + X_{pt} \times yd \times unit_p - d_{pt} \tag{3}$$

Constraint (4) is the available capacity constraint for the production in every period t. Both setup time and process time consume the capacity.

$$\sum_p \left(v_{pt} \times X_{pt} + st_{pt} \times S_{pt} \right) \le b_t \qquad \forall t \tag{4}$$

In constraint (5) is shown whether the plant makes product p in period t. The symbol, M, is defined as a big enough number larger than the maximum quantities of releasing the production in period t for product p.

$$X_{pt} \le M \times Y_{pt} \qquad \forall p,t \,, \tag{5}$$

and M is a big enough number.

In constraint (6) is shown whether the plant has a setup for product p in period t.

$$S_{pt} \ge Y_{pt} - Y_{p,t-1} \qquad \forall p,t \tag{6}$$

In constraint (7) the matching constraints for re-entrant layers, products, and machines are shown.

$$Z_{psc} \le e_{psc} \qquad \forall p,s,c \tag{7}$$

In constraint (8) is shown whether the re-entrant layer s for product p is processed in the bottleneck machine c.

$$A_{psc} \le M \times Z_{psc} \qquad \forall p,s,c \,, \tag{8}$$

and M is a big enough number.

Constraint (9) is the capacity constraints for machine c. (assume: 24 working hours per day)

$$L_c \le 24 \times SP \times mu_c \qquad \forall c \,, \tag{9}$$

where $L_c = \sum_p \sum_s \left(A_{psc} \times PT_{ps} \right)$

In constraint (10) is shown the total allocated production amounts for the s-th re-entrant layer of product p, which equals the summation of the production amounts for the s-th re-entrant layer of product p in every machine c.

$$AQ_{ps} = \sum_{c=1}^{C} A_{psc} \qquad \forall p,s \tag{10}$$

In constraint (11) is shown the discrepancy between the processing quantity demanded and the allocated production amounts for the s-th re-entrant layer of product p. Here, SP is defined as the planning horizon for pre-allocating the matching problems of the re-entrant layers, products, and machines. Moreover, we assume that every re-entrant layer of product p has the same processing quantity demanded.

$$VQ_{ps} = TD_{ps} - AQ_{ps} \qquad \forall p,s \,, \tag{11}$$

where $TD_{ps} = \sum_{t \in SP} X_{pt}$

Constraint (12)-(16) are the basic restrictions on the decision variables.

$$X_{pt}, I_{pt} \ge 0 \qquad \forall p,t \tag{12}$$

$$A_{psc} \ge 0 \qquad \forall p,s,c \tag{13}$$

$$AQ_{ps}, VQ_{ps} \ge 0 \qquad \forall p,s \tag{14}$$

$$Z_{psc} \in \{0,1\} \qquad \forall p,s,c \tag{15}$$

$$Y_{pt}, S_{pt} \in \{0,1\} \qquad \forall p,t \tag{16}$$

The objective function [Eq.(17)] is for minimizing the total costs including inventory holding costs, setup costs, fixed charge production costs, the penalty cost due to changeover times among machines for the re-entrant layers of products, and the penalty cost due to the discrepancy between the processing quantity demanded and the allocated production amounts.

$$\begin{aligned} Min \quad & \sum_p \sum_{t'} \left(h_{pt'} R_{pt'} + sc_p S_{pt'} + f_{t'} Y_{pt'} \right) \\ & + \alpha \times \sum_p \sum_s \sum_c Z_{psc} \\ & + \beta \times \sum_p \sum_s VQ_{ps} \end{aligned} \tag{17}$$

III. A VARIANT OF PSO

The PSO algorithm has been successfully applied to many kinds of optimization problems. However, though the approach has shown some important advantages by providing high-speed convergence in specific problems, studies are shown that the algorithm has a propensity for obstruction near the optimal solution regions and find it difficult to improve solution quality by fine tuning. In addition, the original updating process of positions and velocities must be modified when engaging in the binary decision variables. Consequently, this paper proposes a new variation of the PSO model. It is a binary PSO with the adaptable inertia weight and mutation mechanism. Meanwhile, we have kept the inherent property of the PSO, that is, the advantage of fast convergence.

With the introduction of the concept of inertia weight, the aim is to balance and adjust the global search and local search. Furthermore, better performance would be obtained if the inertia weight were chosen a time varying, linearly decreasing quantity, rather than being a constant value. Consequently, a higher inertia weight implies larger incremental changes in velocity per iteration, and thus the exploration of new search areas for better solution. However, a smaller inertia weight signifies less variation in velocity, providing slower change in terms of fine tuning a local search. Therefore, it would be better that the searching process should start with a high inertia weight for global exploration, with the inertia weight linearly decreasing to facilitate finer local explorations in later iterations.

A novel *nonlinear* function which regulates the inertia weight with variation in time is proposed in this study. This PSO version with adaptable inertia weight, as well as the mutation mechanism, improves the efficiency of performance once applied to problems with binary variables. The proposed inertia weight w is given as follows:

$$w = w_{\max} - \left(w_{\max} - w_{\min}\right) \times \left(\frac{iter}{iter_{\max}}\right)^{\alpha} \tag{18}$$

where w_{max} is set as the maximum of inertia weight; w_{min} is set as the minimum of inertia weight; $iter$ is the iteration number at the current time step; $iter_{max}$ is the maximum number of iterations in a given run; and α is the nonlinear adaptable coefficient during the iterations.

With α=1, the changes become a special case of linearly regulative inertia weight with time, as proposed by Shi and Eberhart [14]. The searching process starts with a larger inertia weight ($w_{\max}$), which facilitates aggressive exploration of new solution areas; this then gradually decreases according to Eq.(18). It will result in the different decreasing path for different values of α to reach $w_{\min}$ at the final iteration ($iter_{\max}$). After repetitive experiments, our model with α=1.5 has a higher value of w during the early iterations, which is more aggressive than Shi and Eberharts' linear model (α=1). Also, in our model during the later iterations, w decreased more rapidly than the linear case, which is beneficial in accelerating the speed of convergence. However, if α is very large, then it may jump over the optimal areas during early iterations due to too large searching steps, and diverge or oscillate excessively during later iterations due to the rapid decrease in the value of w.

Concerning the release production planning model in the TFT Array process as illustrated in Section 2, it is mainly to determine both binary decision variables, Y_{pt} (whether product p is produced in period t.) and Z_{psc} (whether the s-th re-entrant layer of product p is processed in machine c). Therefore, this paper solves the problem by way of the binary PSO with the adaptable inertia weight and mutation mechanism, as well as techniques of constraints handling. Its steps are stated as follows:

(1). Initialization

The binary version of the PSO algorithm is employed in this study, so an initial population of particles is randomly constructed in that each particle's position is either 0 or 1. Next, we set the maximum and minimum velocities of particles, which is limited to the boundary, $V=[V_{\min}, V_{\max}]=[-4,4]$, and the initial velocity is generated by the following:

$$V = V_{\min} + rand^{*}(V_{\max} - V_{\min}), \tag{19}$$

where *rand* means the random number.

The fitness value is evaluated by both the objective and penalty functions due to violation of the constraints of the programming model.

(2). Updating position and velocity

During the repetitive iterations, the velocity of each particle is updated by Eq.(20), where c_1 and c_2 are social and cognitive parameters; $rand_1$ and $rand_2$ are random numbers between (0,1); *pBest* is the current best position of each particle in its own searching trajectory and *gBest* is the best value of the whole swarm.

$$\begin{aligned} V_{new} &= w \times V_{old} \\ &\quad + c_1 \times rand_1 \times (pBest - X_{old}) \\ &\quad + c_2 \times rand_2 \times (gBest - X_{old}) \end{aligned} \tag{20}$$

Here, the inertia weight is nonlinearly regulated during iterations according to Eq.(21).

$$w = w_{\max} - \left(w_{\max} - w_{\min}\right) \times \left(\frac{iter}{iter_{\max}}\right)^{\alpha} \tag{21}$$

If the particle's velocity exceeds the maximum, $V_{\max}$, it has to be replaced by $V_{\max}$. Similarly, if the particle's velocity is smaller than the minimum, it also has to be replaced by $V_{\min}$. Additionally, the sigmoid function [Eq.(22)] is used to scale the velocities between 0 and 1 because the sigmoid function value advances to 1 when ε approaches the positive infinity and to 0 when ε approaches the negative infinity.

$$Sigmoid(\varepsilon) = \frac{1}{1 + e^{-\varepsilon}} \tag{22}$$

Finally, each particle's position is updated according to Eq.(23).

$$X = \begin{cases} 1, & if \quad U(0,1) < Sigmoid(V) \\ 0, & otherwise \end{cases} \tag{23}$$

(3). Updating particle best (pBset)

The *pBest* is the best position of each particle in its own searching process. During the iterations, the particle's fitness evaluation is compared with *pBest*. If the current value is better than *pBest*, then set *pBest* value equal to the current value.

(4). Updating global best (gBest)

Compare fitness evaluation with the population's overall previous best, *gBest*. If the current value is better than *gBest*, then update the current particle's value to *gBest*.

(5). Mutation mechanism

With the later iterations, the mutation mechanism is involved in the proposed PSO approach in order to avoid falling into the local optimal area. In this study, the mutation rate is set to 0.03.

(6). Stopping criterion

If the number of iteration exceeds the maximum number of iterations, then stop.

The GA, which is similar to the PSO, is also an evolutionary population-based search method that provides optimal or near-optimal solutions for combinatorial optimization problems. In the literature, it has been successfully applied to a number of research fields. The main factors in developing a GA are chromosome representation, population initialization, evaluation measure, crossover, mutation, and selection strategy. The comparison between PSO and GA had been discussed in a previous investigation [15]. In our study, relative comparisons are illustrated in the following section 4.

IV. An Illustrated Example

We employed the proposed PSO approach to solve this problem. The unit cost of inventory for 19-inch XG01 and 19-inch XG02 products for every period are \$0.003 and \$0.004, respectively. The setup costs for these two products are \$0.02 and \$0.03, respectively. The fixed charge for the array plant is \$1 every time. The capacity constraint of the array plant is 30 lots per day.

Generally speaking, every product in the TFT Array process needs re-entrant manufacturing processes five times, including the gate layer, an a-silicon layer, the source/drain layer, the back channel passivation layer, and the indium tin oxide layer. Photo masks and scanners are both indispensable in executing exposure operation during the photolithography stage. Nevertheless, a problem surfaced with the amount restriction in that even with adequate scanners, with insufficient or appropriate photo masks, the process cannot work. This is the reason why scanners and photo masks closely depend on each other.

Table.1 The available processing machines (M) for products with different re-entrant layers.

Product:	19 in. XG01	19 in. XG02
Layer 1: Gate	M1, M2	M1
Layer 2: a-Silicon	M1	M2, M3
Layer 3: Source Drain	M1, M3	M2
Layer 4: Back Channel Passivation	M3	M2, M3
Layer 5: Indium Tin Oxide	M2, M3	M1

Table.1 shows the available processing machines for products with different re-entrant layers. For example, the first layer (gate layer) of the 19-inch XG01 product can be processed on the bottleneck machines M1 and M2. In contrast, the second layer (a-Silicon layer) of the 19-inch XG01 product can only be manufactured on machine M1. Meanwhile, the process time for making one unit of the product in different re-entrant layers is tabulated in Table.2.

Table.2 The process time for making one unit of product in different re-entrant layers (unit: minute)

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Process time	0.50	0.65	0.85	0.9	0.55

The minimum total cost solved by the branch and bound approach is 29.144; TABLE.3 displays the allocation results.

Table.3 The allocation result: machines (M) vs. products for different re-entrant layers

Product:	19 in. XG01	19 in. XG02
Layer 1: Gate	M1	M1
Layer 2: a-Silicon	M1	M3
Layer 3: Source Drain	M3	M2
Layer 4: Back Channel Passivation	M3	M2
Layer 5: Indium Tin Oxide	M2	M1

Several researches have referred to the capacitated production lot-sizing allocation problems as NP-Hard. Clearly, with its practical characteristics and constraints in the TFT Array process, the capacitated production allocation is more difficult to solve. If these kinds of problems, when solved by optimization techniques, like the branch and bound approach, cost very efforts and time to acquire the optimal results. For this reason, we employed the proposed PSO to solve it and obtained the optimal total cost of 29.566. We likewise used the GA to solve the same problem; this resulted in the optimal total cost of 30.164.

Based on the results of experiments, it is apparent that the proposed PSO is faster for convergence during the early iterations. Also done is a comparison between branch and bound (BB), the traditional PSO, GA, and the proposed PSO in terms of superiority of solution and percentage of discrepancy as opposed to the optimal value, as tabulated in TABLE.4. In addition, the CPU time required for the proposed PSO is only 36.69% of the GA's solving time. Hence, it can be concluded that the proposed PSO is more effective than the other approaches from the perspective of the best solutions and the CPU time required.

Table.4 The comparison of the three approaches

	The optimal value f^* (Branch and Bound)	The traditional PSO	GA	The proposed PSO
The best solution	29.144	30.697	30.164	29.566
Discrepancy (%)		5.06%	3.38%	1.42%

V. Conclusion

Numerous researchers have referred to the capacitated production lot-sizing allocation problems as NP-Hard. Therefore, it is more difficult to solve the capacitated production allocation problem considering some practical characteristics, such as allocation problems among bottleneck machines, photo masks, and products with different re-entrant layers. This study proposed a novel variation of the PSO model, which is a binary PSO model with adaptable inertia weight and mutation mechanism. It can be converted to be able to solve the model of binary decision variables. Moreover, it improves some weaknesses as opposed to the original version of the PSO, including a propensity for obstruction near the optimal solution regions that hardly improve solution quality by fine tuning. Comparing effectiveness, the traditional PSO, genetic algorithm, and the proposed PSO in this paper are compared by the production planning problem in the TFT Array process. According to the results, it can be concluded that the proposed PSO is more effective than the other approaches in terms of superiority of solution and required CPU time.

Acknowledgment

The author gratefully acknowledge the National Science Council, Taiwan, R.O.C., for support under contract No. NSC 97-2221-E-150-080-MY2.

References

[1] L. Ozdamar and M. A. Bozyel, "The capacitated lot sizing problem with overtime decisions and setup times," *IIE Transactions*, vol. 32, pp. 1043–1057, 2000.

[2] R. Berretta and L. F. Rodrigues, "A memetic algorithm for a multistage capacitated lot-sizing problem," *International Journal of Production Economics*, vol. 87, pp. 67–81, 2004.

[3] Y. Song and G. H. Chan, "Single item lot-sizing problems with backlogging on a single machine at a finite production rate," *European Journal of Operational Research*, vol. 161, pp. 191–202, 2005.

[4] M. A. Al-Fawzan, "An algorithm for production planning in a flexible production system," *Computers & Industrial Engineering*, vol. 48, pp. 681–691, 2005.

[5] L. Ozdamar and S. I. Birbil, "Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions," *European Journal of Operational Research*, vol. 110, pp. 525–547, 1998.

[6] M. Sambasivan and S. Yahya, "A Lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers," *Computers & Operations Research*, vol. 32, pp. 537–555, 2005.

[7] R. Sridharan, "The capacitated plant location problem," *European Journal of Operational Research*, vol. 87, pp. 203–213, 1995.

[8] G. R. Bitran and H. H. Yanasse, "Computational complexity of the capacitated lot size problem," *Management Science*, vol. 28(10), pp. 74–86, 1982.

[9] Y. J. Jang, S. Y. Jang, B. M. Chang, and J. Park, "A combined model of network design and production/ distribution planning for a supply network," *Computers & Industrial Engineering*, vol. 43, pp. 263–281, 2002.

[10] Y. H. Lee, S. H. Kim, and C. Moon, "Production-distribution planning in supply chain using a hybrid approach," *Production Planning & Control*, vol.13, pp. 35–46, 2002.

[11] C. Moon, J. Kim, and S. Hur, "Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain," *Computers & Industrial Engineering*, vol. 43, pp. 331–349, 2002.

[12] W. C. Chiang and R. A. Russell, "Integrating purchasing and routing in a propane gas supply chain," *European Journal of Operational Research*, vol. 154, pp. 710–729, 2004.

[13] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," In: *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, 1995, pp. 1942–1948.

[14] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," In: *IEEE International Conference on Evolutionary Programming*, Alaska1, 1998, pp. 69–73.

[15] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," In: *Annual Conference on Evolutionary Programming*, San Diego, 1998, pp. 611–616.