

A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem

M. Fatih Taşgetiren* & Yun-Chia Liang**

Abstract. This paper presents a binary particle swarm optimization algorithm for the lot sizing problem. The problem is to find order quantities which will minimize the total ordering and holding costs of ordering decisions. Test problems are constructed randomly, and solved optimally by Wagner and Whitin Algorithm. Then a binary particle swarm optimization algorithm and a traditional genetic algorithm are coded and used to solve the test problems in order to compare them with those of optimal solutions by the Wagner and Whitin algorithm. Experimental results show that the binary particle swarm optimization algorithm is capable of finding optimal results in almost all cases.

Key Words: Particle swarm optimization; Lot sizing; Genetic algorithm; Evolutionary Algorithms.

Jel Codes: C60, C61.

1. Introduction

The lot-sizing problem attracted attention because of its impact on the inventory levels and, hence the inventory holding cost and the setup/ordering cost. It is basically concerned with finding order quantities minimizing the total cost of lot sizing decisions. Lot quantity might be either an amount of purchase or production depending on the problem domain on hand in order to

* Management Department, Fatih University, 34500 Buyukcekmece, Istanbul, Turkey.
Email: ftasgetiren@fatih.edu.tr

** Department of Industrial Engineering and Management, Yuan Ze University
No 135 Yuan-Tung Road, Chung-Li, Taoyuan County, Taiwan 320, R.O.C.

meet the net requirements of the customer demand. In lot sizing problems, time horizon is defined as given time buckets in which quantity decisions are generally given at the beginning of each time bucket. Lot sizing decisions are made in such a way that all customer requirements are met at the end of the time horizon.

In general, lot quantities are determined as the total requirement for a number of periods in which the total cost is minimized. It balances the tradeoff between the ordering and the holding costs. In other words, it depends on the requirement in the current period plus the requirements for the future periods. So the order quantity is determined by grouping the net requirements for a number of periods ahead. There exist different techniques to determine the lot quantities. Most popular one is the *lot-for-lot* where whatever needed is ordered. One other strategy is to order a *fixed order quantity*, which is common in industry, at each period regardless of any variation in the demand requirements. Another technique is to cover the net requirements for a number of future periods, called *fixed periods*. In addition, it is also possible to combine the different strategies together.

Several factors should be considered when lot-sizing decisions are given. These factors are ordering cost, holding cost, shortage cost, capacity constraints, minimum order quantity, maximum order quantity, quantity discounts and so on. Combination of these factors results in different models to analyze and different solution procedures are used depending on the model employed. The model and its solution procedure can be made complicated by considering these factors in the models, which can be classified as capacitated or uncapacitated, single-level or multi-level, single-item or multi-item models.

In this paper, we consider the uncapacitated, single-item, no shortages-allowed and single-level lot sizing model and solve it by a binary particle swarm optimization (PSO) algorithm. The rest of the paper is organized as follows: The next section formulates the problem, followed by a literature review. Then the binary PSO algorithm applied to lot sizing problem is given along with a traditional genetic algorithm, followed by some experimental results. Finally, conclusion and future work is presented.

2. Problem Formulation

The mathematical formulation of the lot sizing model considered in this paper is given as follows:

$$\min \left(\sum_{i=1}^n (Ax_i + cI_i) \right) \quad (1)$$

subject to :

$$I_0 = 0 \quad \forall_i \quad (2)$$

$$I_{i-1} + x_i Q_i - I_i = R_i \quad \forall_i \quad (3)$$

$$I_i \geq 0 \quad \forall_i \quad (4)$$

$$Q_i \geq 0 \quad \forall_i \quad (5)$$

$$x_i \in \{0,1\} \quad \forall_i \quad (6)$$

Where

n = number of periods

A = ordering/setup cost per period

c = holding cost per unit per period

R_i = net requirement for period i

Q_i = Order quantity for period i

I_i = projected inventory balance for period i

$x_i = 1$ if an order is placed in period i , $x_i = 0$ otherwise.

In the objective function (1), a penalty A is charged for each order placed along with a penalty c for each unit carried in inventory over the next period. Equation (2) guaranties that no initial inventory is available. Equation (3) is the inventory balance equation in which the order quantity Q_i covers all the requirements until the next order. Equation (4) satisfies the condition that no shortages are allowed. And finally, equation (5) shows the decision variable x_i to be either 1 (place an order) or 0 (do not place an order). It should be noted that initial inventory is zero, $I_0 = 0$, such that $x_1 = 1$ by equation (3) if $R_1 > 0$. Because of the minimization nature of the problem,

the ending inventory at each period is minimized to avoid the penalty charge c , particularly $I_n = 0$.

3. Literature Review

Different solution procedures are available to determine the lot quantities. The most popular one is the economic order quantity, EOQ (Mennell, 1961). Theoretically, EOQ minimizes the ordering and holding cost, but assumes that the requirements are constant or stationary from period to period. For the case of dynamic requirements in which the requirements are significantly variable over the periods, Silver and Meal (1973) presented a heuristic, so called Silver-Meal heuristic. The heuristic tries to minimize the ordering and holding costs per unit of time. Other heuristics are common in most production text books such as Least Unit Cost, LUC and Part Period Balancing, PPB. See Sipper and Bulfin (1997) for details. Wagner and Whitin, WW (1958) provided a dynamic programming to solve the problem optimally.

4. Binary PSO Algorithm For Lot Sizing Problem

Particle Swarm Optimization (PSO) is one of the evolutionary optimization methods inspired by nature which include evolutionary strategy (ES), evolutionary programming (EP), genetic algorithm (GA), and genetic programming (GP). PSO is distinctly different from other evolutionary-type methods in that it does not use the filtering operation (such as crossover and/or mutation) and the members of the entire population are maintained through the search procedure. In PSO algorithm, each member is called “particle”, and each particle flies around in the multi-dimensional search space with a velocity, which is constantly updated by the particle’s own experience and the experience of the particle’s neighbors. Since PSO was first introduced by Kennedy and Eberhart (1995, 2001), it has been successfully applied to optimize various continuous nonlinear functions. Although the applications of PSO on combinatorial optimization problems are still limited, PSO has its merit in the simple concept and economic computational cost.

The main idea behind the development of PSO is the social sharing of information among individuals of a population. In PSO algorithms, search is

conducted by using a population of particles, corresponding to individuals as in the case of evolutionary algorithms. Unlike GA, there is no operator of natural evolution which is used to generate new solutions for future generation. Instead, PSO is based on the exchange of information between individuals, so called *particles*, of the population, so called *swarm*. Each particle adjusts its own position towards its previous experience and towards the best previous position obtained in the swarm. Memorizing its best own position establishes the particle's experience implying a local search along with global search emerging from the neighboring experience or the experience of the whole swarm. Two variants of the PSO algorithm were developed, one with a global neighborhood, and other one with a local neighborhood. According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm, called *gbest* model. On the other hand, according to the local variant, called *lbest* model, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood (Kennedy, 2001). Kennedy and Eberhart (1997) also developed the discrete binary version of the PSO. PSO has been successfully applied to a wide range of applications such as power and voltage control (Abido, 2002), mass-spring system (Brandstatter & Baumgartner, 2002), and task assignment (Salman et al., 2003). The comprehensive survey of the PSO algorithms and applications can be found in Kennedy et al. (2001).

In this paper, we use the global variant with binary version applied to the simple lot sizing problem.

A. PSO algorithm

Pseudo code of the general PSO is given in Figure 1. In a PSO algorithm, population is initiated randomly with particles and evaluated to compute fitnesses together with finding the particle best (best value of each individual so far) and global best (best particle in the whole swarm). Initially, each individual with its dimensions and fitness value is assigned to its particle best. The best individual among particle best population, with its dimension and fitness value is, on the other hand, assigned to the global best. Then a loop starts to converge to an optimum solution. In the loop, particle and global bests are determined to update the velocity first. Then the current position of each particle is updated with the current velocity. Evaluation is again performed to compute the fitness of the particles in the swarm. This loop is terminated with a stopping criterion predetermined in advance.

```

Initialize parameters
Initialize population
Evaluate
Do{
    Find particlebest
    Find globalbest
    Update velocity
    Update position
    Evaluate
}while (Termination)

```

Figure 1. Simple PSO algorithm

The basic elements of PSO algorithm is summarized as follows:

- *Particle*: x_i^k is a candidate solution i in swarm at iteration k . The i^{th} particle of the swarm is represented by a d -dimensional vector and can be defined as $x_i^k = [x_{i1}^k, x_{i2}^k, \dots, x_{id}^k]$, where x 's are the optimized parameters and x_{id}^k is the position of the i^{th} particle with respect to d^{th} dimension. In other words, it is the value d^{th} optimized parameter in the i^{th} candidate solution.
- *Population*: pop^k is the set of n particles in the swarm at iteration k , i.e. $pop^k = [x_1^k, x_2^k, \dots, x_n^k]$
- *Particle velocity*: v_i^k is the velocity of particle i at iteration k . It can be described as $v_i^k = [v_{i1}^k, v_{i2}^k, \dots, v_{id}^k]$, where v_{id}^k is the velocity with respect to d^{th} dimension.
- *Particle best*: PB_i^k is the best value of the particle i obtained until iteration k . The best position associated with the best fitness value of the particle i obtained so far is called particle best and defined as $PB_i^k = [pb_{i1}^k, pb_{i2}^k, \dots, pb_{id}^k]$ with the fitness function $f(PB_i^k)$
- *Global best*: GB^k is the best position among all particles in the swarm, which is achieved so far and can be expressed as $GB^k = [gb_1^k, gb_2^k, \dots, gb_d^k]$ with the fitness function $f(GB^k)$

- *Termination criterion:* it is a condition that the search process will be terminated. In this study, search is terminated when the number of iteration reaches a predetermined value, called maximum number of iteration.

B. Solution representation

Solution representation of particle i , x_i^k , for the binary PSO is given in Figure 2. This representation is due to Hernandez and Suer (1999). Each particle has d dimensions referring to as the number of periods in the lot sizing problem. The dimension x_{id}^k indicates if an order is placed for particle i in period d at iteration k . In other words, x_{id}^k is a binary value such that $x_{id}^k = 1$ if lot sizing decision is given, $x_{id}^k = 0$ otherwise. R_d denotes the net requirements for the period d . v_{id}^k is the velocity of the particle i in period d at iteration k . Q_{id}^k is the lot size of the particle i in period d at iteration k and I_{id}^k is the inventory balance of particle i in period d at iteration k . Then the calculation of the cost of the ordering plan is trivial as shown in Figure 2 assuming the $c=\$1$ per unit per period and $A=\$100$ per order.

d	1	2	3	4	5	6	$f(X_i^k)$
R_d	100	60	40	50	80	70	
x_{id}^k	1	0	1	0	1	0	
v_{id}^k	3.8	2.9	3.0	-0.7	-1.2	3.1	
Q_{id}^k	160		90		150		
I_{id}^k	60		50		70		
cI_{id}^k	60		50		70		
Ax_{id}^k	100		100		100		
$c(X_i^k)$	160		150		170		480

Figure 2. Representation of the Solution

C. Initial Population

A population of particles is constructed randomly for the binary PSO algorithm for lot sizing problem. The values of dimensions are established randomly. For each dimension of a particle, a binary value of 0 or 1 is assigned with a probability of 0.5. In particular,

$$\begin{aligned} & \text{if } U(0,1) > 0.5, \text{ then } x_{id}^0 = 1 \\ & \text{else } x_{id}^0 = 0 \end{aligned}$$

Velocity values are restricted to some minimum and maximum values, namely $v_i^k = [v_{\min}, v_{\max}] = [-4, 4]$ where $v_{\min} = -v_{\max}$. The velocity of particle i in the d^{th} dimension is established by $v_{id}^0 = v_{\min} + (v_{\max} - v_{\min}) * \text{rand}()$. This limit enhances the local search exploration of the problem space. Population size is twice the number of dimensions. As the formulation of the lot sizing problem suggests, the objective is to minimize the total ordering and holding costs, the fitness function value for the particle i is given as follows:

$$f(X_i^k) = \sum_{j=1}^d (Ax_{ij}^k + cx_{ij}^k).$$

D. Finding new solutions

Since the binary version of the PSO algorithm is employed in this study, we need to use two useful functions for generating new solutions, namely a sigmoid function to force the real values between 0 and 1, and a piece-wise linear function to force velocity values to be inside the maximum and minimum allowable values. So whenever a velocity value is computed, the following piece-wise function, whose range is closed interval $[v_{\min}, v_{\max}]$, is used to restrict them to minimum and maximum value.

$$h(v_{id}^k) = \begin{cases} v_{\max}, & \text{if } v_{id}^k > v_{\max} \\ v_{id}^k, & \text{if } |v_{id}^k| \leq v_{\max} \\ v_{\min}, & \text{if } v_{id}^k < v_{\min} \end{cases}$$

After applying the piece-wise linear function, the following sigmoid function is used to scale the velocities between 0 and 1, which is then used for converting them to the binary values. That is,

$sigmoid(v_{id}^k) = \frac{1}{1 + e^{-v_{id}^k}}$. So, new solutions are found by updating the velocity

and dimension respectively. First, we compute the change in the velocity v_{id}^k such that

$$\Delta v_{id}^{k-1} = c_1 r_1 (pb_{id}^{k-1} - x_{id}^{k-1}) + c_2 r_2 (gb_d^{k-1} - x_{id}^{k-1})$$

Then we update the velocity v_{id} by using the piece-wise linear function such that

$$v_{id}^k = h(v_{id}^{k-1} + \Delta v_{id}^{k-1}).$$

Finally we update the dimension d of the particle i such that

$$x_{id}^k = \begin{cases} 1, & \text{if } U(0,1) < sigmoid(v_{id}^k) \\ 0, & \text{otherwise} \end{cases}$$

The complete computational flow of the binary PSO algorithm is given below:

Step 1: Initialization

- Set $k=0$, n =twice the number of dimensions
- Generate n particles randomly as explained before, $\{X_i^0, i=1,2,\dots,n\}$, where $X_i^0 = [x_{i1}^0, x_{i2}^0, \dots, x_{id}^0]$.
- Generate the initial velocities of all particles randomly, $\{V_i^0, i=1,2,\dots,n\}$, where $V_i^0 = [v_{i1}^0, v_{i2}^0, \dots, v_{id}^0]$. v_{id}^0 is generated randomly with $v_{id} = V_{\min} + (V_{\max} - V_{\min}) * rand()$
- Evaluate each particle in the swarm using the objective function, $f(X_i^0)$.
- For each particle i in the swarm, set $PB_i^0 = X_i^0$, where $PB_i^0 = [pb_{i1}^0 = x_{i1}^0, pb_{i2}^0 = x_{i2}^0, \dots, pb_{id}^0 = x_{id}^0]$ along with its best fitness value, $f_i^{pbest}(PB_i^0, i=1,2,\dots,n)$
- Set the global best to, $f^{gbest}(GB^0) = \min\{f_i^{pbest}(PB_i^0, i=1,2,\dots,n)\}$ with $GB^0 = [gb_1, gb_2, \dots, gb_d]$

Step 2: Update iteration counter

- $k = k + 1$

Step 3: Update velocity by using the piece-wise linear function

- $\Delta v_{id}^{k-1} = c_1 r_1 (pb_{id}^{k-1} - x_{id}^{k-1}) + c_2 r_2 (gb^{k-1} - x_{id}^{k-1})$ $v_{id}^k = h(v_{id}^{k-1} + \Delta v_{id}^{k-1})$

c_1 and c_2 are social and cognitive parameters and r_1 and r_2 are uniform random numbers between (0,1)

Step 4: Update dimension (position) by using the sigmoid function

- $x_{id}^k = \begin{cases} 1, & \text{if } U(0,1) < \text{sigmoid}(v_{id}^k) \\ 0, & \text{otherwise} \end{cases}$

Step 5: Update particle best

- Each particle is evaluated again with respect to its updated position to see if particle best will change. That is,

$$\text{If } f_i^k(X_i^k, i=1,2,\dots,n) < f_i^{pbest}(PB_i^{k-1}, i=1,2,\dots,n)$$

then

$$f_i^{pbest}(PB_i^k, i=1,2,\dots,n) = f_i^k(X_i^k, i=1,2,\dots,n)$$

else

$$f_i^{pbest}(PB_i^k, i=1,2,\dots,n) = f_i^{pbest}(PB_i^{k-1}, i=1,2,\dots,n)$$

Step 6: Update global best

$$f^{gbest}(GB^k) = \min\{f_i^{lbest}(PB_i^k, i=1,2,\dots,n)\}$$

$$\text{if } f^{gbest}(GB^k) < f^{gbest}(GB^{k-1}), \text{ then}$$

$$f^{gbest}(GB^k) = f^{gbest}(GB^k)$$

$$\text{else } f^{gbest}(GB^k) = f^{gbest}(GB^{k-1})$$

Step 7: Stopping criterion

- If the number of iteration exceeds the maximum number iteration, then stop, otherwise go to step 2.

5. Numerical Example

In order to illustrate how the binary PSO algorithm solves the lot sizing problem, the following problem with $d=5$ periods, $c=\$1$, $A=\$100$ is constructed as well as the net requirements given in Figure 2. The complete computational flow for illustration purpose is given below:

Step 1: Initialization

- $k=0$, and $n=3$
- The initial particles in the swarm generated randomly are $x_1^0, x_2^0, \text{and}, x_3^0$, and corresponding velocities $v_1^0, v_2^0, \text{and}, v_3^0$, which are given as follows:

	d	1	2	3	4	5
x_1^0	x_{1d}	1	1	0	0	0
v_1^0	v_{1d}	2.90	1.60	-1.80	3.5	-1.60
x_2^0	x_{2d}	1	0	1	0	0
v_2^0	v_{2d}	3.80	2.90	3.00	-0.70	-1.20
x_3^0	x_{3d}	1	0	0	1	1
v_3^0	v_{3d}	-3.0	1.50	-2.70	2.00	1.40

- Details of the computation of the fitness function for the particles are given in Figure 2. Now considering the first 5 periods of Figure 2, the fitness values of the particles in the swarm are:

$$f_1^0(x_1^0) = 580, \quad f_2^0(x_2^0) = 470, \quad f_3^0(x_3^0) = 440$$

- For each particle in the swarm, set the particle best to:

$$PB_1^0 = [pb_{11}^0 = p_{11}^0, pb_{12}^0 = x_{12}^0, \dots, pb_{1d}^0 = x_{1d}^0]$$

$$\text{with } f_1^{pbest}(PB_1^0 = x_1^0) = 580$$

$$PB_2^0 = [pb_{21}^0 = x_{21}^0, pb_{22}^0 = x_{22}^0, \dots, pb_{2d}^0 = x_{2d}^0]$$

$$\text{with } f_2^{pbest}(PB_2^0 = X_2^0) = 470$$

$$PB_3^0 = [pb_{31}^0 = x_{31}^0, pb_{32}^0 = x_{32}^0, \dots, pb_{3d}^0 = x_{3d}^0],$$

$$\text{with } f_3^{pbest}(PB_3^0 = X_3^0) = 440, \text{ so}$$

	d	1	2	3	4	5	$f_i^{pbest}(PB_i^0)$
PB_1^0	pb_{1d}	1	1	0	0	0	580
PB_2^0	pb_{2d}	1	0	1	0	0	470
PB_3^0	pb_{3d}	1	0	0	1	1	440

- Set the global best to:

$$f^{gbest}(GB^0) = \min\{f_i^{pbest}(PB_i^0)\} = f_3^{pbest}(PB_3^0) = 440$$

$$\text{with } GB^0 = [gb_1, gb_2, \dots, gb_d]. \text{ So}$$

	d	1	2	3	4	5	$f^{gbest}(GB^0)$
GB^0	gb_d	1	0	0	1	1	440

Step 2: Update the iteration number

- $k = 0 + 1 = 1$

Step 3: Update velocity by using the piece-wise linear function.

- Assume $c_1 = c_2 = r_1 = r_2 = 0.5$. As an example for particle 1, second dimension is updated as follows:

$$\Delta v_{12}^0 = 0.5 * 0.5 (pb_{12}^0 - x_{12}^0) + 0.5 * 0.5 (gb_2^0 - x_{12}^0)$$

$$\Delta v_{12}^0 = 0.5 * 0.5 (1 - 1) + 0.5 * 0.5 (0 - 1) = -0.25$$

$$v_{12}^1 = h(v_{12}^0 + \Delta v_{12}^0) = h(1.6 + (0 - 0.25)) = 1.35$$

Step 4: Update position by using the sigmoid function

Since $U(0,1) = 0.99 < \text{sigmoid}(v_{12}^1 = 1.35) = 0.79$

$$x_{12}^1 = 0$$

- After completing all the calculations for velocity and dimensions we have the following particles updated in the first iteration:

	<i>d</i>	1	2	3	4	5
x_1^1	x_{1d}^1	1	0	0	1	0
v_1^1	v_{1d}^1	2.90	1.35	-1.80	3.75	-1.60
	$\text{sig}(v_{1d}^1)$	0.95	0.79	0.14	0.98	0.17
	$u(0,1)$	0.12	0.99	0.90	0.60	0.34
x_2^1	x_{2d}^1	1	0	1	1	0
v_2^1	v_{2d}^1	3.10	2.17	2.25	-0.27	-0.90
	$\text{sig}(v_{2d}^1)$	0.96	0.90	0.90	0.43	0.29
	$u(0,1)$	0.89	0.99	0.12	0.21	0.63
x_3^1	x_{3d}^1	1	0	0	1	1
v_3^1	v_{3d}^1	-3.00	1.50	-2.70	2.00	0.90
	$\text{sig}(v_{3d}^1)$	0.05	0.82	0.06	0.88	0.71
	$u(0,1)$	0.02	0.98	0.19	0.34	0.49

Step 5: Update particle best

- Each particle updated is evaluated for first its new positions and then for finding the particle best as follows:

$$f_1^1(x_1^1) = 420, \quad f_2^1(x_2^1) = 440,$$

$$f_3^1(x_3^1) = 440$$

Since $f_1^1(x_1^1) = 420 < f_1^{pbest}(PB_1^0 = x_1^0) = 580$,

$$f_1^{pbest}(PB_1^1) = f_1^1(x_1^1) = 420 \quad \text{with} \quad PB_1^1 = x_1^1.$$

Since $f_2^1(X_2^1) = 440 < f_2^{pbest}(PB_2^0 = X_2^0) = 470$, $f_2^{pbest}(PB_2^1) = f_2^1(X_2^1) = 440$ with $PB_2^1 = X_2^1$.

Since $f_3^1(X_3^1) = 440 = f_3^{pbest}(PB_3^0 = X_3^0) = 440$, $f_3^{pbest}(PB_3^1) = f_3^1(X_3^1) = 440$ with $PB_3^1 = X_3^1$.

Note that neutral moves are allowed. After completing the similar comparisons for the other particles, we have

	d	1	2	3	4	5	$f_i^{pbest}(PB_i^1)$
PB_1^1	pb_{1d}	1	0	0	1	0	420
PB_2^1	pb_{2d}	1	0	1	1	0	440
PB_3^1	pb_{3d}	1	0	0	1	1	440

Step 6: Update global best

$$f^{gbest}(GB^1) = \min\{f_1^{pbest}(PB_1^1), f_2^{pbest}(PB_2^1), f_3^{pbest}(PB_3^1)\}$$

$$= f_3^{pbest}(PB_3^1) = 420$$

since, $f^{gbest}(GB^1) = 420 < f^{gbest}(GB^0) = 440$, then

$$f^{gbest}(GB^1) = f^{gbest}(GB^0) = 420$$

$$\text{with } GB^1 = \{gb_1^1, gb_2^1, \dots, gb_d^1\}$$

So, the global best is:

	d	1	2	3	4	5	$f^{gbest}(GB^0)$
GB^0	gb_d	1	0	0	1	0	420

Step7: Stopping criterion

if $k < \max \text{ iteration}$, goto step2

else stop

6. Experimental Results

The binary PSO algorithm presented for the uncapacitated lot sizing problem is coded in C and run on an Intel P4 2.6 GHz, 256 machine. The performance of the binary PSO is measured with a traditional GA and the optimal Wagner and Whitin algorithm. For this purpose, a traditional GA is coded in C and test problems are generated randomly for experimentation. The GA is a traditional one with a uniform crossover, simple inversion mutation and a tournament selection of size 2. We used the following parameters for the binary PSO and the traditional GA. For the binary PSO, the size of the population in the swarm is taken as the twice the number of periods. Social and cognitive parameters are taken as $c_1 = c_2 = 2$ consistent with the literature. For the GA, the population size is the same as the binary PSO. The crossover and the mutation rates are 0.70 and 0.10 respectively. Both GA and PSO algorithms are run for 1000 generations/iterations.

First test suit consisting of 10 problem instances with net requirements for 50 periods is generated from a uniform distribution, $UNIF(50,250)$, and the second test suit consisting of 10 problem instances with net requirements for 50 periods is generated from a uniform distribution, $UNIF(100,250)$. The total 20 problem instances are run for both GA and the binary PSO with holding cost of $c = \$0.50$ and ordering cost of $A = \$100$ in order to compare the results with those optimals by the Wagner-Whitin algorithm. For each problem instance, 10 replications are conducted. Minimum, maximum, average, and standard deviation are given together with the CPU times. As can be seen from Table 1 and 2, the binary PSO produced comparable results with GA, and it even produced better results. The GA was able to find the 14 optimal solutions out of 20 while the binary PSO was able find the 19 optimal solutions out of 20 even though the average standard deviation of the GA over 20 replications was slightly better than the binary PSO, i.e., $\overline{\sigma}_{GA} = 6.48$, $\overline{\sigma}_{PSO} = 6.94$. In terms of the computational time, the GA took approximately 10 seconds for each instance while PSO took 16 seconds, which is computationally expensive than GA. But PSO's good performance on finding optimal solutions more than GA compensates its computational inefficiency.

7. Conclusions and Future Works

We do realize that the problem presented here can be solved to optimality by Wagner and Whitin (1958) algorithm. Most research on the PSO concentrates on the continuous optimization problems. The objective of this paper is to present the potential power of the binary PSO to solve the binary/discrete optimization problems, which has a variety of applications in the real world.

During the past several years, the PSO has been successfully applied to the continuous optimization problems. To the best of our knowledge, this is the first reported application of the binary PSO applied to the lot sizing problem. The results are encouraging to apply the binary PSO to the combinatorial optimization problems formulated as $(0, 1)$ integer programming. In addition, we hope that this work leads to the application of the PSO to the much more complex, NP-hard lot sizing problems. The advantages of the PSO are very few parameters to deal with and the large number of processing elements, so called dimensions, which enable to fly around the solution space effectively. On the other hand, it converges to a solution very quickly which should be carefully dealt with when using it for combinatorial optimization problems. As a future work, the binary PSO can be applied to the capacitated, multi-item, multi-level lot sizing problems.

Table1. PSO Results

P	WW	PSO			
	Opt	Best	Max	Avg	Std
P01	4211.00	4211.00	4223.50	4216.00	6.45
P02	4354.00	4354.00	4371.50	4357.55	5.39
P03	4102.00	4102.00	4102.00	4102.00	0.00
P04	4078.00	4078.00	4078.00	4078.00	0.00
P05	4022.50	4022.50	4032.50	4023.50	3.16
P06	4249.00	4249.00	4259.00	4251.00	4.22
P07	4069.00	4069.50	4093.00	4071.85	7.43
P08	4301.00	4301.00	4306.00	4301.50	1.58
P09	4433.00	4433.00	4435.00	4433.20	0.63
P10	4024.00	4024.00	4048.00	4031.80	7.10
P11	4510.00	4510.00	4527.00	4515.40	5.89
P12	4494.00	4494.00	4531.50	4502.65	13.33
P13	4424.00	4424.00	4441.00	4427.30	5.41
P14	4480.50	4480.50	4541.50	4503.80	23.03
P15	4451.00	4451.00	4492.50	4465.35	12.60
P16	4493.00	4493.00	4511.50	4501.75	8.32
P17	4353.50	4353.50	4354.50	4354.00	0.47
P18	4420.00	4420.00	4489.00	4444.60	25.48
P19	4415.00	4415.00	4438.00	4421.80	9.16
P20	4397.00	4397.00	4414.50	4400.10	6.60

Table 2. GA Results

P	WW	GA			
	Opt	Best	Max	Avg	Std
P01	4211.00	4211.00	4270.00	4228.80	17.70
P02	4354.00	4354.00	4371.50	4359.80	6.78
P03	4102.00	4102.00	4110.50	4103.70	3.58
P04	4078.00	4078.00	4102.50	4089.70	12.28
P05	4022.50	4022.50	4032.50	4025.50	4.83
P06	4249.00	4249.00	4293.00	4257.80	14.67
P07	4069.00	4069.50	4093.00	4071.85	7.43
P08	4301.00	4301.00	4306.00	4302.50	2.42
P09	4433.00	4433.00	4438.50	4433.95	1.80
P10	4024.00	4024.00	4056.00	4037.75	10.82
P11	4510.00	4512.00	4540.00	4520.75	10.47
P12	4494.00	4495.50	4540.50	4511.35	12.36
P13	4424.00	4424.00	4474.00	4447.30	17.64
P14	4480.50	4483.00	4510.50	4494.85	8.72
P15	4451.00	4451.00	4485.00	4460.00	11.26
P16	4493.00	4498.00	4559.50	4520.25	23.52
P17	4353.50	4353.50	4406.50	4373.95	19.82
P18	4420.00	4429.00	4486.00	4451.85	22.51
P19	4415.00	4415.00	4461.50	4428.75	18.33
P20	4397.00	4397.00	4421.50	4409.70	11.24

REFERENCES

- Abido M. A., "Optimal Power Flow Using Particle Swarm Optimization", *Electrical Power and Energy Systems*, 24(2002) 563-571.
- Brandstatter B. and Baumgartner U., "Particle Swarm Optimization—Mass-Spring System Analogon," *IEEE Transactions on Magnetics*, vol.38, pp.997-1000, 2002.
- Sipper and Bulfin, "Production Planning, Control and Integration" , *McGraw-Hill*, 1997.
- Hernandez W. and Suer G. A, "Genetic Algorithms in Lot Sizing Decisions", *Proceedings of the Congress on Evolutionary Computation CEC99*, pp. 2280-2286, 1999.
- Kennedy J. and Eberhard R. C., "Particle Swarm Optimization," *Proc. of IEEE Int'l Conf. on Neural Networks*, Piscataway, NJ, USA, pp. 1942-1948, 1995.
- Kennedy J. and Eberhart R. C., "A Discrete Binary Version of the Particle Swarm Optimization", *Proc. Of the conference on Systems, Man, and Cybernetics SMC97*, pp. 4104-4109, 1997.
- Kennedy J., Eberhart R. C. and Shi Y., "Swarm Intelligence", *Morgan Kaufmann*, San Mateo, 2001, CA
- Mennell, R. F, "Early History of the Economical Lot Size", *APICS Quarterly Bulletin*, Vol.2, No.2, 1961
- Salman A., Ahmad I., and Al-Madani S., "Particle Swarm Optimization for Task Assignment Problem", *Microprocessors and Microsystems*, 26(2002) 363-371.
- Silver E. A and Meal H. C, "A Heuristic for Selecting Lot Size Requirements for the Case of A Deterministic Time-Varying Demand Rate and Discrete Opportunities for Replenishment", *Production and Inventory Management*, Vol. 14, No. 2, 1973

Wagner H. M and Whitin T. M, “Dynamic Version of the Economic Lot Size Model”, *Management Science*, Vol. 5, 1958