ICMIEE-PI-140308

Application of Particle Swarm Optimization in Aggregate Production Planning and Comparison with Genetic Algorithm

Md. Rakibul Islam[1], Md. Shahriar Aziz[2], Md. Mahfuzul Haque Muftee[3], Md. Sanowar Hossain[4]*
[1, 2, 4]Department of Industrial & Production Engineering,
Rajshahi University of Engineering & Technology, Rajshahi- 6240, BANGLADESH
[3]Department of Computer Science & Engineering,
Rajshahi University of Engineering & Technology, Rajshahi- 6240, BANGLADESH

ABSTRACT
This paper is aimed towards application of particle swarm optimization for constrained optimization as aggregate production planning and comparison of the result with genetic algorithm under uncertain demand in a predefined range. As in aggregate production planning 80% decision depends on cost only, we have eliminated other objective functions of APP in this case. We have used some data from CCKL, one the leading company in RMG sector in Bangladesh. There was a 24 variables problem with 24 constraints which was solved in MATLAB and the results between the aforementioned algorithms were compared. It is found that under these constraints PSO produces better result. In contrast with GA, PSO requires far less parameters to adjust and minimal time. Due to these reasons the authors suggest using PSO in such multi constraint problem.

Key words: Particle swarm optimization (PSO); Genetic Algorithm (GA); Aggregate Production Planning (APP)

1. Introduction
Aggregate production planning is concerned with the determination of production, inventory, and work force levels to meet fluctuating demand requirements over a planning horizon that ranges from six months to one year. Typically the planning horizon incorporates the next seasonal peak in demand. The planning horizon is often divided into periods. For example, a one year planning horizon may be composed of six one-month periods plus two or three month periods. Normally, the physical resources of the firm are assumed to be fixed during the planning horizon of interest and the planning effort is oriented toward the best utilization of those resources, given the external demand requirements. A firm must plan its manufacturing activities at a variety of levels and operate these as a system. Planners must make decisions on output rates, employment levels and changes, inventory levels and changes, back orders, and subcontracting. Aggregate planning determines not only the output levels planned but also the appropriate resource input mix to be used.
Aggregate planning might seek to influence demand as well as supply. If this is the case, variables such as price, advertising, and product mix might be used. If changes in demand are considered, then marketing, along with operations, will be intimately involved in aggregate planning. Aggregate planning is essentially a big-picture approach to planning.
There are many solving procedure for APP problem but in this paper we have used two new algorithms PSO and GA. PSO has been used by many applications of several problems. The algorithm of PSO emulates from behavior of animals societies that don't have any leader in their group or swarm, such as bird flocking and fish schooling. Typically, a flock of animals that have no leaders will find food by random, follow one of the members of the group that has the closest position with a food source (potential solution). The flocks achieve their best condition simultaneously through communication among members who already have a better situation. Animal which has a better condition will inform it to its flocks and the others will move simultaneously to that place. This would happen repeatedly until the best conditions or a food source discovered. The process of PSO algorithm in finding optimal values follows the work of this animal society. Particle swarm optimization consists of a swarm of particles, where particle represent a potential solution.
Most APP models can be formulated as linear programming problems but this is not the case for the proposed model. This warrants an opportunity of application for near-optimal heuristics to solve this multiple objectives optimization problem. In this study, PSO is a relatively new approach for solving optimization problem is employed to solve the proposed APP problem due to its simplicity, speed, and robustness. The PSO allows a group of particles to search for the solution. Knowledge gained by each agent is shared among one another in order to iteratively find an improved solution.

2. Literature Review
Aggregate production planning has lured a significant academic researchers & practitioners because of its immense importance. Shorten product life cycle in market & fickle customer perceptions push the researchers to choose this broad area to research.APP is the problem to determine the resource capacity needed to meet demand in the production line. Many researchers have studied to solve this type of management problems. Linear programming model with linear cost structure was proposed by Hanssman and Hess (1960) [1] to schedule production and employment. Multiple regressions were also used to determine proper coefficients of APP decision model (Bowman, 1963) [2] proposed a search-

* Corresponding author. Tel.: +88-01774686877
E-mail addresses: rbn_khan@yahoo.com

base simulation model. Some heuristic optimization techniques have been developed to solve APP problems. A search decision rule (SDR) was developed to generate an acceptable solution for APP (Taubert, 1968) [3]. Mellichamp and Love (1978) [4] presented the adaptable production switching heuristics (PSH) model whose results were quite consistent with the actual managerial practices.
The newer works included the use of spreadsheet software to solve APP problems in easier accessible way (Techawiboonwong and Yenradee, 2003) [5]. Das et al. (2000) [6] integrated APP, master production scheduling, and short-term production scheduling to a common data model.
Some meta-heuristic algorithms were also employed to solve APP problems. Stockton and Quinn (1995) [7] proposed a genetic algorithm based method for solving an APP problem. Wang and Fang (1997) [8] applied genetic algorithm (GA) based method with fuzzy logic to imitate the human decision procedure. Instead of locating exact optimal solution, this algorithm searched for a family of inexact solutions within acceptable level. Then, a final solution was selected by examining a convex combination of these solutions. Kumar and Haq (2005) [9] solved an APP problem by using ant colony algorithm (AGA), genetic algorithm (GA), and hybrid genetic-ant colony algorithm (HGA). From the outcomes obtained, GA and HGA showed comparably good performance. Constrained optimizations are often being solved by different direct & indirect approaches. Among indirect approaches genetic approach is mostly lucrative because of its consistent & optimized results (Ioannis, 2009) [10].Genetic algorithm is furnished with different genetic parameters like crossover, mutation, selection functions etc and different researchers used different combinations to solve constrained & unconstrained optimization problems (Bunnag & Sun, 2005) [11].
Particle Swarm Optimization (PSO) is another relatively new bio-inspired algorithm that may be used to find optimal or near-optimal solutions to numerical and quantitative problems. It was originally developed by a social psychologist, James Kennedy and Russel Eberhart (1995) [12]. The algorithm was modeled the flocking behavior seen in many species of birds. It embeds some mechanisms that are quite robust and can avoid local optima trap. Moreover, its evaluating function does not have to be twice differentiable. These make the PSO very attractive as one of the most efficient and effective optimization algorithm. Furthermore, the PSO is very easy to implement with few lines computer code. It has been applied to solve a wide variety of applications. El Mounayri et al. (2003) [13] used PSO to predict parameters of surface roughness in end milling. Prakasvudhisarn (2004) [14] used PSO to determine minimum tolerance zones of all basic form features for discrete parts inspection. The PSO was also extended to solve discrete problems. Kennedy and Eberhart (1997) [15] modified PSO to handle discrete binary variables. Experiments were conducted on standard test functions. The obtained outcomes showed that the PSO still performed well in terms of quality of solutions, robustness, and speed. Later, PSO was applied to other discrete problems including lot sizing problem (Tasgetiren and Liang, 2003) [16], flow-shop scheduling (Lian et al. 2006) [17].
Some researchers applied PSO to solve optimization problems with constraints. Hu et al. (2003) [18] modified the PSO to solve constrained nonlinear problems by preserving only feasible solutions. In this method, the PSO checks whether the current particle violates any constraints or not. If none of constraints is violated, mechanisms of PSO will continue normally. Otherwise, a wasted iteration occurs. This will loop until a feasible solution is found.

3. Algorithm

3.1 Particle Swarm Optimization

PSO is a new technique to deal with the problems whose solutions can be represented as a point in a *D*-dimensional solution space. PSO is initialized with a population of random particles $(X_1, X_2, \dots, X_D)$ which distribute uniformlyaround search space at first. Assuming that, the position and velocity of the i^{th}particle is represented by *D*-dimensional vectors $X_i = (x_{i1}, x_{i2} \dots., x_{iD})andV_i = (v_{i1}, v_{i2}, \dots., v_{iD})$,respectively. The best previous position (*pbest*) of the i^{th}particle is defined as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the best position of the population (*gbest*) is denoted by$P_g = (p_{g1}, p_{g2}, \dots, p_{gD})$. The new velocity and position are updated according to following equations:

$$V_i^{k+1} = wV_i^k + c_1 r_1 (P_i - X_i^k) + c_2 r_2 (P_g - X_i^k) \quad (1)$$

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

Where$i = 1, 2, \dots, N$and N is the size of the population; $k = 1, 2, \dots, K$ and K is the maximum number of iterations; w is the inertia weight; c_1 and c_2 are two positive constants,usually we choose $c_1 = c_2 = 2$; r_1 and $r_2$are two randomfunctions in the range from 0 to 1. In PSO, the constraint conditions of velocity and position are:

$$-v_{max} \le v_{id} \le v_{max}, \; x_{min} \le x_{id} \le x_{max} \quad (3)$$

Where v_{max} is the maximum velocity, which allows actually serves as a constraint that controls the maximum global exploration ability PSO can have; x_{min} and x_{max} are the lower boundary and upper boundary of the solution space. The performance of each particle is measured according to a pre-defined fitness function which is problem dependent. Each particle observes the "fitness" of itself and its neighbors and emulates successful neighbors by moving towards them. This extremely simple approach has been surprisingly effective across a variety of problem domains.
In PSO, the inertia weight $w$plays a considered important role, because the balance between the global and local exploration abilities is mainly controlled by the inertia weight. Therefore, the parameter w will influence the

PSO'sconvergence behavior and choose a suitable w will help algorithm find the optimum solution accurately and rapidly. Large inertia weight at the beginning helps to find good seeds and the later small inertia weight facilitates fine search. So, a linearly decreasing inertia weight technique is developed, which linearly vary from 0.95 at the beginning of the search to 0.4 at the end. This technique has proven to be very efficient for balancing between the global and local exploration abilities. For this reason, this technique is used in our research and the inertia weight is determined by following equation:

$$w = w_{start} - \frac{w_{start} - w_{end}}{k} k \qquad (4)$$

Where w_{start} and w_{end} denote the start and end value ofinertia weight, respectively.

The procedure of standard PSO can be summarized as follows (Algorithm 1):

```
Step 1: Initialize the size of the population $N$
        Initialize the dimension of the solution space $D$
        Initialize the maximum number of iterations $K$
        Initialize the inertia weight $w_{start}$ and $w_{end}$
Step 2: For each particle
        Initialize the particle position $X_i$ randomlyInitialize the particle velocity $V_i$randomlyInitialize the current position as $P_i$
        Evaluate the fitness value
        Initialize$P_g$according to the fitness value
Step 3: Calculate new inertia weight according to (4).
Step 4: Update velocity of each particle according to (1),
        If$v_{id} > v_{max}$ , then $v_{id} = v_{max}$
        If$v_{id} < -v_{max}$ , then $v_{id} = -v_{max}$
Step 5: Update position of each particle according to (2),
        If $x_{id} > x_{max}$ , then $x_{id} = x_{max}$
        If $x_{id} < x_{min}$ , then $x_{id} = x_{min}$
Step 6: Evaluate the fitness values of all particles. For each particle, compare its current fitness value with the fitness of its pbest. If current value is better, then update pbest and its fitness value. Furthermore, determine the best particle of current population with the best fitness value. If the fitness value is better than the fitness of gbest, then update gbest and its fitness value with the position and objective value of the current best particle.
Step 7: If the maximum number of iterations or any other predefined criterion is met, then stop; otherwise go back to Step 3.
```

3.2Genetic Algorithm

Genetic Algorithms were invented to mimic some of the processes observed in natural evolution. Many people, biologists included, are astonished that life at the level of complexity that we observe could have evolved in the relatively short time suggested by the fossil record. The idea with GA is to use this power of evolution to solve optimization problems. The father of the original Genetic Algorithm was John Holland who invented it in the early 1970's. The GA proposed by Holland (1975) to encode the features of a problem by chromosomes, where each gene represents a feature of the problem. Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution; especially those follow the principles first laid down by Charles Darwin of "survival of the fittest". Since, in nature competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.

In general, GA consists of the following steps:

```
Step 1: Initialize a population of chromosomes.
Step 2: Evaluate the fitness of each chromosome.
Step 3: Create new chromosomes by applying genetic operators such as reproduction, crossover and mutation to current chromosomes.
Step 4: Evaluate the fitness of the new population of chromosomes.
Step 5: If the termination condition is satisfied, stop and return the best chromosome; otherwise, go to Step 3.
```

The ability of the algorithm to explore and exploit simultaneously, a growing amount of theoretical justification, and successful application to real-world problems strengthens the conclusion that GAs are a powerful, robust optimization technique.

4. Problem Formulation

4.1 Problem Description & Notation

Authors have used a 2 product 2 period scenario for this case. We've employed the following notations for formulating the APP problem which predominantly akin in different literatures (Wang and Fang, 2001; R. C. Wang & T. F. Liang, 2004) [19, 20].

D_{nt}	Forecast demand for nth product in period t (units)
a_{nt}	Regular time production cost per unit for nth product in period t (TK. /unit)
Q_{nt}	Regular time production for nth product in period t (units)
i_a	Escalating factor for regular time production cost (%)
b_{nt}	Overtime production cost per unit for nth product in period t (TK. /unit)
O_{nt}	Overtime production for nth product in period t (units)
i_b	Escalating factor for overtime production cost (%)
c_{nt}	Subcontracting cost per unit for nth product in period t (TK. /unit)
S_{nt}	Subcontracting volume for nth product in period t (units)

i_c	Escalating factor for subcontract cost (%)
d_{nt}	Inventory carrying cost per unit for nth product in period t (TK. /unit)
I_{nt}	Inventory level in period for nth product (units)
i_d	Escalating factor for inventory carrying cost (%)
e_{nt}	Backorder cost per unit of nth product in period t (TK. /unit)
B_{nt}	Backorder level for nth product in period t (units)
i_e	Escalating factor for Backorder cost (%)
K_t	Cost to hire one worker in period t (Tk. /man-hour)
H_t	Worker hired in period t (man-hour)
m_t	Cost to layoff one worker in period t (Tk. /man-hour)
F_t	Workers laid off in period t (man-hour)
i_f	Escalating factor for hire and layoff cost (%)
i_{nt}	Hours of labor per unit of nth product in period (man-hour/unit)
r_{nt}	Hours of machine usage per unit of nth product in period t (machine-hour/unit)
V_{nt}	Warehouse spaces per unit of nth product in period (man-hour/unit)
W_{tmax}	Maximum labor level available in period t (man-hour)
M_{tmax}	Maximum capacity available in period t (machine-hour)
V_{tmax}	Maximum warehouse space available in period t (ft^2)

4.2 Objective Function

Most practical decisions made to solve APP problems usually consider total costs. Total costs as objective function (Wang and Fang, 2001) [19]. The total costs are the sum of the production costs and the costs of changes in labor levels over the planning horizon T. Accordingly, the objective function of the proposed model is as follows:

$$\boldsymbol{MinZ} = \sum_{n=1}^{N}\sum_{n=1}^{T}\left[a_{nt}Q_{nt}(1+i_a)^t + b_{nt}O_{nt}(1+i_b)^t + c_{nt}S_{nt}(1+i_c)^t + d_{nt}I_{nt}(1+i_d)^t + e_{nt}B_{nt}(1+i_e)^t + \sum_{t=1}^{T}(K_tH_t + m_tF_t)(1+i_f)^t \right. \quad (5)$$

4.3 Constraints

Constraints on carrying inventory:

$$I_{nt} - B_{nt} = I_{n(t-1)} - B_{n(t-1)} + Q_{nt} + O_{nt} + S_{nt} - D_{nt}\, for \forall n, \forall t \quad (6)$$

Constraints on Labor levels:

$$\sum_{n=1}^{N} i_{nt}\,(Q_{nt} + O_{nt}) \le W_{tmax}\, for \forall t \quad (7)$$

$$\sum_{n=1}^{N} i_{n(t-1)}\left(Q_{n(t-1)} + O_{n(t-1)}\right) + H_t - F_t - \sum_{n=1}^{N} i_{nt}\,(Q_{nt} + O_{nt}) = 0 \quad for \forall t \quad (8)$$

Constraints on Machine capacity & Warehouse space:

$$\sum_{n=1}^{N} r_{nt}\,(Q_{nt} + O_{nt}) \le M_{tmax}\, for \forall t \quad (9)$$

$$\sum_{n=1}^{N} V_{nt}\, I_{nt} \le V_{tmax}\, for \forall t \quad (10)$$

Others:

$$I_{nt} \ge I_{ntmin}\, for \forall n, \forall t \quad (11)$$

$$B_{nt} \le B_{ntmax}\, for \forall n, \forall t \quad (12)$$

$$S_{nt} \le S_{ntmax}\, for \forall n, \forall t \quad (13)$$

No negativity constraints on decision variables are:

$$Q_{nt}, O_{nt}, S_{nt}, I_{nt}, B_{nt}, H_t, F_t \ge 0 \;\; for \forall n, \forall t \quad (14)$$

5. Case Description

The Comfit Composite Knit Limited is the sister concern of Youth Group, which is one of the pioneer company of Ready Made Garments (RMG) sector in Bangladesh. This company readily produced knit ware items among them some are fancy & some are expensive. The jacket items as well as cardigan items are most expensive and most time & cost incurring manufacturing items. So it needs a lot of precise observations & perfect manufacturing practices to catch up the market& satisfy the buyers within specified lead time. Since they are the most expensive items, major concentration was on one particular style of hooded jacket (Product 1) & another special type of ladies cardigan (Product 2).

The APP decision problem for CCKL's Knit garments manufacturing plant presented here focuses on developing an interactive Genetic Algorithm approach for minimizing total costs. The planning horizon is 2 months long, including May and June. The model includes two types of knit ware items, namely the hooded jacket (Product 1) and special type of ladies cardigan (Product 2).According to the preliminary environmental information, Tables 1 & 2 summarizes the forecast demand, related operating cost, and capacity data used in the CCKL case. Other relevant data are as follows.

I. Initial inventory in period 1 is 500 units of product 1 and 200 units of product 2. End inventory in period 2 is 400 units of product 1and 300 units of product 2.
II. Initial labor level is 225 man-hours. The costs associated with hiring and layoffs are Tk. 22 and Tk. 8 per worker per hour, respectively.
III. Hours of labor per unit for any periods are fixed to 0.033 man-hours for product 1 and 0.05 man hours for product 2. Hours of machine usage per unit for each of the two planning periods are 0.1 machine-hours for product 1 and 0.08, machine-hours for product 2. Warehouse spaces required per unit are 1 square feet for product 1 and 1.5 square feet for product 2.
IV. The expected escalating factor in each of the costs categories are 1%.

Table 1 Forecasted demand, maximum labor, machine, warehouse capacity, back order level, subcontracted volume & minimum Inventory data

Item (Unit)	Period		Item	Period	
	1	2		1	2
D_{1t}	**1350-1450**	**2950-3050**	S_{1tmax} (pieces)	200	350
D_{2t}	**1550-1650**	**750-850**	S_{2tmax} (pieces)	100	100
W_{tmax} (man-hours)	225	225	I_{1tmax} (pieces)	300	500
M_{tmax} (machine-hours)	400	500	I_{2tmax} (pieces)	150	200
V_{tmax} (ft2)	1000	1000	B_{1tmax} (pieces)	200	600
			B_{2tmax} (pieces)	150	100

Table 2 Related operating cost data for the CCKL case

Product	a_{nt} (tk./unit)	b_{nt}(tk./unit)	c_{nt}(tk./unit)	d_{nt}(tk./unit)	e_{nt}(tk./unit)
1	22	40	27	3.5	42
2	20	40	30	4	47

6. Results & Findings

After running the problem in MATLAB 2014a by a pc with the configuration AMD A6 4400m dual core, 4 gb ram, radeon hd 7520g graphics card for both genetic algorithm and PSO significant difference were noticed. In this problem PSO excelled in all criterion. It had the better objective function value in shortest time. In this case we have used c_1= c_2=1.49 and the inertia range 0.1 to 1.1. On comparison GA took ridiculously high time and inferior result. For a 3 variable 3 constraints problem GA took 10.536 seconds while for this particular problem it took as much as 177.684 second. So even with time consideration, PSO is more suitable than GA. The APP decision problem presented in the CCKL case was solved using the PSO and GA, as summarized in Table 3.Consequently, the optimal value when applying PSO to minimize the total costs was Tk. 230075.1925. In contrast with the GA approach, the results were Tk. Finally, the PSO approach is useful for solving APP decision problems and can generate better decisions than other models within very short time.

Table 3 Objective function value & required time

Algorithm	GA	PSO
Z	275931	230075.1925
Time	177.684s	2.107s

Table 4APP plan for the CCKL case

	GA		PSO	
Variables	Period 1	Period 2	Period 1	Period 2
Q_{1t}	405.999	1.98E-05	569.2108	999.7696
Q_{2t}	2.34E-05	212.6665109	629.2627	445.9673
O_{1t}	403.9996	3276.999959	568.8371	999.1826
O_{2t}	1824.667	212.6665381	628.7374	446.0327
S_{1t}	200.0002	2.28E-05	200	350
S_{2t}	1.55E-05	100.000023	99.99999	99.99999
I_{1t}	299.9997	499.9993055	428.0479	500
I_{2t}	466.6667	199.9991704	150	200
B_{1t}	2.00E+02	2.80E-05	1.00E-05	600
B_{2t}	4.84E-05	99.99998836	150	99.99999
H_t	120.2314	11.44432553	100.4556	5.01894
B_t	2.27E+00	8.03E-06	1.00E-05	5.090898

Table 5 Corresponding demand value within the range

Demand Value	Period 1	Period 2
D_{1t}	1410	2977
D_{2t}	1558	792

7. Conclusion & Future Works

The results show that PSO is much more feasible in this type of case. Besides, fewer parameters selection has made it much easier to work with PSO. For these reasons we can use PSO instead of GA in various engineering problem. In future we can work with total uncertain condition. Besides some modifications regarding velocity upgrade or velocity clamping can be made which will result in even faster performance. And obviously there's an option that we can perform some comparative study of PSO with other algorithms and find suitable options for solving engineering optimization problems.

8. References

[1] F. Hanssman, and S. Hess, A linear programming approach to production and employment scheduling, *Management Technology*, 1(1): 46-50, (1960).
[2] E.H. Bowman, Consistency and optimality in managerial decision making, *Management Science*, 9(2):310-321, (1963).
[3] W. Taubert, A search decision rule for the aggregate scheduling problem, *Management Science*, 143(6):343-359, (1968).
[4] J.M. Mellichamp, and R.M. Love, Production switching heuristics for aggregate planning problem, *Management Science*, 24(12): 1242-1251, (1978).
[5] A. Techawiboonwong, and P. Yenradee, Aggregate production planning with workforce transferring plan for multiple product types, *Production Planning & Control*, 14(5): 447-458, (2003).
[6] B.P. Das, J.G. Rickard, N. Shan, and S. Macchietto, An investigation of integration of aggregate production planning, master production scheduling and short-term production scheduling of batch process operations through a common data model, *Computers and Chemical Engineering*, 24: 1625-1631,(2000).
[7] D.J. Stockton, and L. Quinn, Aggregate production planning using genetic algorithms, *Proc. Instn. Mech. Engrs., Part B, Journal of Engineering Manufacture*, 209(B3): 201-208, (1995).
[8] D. Wang, and S.C. Fang, A genetics-based approach for aggregated production planning in a fuzzy environment, *IEEE Transactions on System, Man, and Cybernetics-Part A: Systems and Humans*, 27(5): 636-645, (1997).
[9] G.M. Kumar and N.N. Haq, Hybrid genetic-ant colony algorithms for solving aggregate production plan, *Journal of Advanced Manufacturing Systems*, 4(1): 103-111, (2005).
[10] G.T. Ioannis, Solving constrained optimization problems using a novel genetic algorithm, *Applied Mathematics and Computation*, Vol. 208, pp. 273–283, (2009).
[11] D. Bunnag & M. Sun, Genetic algorithm for constrained global optimization in continuous variables, *Applied Mathematics and Computation*, Vol. 171, pp. 604–636, (2005).
[12] R. Eberhart, and J. Kennedy, A new optimizer using particle swarm theory, *Proceeding of the 6th International Symposium on Micro Machine and Human Science*, pp. 39-43, (1995).
[13] H. El-Mounayri, Z. Dugla, and H. Deng, Prediction of surface roughness in end milling using swarm intelligence, *Proceedings of IEEE Swarm Intelligence Symposium*, pp. 220-227, (2003).
[14] C. Prakasvudhisarn, A particle swarm search for determination of minimum zone, *proceeding of the Seventh International Conference on Industrial Managemen*t, pp. 109-115, (2004).
[15] J. Kennedy, and R.C. Eberhart, A discrete binary version of the particle swarm algorithm, *IEEE International Conference on Systems, Man, and Cybernetics*, 5: 4104-4108, (1997).
[16] M.F. Tasgetiren, and Y.C. Liang, A binary particle swarm optimization algorithm for lot sizing problem, *Journal of Economic and Social Research*, 5(2): 1-20, (2003).
[17] Z. Lian, X. Gu, and B. Jiao, A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, *Chaos, Solitons, and Fractals, In press*, (2006).
[18] X. Hu, R.C. Eberhart, and Y. Shi, Engineering optimization with particle swarm, *Proceedings of the 2003 IEEE on Swarm Intelligence Symposium*, pp. 53-57, (2003).
[19] R.C. Wang, R.C & H. H. Fang, Aggregate production planning with multiple objectives in a fuzzy environment, *European Journal of Operational Research*, Vol. 133, pp. 521–536, (2001).
[20] R.C. Wang & T.F. Liang, Application of fuzzy multi-objective linear programming to aggregate production planning, *Computers and Industrial Engineering*, Vol. 46, No. 1, pp. 17–41, (2004).