International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

An Iterative Improvement Search and Binary Particle Swarm Optimization for Large Capacitated Multi Item Multi Level Lot Sizing (CMIMLLS) Problem

V.V.D.Sahithi[1] ,P.Sai Krishna[2] ,K.Lalithkumar[3] ,C.S.P.Rao[4]

1 Research assistant Professor Department of mechanical engineering, VNR Vignana Jyothi Institute Of Engineering And Technology, Hyderabad 500090, India

2,3Research Student, Department mechanical Engineering, VNR Vignana Jyothi Institute of Engineering And Technology, Hyderabad 500090, India

4 Professor Department of Mechanical Engineering, National Institute of Technology Warangal, Warangal 506004, India

sahithi.vaka@gmail.com[1] lalith.143143@gmail.com[2] Saikrishna.parikirala@gmail.com [3] csp_rao@rediffmail.com[4]

Abstract

Lot sizing problem in Material Requirement Planning (MRP) systems belongs to those problems that industrial manufacturers face daily in organizing their overall production plans. Lot sizing plays an important role in minimization of totalcost (i.e. sum of setup and holding cost). When multiple levels, multiple items and capacity restrictions are involved in an inventory lot sizing problem, determination of optimum lot sizes becomes very complicated and may be treated as NP hard class of problems. However this combinatorial optimization problem can be solved by using soft computing techniques in a reasonable CPU time when small instances are considered. Many heuristic techniques were developed in the past to solve lot sizing problems but most of them were failed in successful implementation. In this paper the authors are presenting an Iterative improvement binary particle swarm optimization (IIBPSO) techniques for solving very large capacitated multi item multi level lot sizing problem (CMIMLLS). In the proposed algorithm first a set of initial solution is randomly choosen then, used the particles to find solution according to standard mechanism of binary particle swarm optimization (BPSO). After reaching a reasonable solution point, a hybrid selection with iterative improvement local search mechanism is applied to restart the algorithm. Hybrid selection is a kind of restart mechanism in BPSO, and finally a local search is used on the global best solution to improve the solution quality. The IIBPSO algorithm showed good experimental results and outperforms all other approaches in terms of quality of solution.

Keywords: inventory lot sizing, material requirement planning, hybrid particle swarm optimization.

1. Introduction

In most manufacturing and distribution companies, the highest individual cost is inventory. The cost of the inventory is directly related to the amount of inventory held and the bulk of manufacturers admit they consistently carry too much of it. Executives usually believe that the higher the service level is, the more stock is required. However, as demand forecasts are often inaccurate, inventory piles up, exposure to obsolescence increases, salable throughput decreases and customer service finally declines. Thus too much inventory further compounds service problems. But inventory reduction may be carried out at the expense of an increased cadence of orders. Unfortunately setup costs cannot come to zero even if they have been considerably reduced on grounds of just-in- time (JIT) guidelines. Thus critical to achieving a satisfactory trade-off between set-up costs and inventory holding costs is to implement proper lotsizing rules. But even the most comprehensive MRP systems do not provide an efficient methodology. In fact, commercially-available MRP software typically comes with the simplest yet suboptimal lot-sizing approaches [1].

Lot sizing problem attracted the attention because of its impact on the inventory levels and hence the total cost of production. It is basically concerned with finding order quantities of different items in the bill of material structure to minimize the setup cost and holding cost. Lot size might be the amount of production or purchase quantity depending on the demand at different time buckets to ensure and satisfy customer requirements [2]. Minimizing total production cost is always a tradeoff decision between ordering and holding cost. In the decision making, a number of factors need to be considered; carrying cost, setup cost, shortage cost, capacity restrictions, minimum order quantity, maximum order quantity, handling restrictions, quantity discounts, etc. All these factors can be combined to generate different models. For instance, some of the costs can be considered zero or infinity and some of the restrictions can be relaxed. Depending on the applicable model, different solution procedures exist. The model is complicated, along with its corresponding solution procedure, by the number of items considered, i.e., single item and multi item considerations. Another possible complication in the model is the inclusion of multi-levels consideration and capacity constraints. Hence the problem of lot sizing still stood as challenging problem of optimization and attracted research community.

The lot sizing problems can be mainly divided into Single level lot sizing problems (SLLS) and Multi-level lot sizing (MLLS) problems with and without capacity restrictions. SLLS problems without capacity restriction are simplest among them. Several heuristics were developed and successfully implemented on SLLS problems. In 1958, Wagner and Whitin (2004) introduced the SLLS model and developed a well-known exact algorithm based on dynamic programming]After that, Silver and Meal (1973) proposed the idea of minimizing average setup and inventory costs over several periods. Mc Knew and Coleman (1991) proposed a part period algorithm for minimizing setup and holding cost over different periods. Hernández, W. and G. Süer, proposed a genetic algorithm (GA) for solving single level uncapacitated lot sizing problem with no shortages. A few heuristics techniques were also developed to solve MLLS problems. N.Dellart, J.Jeunet successfully applied a Randomized multi-level lot-sizing heuristics for general product structures. Regina Berretta, Luiz Fernando Rodriguez proposed A memetic algorithm for a multi stage capacitated lot sizing problem .Taˇsgetiren and Liang presented particle swarm optimization (PSO) in 2003 to minimize the inventory setup and holding cost for minimization of simple product structures N.Dellart, J.Jeunet, N.Jonard successfully applied PSO for uncapacitated multi level lot sizing problem with flexible initial weight .Klorklear Wajanawichakon and Rapeepan Pitakaso implemented PSO (2011) for multi level unconstrained problems of general product structures.

In this paper, the authors have made an attempt to solve very large and complex product structure of capacity constrained multi item multi level lot sizing problem (MIMLLS). An iterative improvement search with BPSO approach is used to simulate CMIMLLS problem and solved several problems with time and solution efficiency. The authors have also solved the problems considered using Genetic Algorithm, BPSO and IIBPSO separately. The results of Binary GA, Iterative Improvement BGA (IIBGA) and BPSO are compared with the proposed method IIBPSO for the same set of problems under consideration. The Paper is organized in six sections: section2: mathematical formulation of CMIMLLS problem section3: IIBPSO procedure Section 4: numerical example section5: problem illustration and section6: conclusion is presented.

2. Mathematical Formulation of problem:

The lot sizing problem that we considered in this paper can be described as follows. We have 'N' items to be produced in 'T' periods in a planning horizon such that a demand forecast would be attained .In a multistage production systems ,the planning horizon of each item depends on the production of other items, which are situated at lower levels. The resources for production and setup are limited. Lead times are assumed to be zero.

Let N be the number of items, T the number of periods in the planning horizon the number of types of resources. C_{it} the unit production cost item I in period t, h_{it} the unit holding cost of item I in period t,S_{it} is the setup cost of item i in period t,d_{it} the demand for item I in period t,V_{ikt} the amount of resource k necessary to produce item i in period t, b_{kt} is the amount of resource k available in period t, M is the upper bound on X_{it} ,S(i) the set of immediate successor items to item I, and r_{ij} is the number of units of item i needed by one unit of item j, where j∊ S(i).

Decision variables are x_{ij} is the lot size of item i in period t, y_{it} is '1' if item is produced in period t and zero otherwise. I_{it} the inventory of item i in period t.

$$\text{Min } (f(x)) = \sum_{i=1}^{N} \sum_{t=1}^{T} (C_{it}X_{it} + h_{it}I_{it} + S_{it}Y_{it}) \quad \text{......(1)}$$

$$I_{i,t-1} + X_{it} - I_{it} = d_{it} + \sum_{j\epsilon S(i)} r_{ij}X_{jt} \quad \text{...... (2)}$$

i=1, 2,…….N ; T=1,2,………T

$$\sum_{i=1}^{N} (V_{ikt}X_{it} + f_{ikt}y_{it}) \leq b_{kt} \quad \text{...... (3)}$$

k=1, 2, 3,…….K; t=1,2,3,…….T

$$X_{it} \leq My_{it} \qquad i=1,\ldots N; \quad t=1,\ldots,T \quad \text{......(4)}$$

$$X_{it}, I_{it} \geq 0 \qquad i=1,\ldots N; \quad t=1,\ldots,T \quad \text{......(5)}$$

$$y_{it} \in \{0,1\} \qquad i=1\ldots N; \quad t=1,\ldots,T \quad \text{......(6)}$$

The objective function (1) is to minimize the sum of production, inventory holding and setup cost in T periods. Equation (2) is inventory balance constraint, which describe the relationship between inventory and production at the beginning and the end of the period. Constraint (3) represents the capacity limitations of production and setup. Constraint(4) ensure that the solution will have setup when it has production .The last two constraints (5) and (6) require that variables must be positive and setup variables must be binary**.**

Several factors like ordering cost, holding cost, shortage cost, capacity constraints, minimum and maximum order quantity etc... Combination of these factors result in different models to be analyzed like capacitated or uncapacitated, single level or multi level, single item or multi item models.simple single product structures can be solved easily using mathematical equations .as CMIMLLS problems are having very large solution space they are considered as NP-hard problems that does not have solution with polynomial time. So soft computing techniques are necessary to compute optimum values of lot sizes.

In this paper authors have made an attempt to solve very large complex product structure of capacity constrained multi product multi level lot sizing problem. An iterative improvement binary PSO approach is used to simulate CMIMLLS problem and solved the same with time and solution efficiency. The authors have also solved similar problems using BGA, IIBGA, and BPSO. The results of BGA, IIBGA, BPSO, and IIBPSO are compared for the same set of problems under consideration.

3. Iterative Improvement Search Binary Particle Swarm Optimization (IIBPSO) Procedure:

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 is basically developed through simulation of bird flocking in the two dimensional space and 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. If binary values (0 or 1) are used as particle dimensions it is called as Binary Particle Swarm Optimization (BPSO).

Even though we might find a good set of parameters for BPSO, Iterative Improvement search is still worth while trying to improve the performance of the solution. Local search algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed and helps to escape from local minima. Iterative Improvement search is one such local search algorithm which helps in improving solution efficiency.

(a) Initialization

In PSO algorithm, each member is called particle and each one represents one particular solution to the given problem. Group of particles is called as swarm.

(i) Initialization of particle

In multi level inventory problems each particle is represented by a matrix of m×n. where m represents the number of items involved in the problem, represents time buckets. And particle representation is $X^{pt}{}_{id}$.

Here p= particle number.

t=iteration number (represents row number)

i=item number (represents column number)

d=time period.

Example:

7 items and 6 periodic demands are involved in the problem then particle is represented by 7×6 matrix.

As it is initial generation, all dimentions of particle are assigned to "0" or "1" randomly.

If R $_{>}$ 0.5 then $X^{pt}{}_{id}$ =1.

Else $X^{pt}{}_{id}$=0.

Here R represents a random number.

$$\text{Particle} = X_{id}^{pt} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure. 1 Particle dimension representation

$X^{pt}{}_{id}$ represents p^{th} particle of t^{th} iteration and swarm contains p different particles like this.

(ii) According to particle dimensions, fitness needs to be calculated for each and every particle, i.e. fitness ($X^{pt}{}_{id}$).

(iii) Initialization of particle velocities

After defining particle dimensions particle velocities needs to be calculated. For initial generation velocity calculation can be done using following formula

$$\mathbf{V^{p0}{}_{id} = V_{mini} + (V_{maxi} - V_{mini}) * R}$$

here [$V_{maxi,}$ V_{mini}]=[-x,x],here x is an integer.

Ex: let [$V_{maxi,}$ V_{mini}]=[-5,5]

$$V_{id}^{p0} = \begin{bmatrix} +1.5 & -2.4 & -3.8 & +2.5 & +4.3 & +1.5 \\ -0.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ +2.5 & +0.6 & +0.2 & -1.4 & -1.2 & +3.2 \\ -1.1 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\ +2.2 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ -1.3 & +0.0 & +1.2 & -1.6 & +4.3 & +0.3 \\ +0.7 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix}$$

Figure. 2 Particle velocity representation

(b) Updating Particle best and global best

After defining swarm i.e. all particle dimensions, fitness needs to be calculated. After calculating fitness value we need to assign global best value to the paticle containing best fitness value. As it is the initial generation all particle best($PB^{p,k}{}_{id}$) values are equal to particle values.

Here $GB^{t}{}_{id}$ represents global best dimensions of t^{th} iteration .

Here $PB^{pt}{}_{id}$ represents paticle best dimensions of p^{th} particle t^{th} iteration.

(c)Updating parameters for next generations

(i)Updating velocity of particle ($V^{pt}{}_{id}$):

New velocity = $V^{pt}{}_{id} = P\,(V^{p,\,t-1}{}_{id} + \Delta V^{p,\,t-1}{}_{id})$

$$\Delta V^{p,\,t-1}{}_{id} = c1\ R1\ (PB^{p,t-1}{}_{id} - X^{p,t-1}{}_{id}) + c2\ R2 (GB^{t-1}{}_{id} - X^{p,t-1}{}_{id})$$

C1, c2 are social and cognitive parameters, R1& R2 are uniform random numbers between (0, 1)

Here Piece wise linear function [P ($V^{pt}{}_{id}$)]

$$P\,(V^{pt}{}_{id}) = \begin{cases} V\text{maxi} & \text{if } V^{pt}{}_{id} > V\text{maxi} \\ V^{pt}{}_{id} & \text{if } |V^{pt}{}_{id}| \leq V_{maxi} \\ V\text{mini} & \text{if } V^{pt}{}_{id} < V\text{mini} \end{cases}$$

(ii)Updating position ($X^{pt}{}_{id}$) by sigmoid function:

$$X^{pt}{}_{id} = \begin{cases} 1 & \text{if } R < S\,(V^{pt}{}_{id}) \\ 0 & \text{otherwise} \end{cases}$$

Sigmoid function S ($V^{pt}{}_{id}$):

This function forces velocity values to be in the limits of '0' to '1'.It helps to update next generation $X^{pk}{}_{id}$ values.

$$S\,(V^{pt}_{id}) = \frac{1}{1+e^{-V^{pt}_{id}}}$$

(iii)Updating particle best and global best ($PB^{p,t}_{i,d}$, $GB^{t}_{i,d}$)

After each and every iteration update particle best and global best values according to the fitness values of particles in the newly generated swarm.

(d)Iterative Improvement Search Algorithm

Iterative Improvement Search Algorithm is a local search that moves from one solution S to another S' according to some neighborhood structure. Search procedure usually consists of the following steps.

(i) Initialization: Choose an initial schedule S to be the current solution and compute the value of the objective function F(S).

(ii) Neighbour Generation: Select a neighbour S' of the current solution S and compute F(S').

(iii) Acceptance Test: Iterative Improvement allows only strict improvement in the objective function value. It accepts a new solution S' only if $F(S') < F(S)$, where S is the current solution. Often instead of accepting the first neighbour with the value of the objective function smaller than $F(S)$ for the current solution, the algorithm constructs all neighbours (or a given number of Neighbours) and selects the best one.

(iv) Update particle best and global best values.

(e) Termination:

If the number of iterations reaches a predetermined value, called maximum number of iterations then stop searching, other wise go to (c) and repeat the procedure.

Pseudo code of IIBPSO is given in Figure3.

STEP1: Initialization phase

```
        Initialize swarm
        Assign velocities to all paticle
        Fitness calculation
        Particle best and global best

STEP2: Iteration phase with IIBPSO search
for (i=0; i<number of iterations; i++)
 {
        Update particles velocities
        Update dimensions of particles
        Calculate Fitness values
        Update Particle and global best values
        Iterative improvement local search
        Update Particle and global best
}
STEP3: Iteration phase by local search for global best value
for (i=0; i<number of iterations; i++)
 {
  Iterative improvement local search
}
```

Figure 3. Pseudo code of IIBPSO algorithm

4. Numerical Example:

A lot sizing problem of 7 items and 6 periods is taken from Jinxing Xie, Jiefang which is a general capacitated lot sizing problem (2002), and this example is also taken for the comparison with other problem considered in the paper.

M.Fatih Tasgetiren and Yun-Chia Liang (2003) say that if population size (number of particles in swarm) is at least double the number of periods in the planning horizon performance would be better. According to Yuhui Shi (2004), PSO with minimum population size 5 gives better performance.

But for the sake of convenience swarm size i.e. population size is taken as 3 in numerical example, even though all the problems are solved with population size of 40.

Step1: Swarm contains 3 particles, each of size 7×6

$$\text{Particle1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} +4.3 & -2.4 & -3.8 & +2.5 & +4.3 & +1.5 \\ -3.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ -1.2 & +0.6 & +0.2 & -1.4 & -1.2 & +3.2 \\ -2.0 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\ +3.6 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ +4.3 & +0.0 & +1.2 & -1.6 & +4.3 & +0.3 \\ +1.2 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 10948$$

$$\text{Particle2} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} +1.5 & +3.4 & -1.8 & +2.5 & +4.3 & +1.5 \\ -0.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ +2.5 & +1.6 & +0.2 & -1.4 & -1.2 & -3.2 \\ -1.1 & +0.5 & -3.5 & -0.9 & +2.0 & +0.9 \\ +2.2 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ -1.3 & +0.0 & +1.2 & -1.6 & +4.3 & +0.3 \\ +0.7 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 11648$$

$$\text{Particle3} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} +0.5 & +5.4 & -1.8 & +2.5 & +3.3 & +2.5 \\ -0.1 & -2.4 & -1.4 & -0.9 & -3.6 & +1.4 \\ +2.5 & +0.6 & +1.2 & -1.4 & -3.2 & -3.2 \\ -3.1 & +0.5 & -3.5 & -1.9 & +2.0 & +0.9 \\ +2.2 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ -1.3 & +0.0 & +2.2 & -1.6 & +4.3 & +0.3 \\ +0.7 & -3.4 & +0.2 & -4.2 & +1.2 & -1.7 \end{bmatrix} \rightarrow \text{Fitness} = 9376$$

Step2: As it is first generation assign all particle values to particle best, and best fitness particle dimensions to global best value

$$\text{PB}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad PB_2 \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad PB_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{Global Best} = \text{GB} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Step3:

Update Velocity using standard procedure of Binary particle swarm optimization

$$\text{Particle1} = \begin{bmatrix} +4.3 & -2.4 & -3.8 & +2.5 & +4.3 & +0.04 \\ -3.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\ -1.2 & +0.6 & +0.2 & -1.4 & -1.2 & +1.74 \\ -2.0 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\ +3.6 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\ +4.3 & +0.0 & +1.2 & -1.6 & +4.3 & -1.16 \\ +1.2 & -4.86 & +1.66 & -4.2 & +1.2 & -3.16 \end{bmatrix} \rightarrow \begin{bmatrix} 0.98 & 0.08 & 0.02 & 0.92 & 0.98 & 0.50 \\ 0.02 & 0.19 & 0.19 & 0.28 & 0.02 & 0.98 \\ 0.23 & 0.64 & 0.54 & 0.19 & 0.23 & 0.85 \\ 0.11 & 0.62 & 0.02 & 0.13 & 0.11 & 0.28 \\ 0.97 & 0.07 & 0.01 & 0.21 & 0.97 & 0.98 \\ 0.98 & 0.50 & 0.76 & 0.16 & 0.98 & 0.23 \\ 0.76 & 0.00 & 0.84 & 0.01 & 0.76 & 0.04 \end{bmatrix}$$

$$\text{Sigmoid}(\text{V} + \Delta\text{V}) = \begin{bmatrix} 0.98 & 0.08 & 0.02 & 0.92 & 0.98 & 0.50 \\ 0.02 & 0.19 & 0.19 & 0.28 & 0.02 & 0.98 \\ 0.23 & 0.64 & 0.54 & 0.19 & 0.23 & 0.85 \\ 0.11 & 0.62 & 0.02 & 0.13 & 0.11 & 0.28 \\ 0.97 & 0.07 & 0.01 & 0.21 & 0.97 & 0.98 \\ 0.98 & 0.50 & 0.76 & 0.16 & 0.98 & 0.23 \\ 0.76 & 0.00 & 0.84 & 0.01 & 0.76 & 0.04 \end{bmatrix} R = \begin{bmatrix} 0.0 & 0.5 & 0.09 & 0.90 & 0.99 & 0.71 \\ 0.0 & 0.3 & 0.99 & 0.11 & 0.33 & 0.99 \\ 0.0 & 0.72 & 0.81 & 0.89 & 0.54 & 0.89 \\ 0.0 & 0.92 & 0.00 & 0.93 & 0.33 & 0.37 \\ 0.0 & 0.10 & 0.80 & 0.10 & 0.98 & 0.99 \\ 0.0 & 0.65 & 0.84 & 0.97 & 0.99 & 0.37 \\ 0.0 & 0.5 & 0.98 & 0.70 & 0.85 & 0.35 \end{bmatrix}$$

Update particle dimension matrix according new velocity matrix of paticle

$$\text{New Particle1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \rightarrow \text{Fitness} = 10300$$

As particle 1 fitness value is improved, so first particles, particle best (PB_1) value will be updated with current particle data. If fitness is not improved then particle best value will remain same.

Like this update particle best and global best values will be updated for all particles in the according to fitness values.

Step4: Repeat this procedure until iteration number k < max iteration.

Local Search:

$$\text{input Particle} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \text{new paticle} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \text{Fitness} = 9820$$

Fitness value of new particle is improved (10300>9820). As the solution is improved old particle (i.e. input particle) will be replaced with a new particle.

Step5:

After this goto step2 and repeat the procedure. If number of iterations are reached stop

5. Problem Illustration

Problems shown in Fig. 4a, 4b and 4c as M×T are taken for modeling and simulation of CMIMLLS problem. Here M represents the total number of items involved in the BOM structure and T represents the number of periods. Table 1 represents different costs involved and Table 2a,2b and 2c carries information regarding demand and available capacity. Figure 4a is a BOM of single product where it contains 50×12 structure with 50 different items, 12 periods in 9 levels, Figure 4b is a BOM of a multi product contains 39×12 structure with 39 different items, 12 periods in 6 levels and Figure 4c is a BOM of a multi product contains 75×36 structure with 75 different items, 36 periods in 10 levels. Table 1 gives the information regarding the setup cost (S.C.) and holding costs (H.C.) of different items of 50 ×12, 39×12 and 75×36 problems. Tables 2a, 2b and 2c give the information regarding demand and availability conditions.

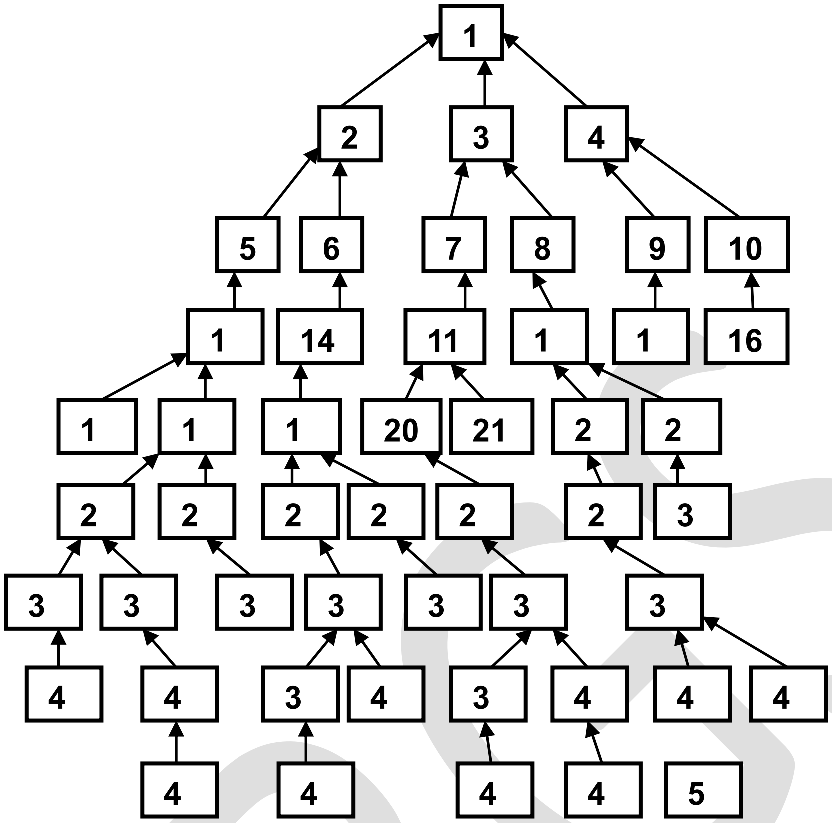

Figure 4a Product structures of 50×12 single product problem

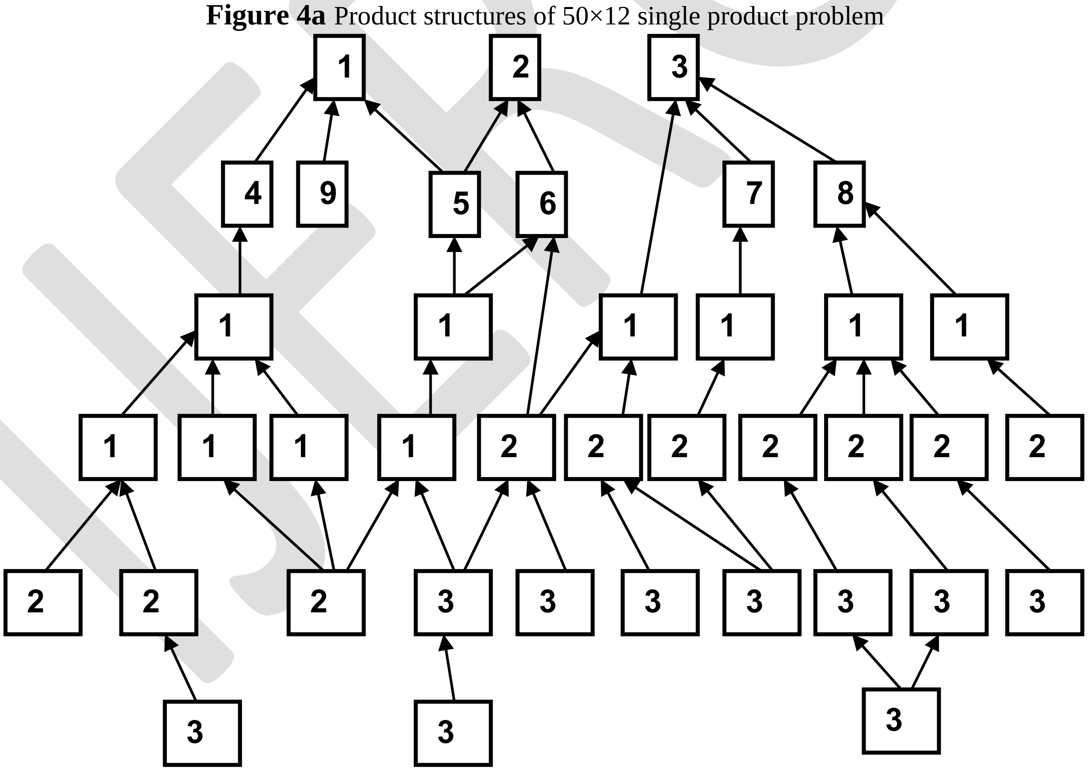

Figure 4b Product structures of 39×12 multi product problem

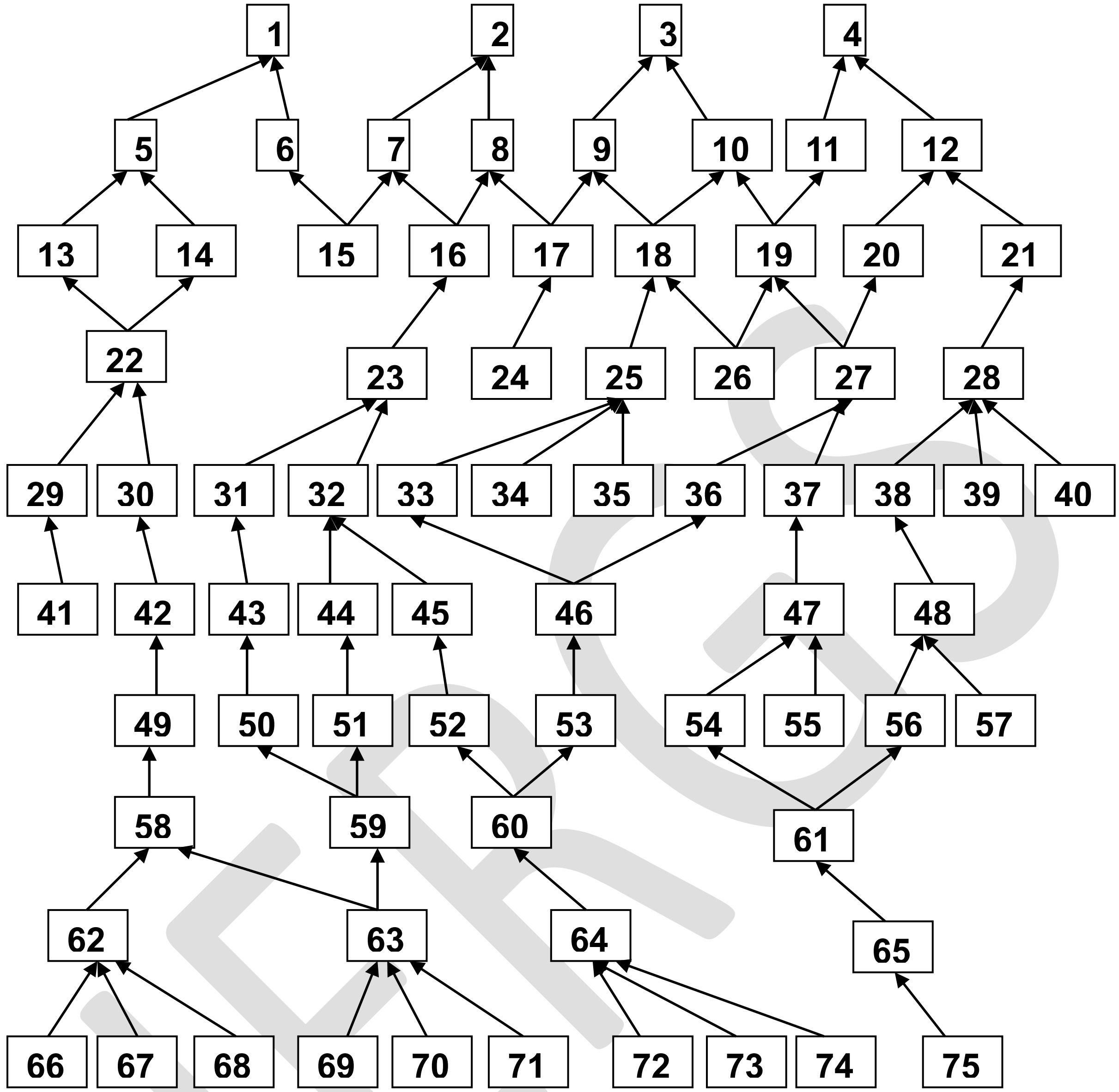

Figure 4c Product structures of 39×12 multi product problem

Table 1 Setup and Holding costs of different items in 50×12, 39×12,75×36 structures

S.No	50*12		39*12		75*36		S.No	50*12		39*12		75*36		S.No	75*36	
	H.C	S.C	H.C	S.C	H.C	S.C		H.C	S.C	H.C	S.C	H.C	S.C		H.C	S.C
1	97.83	780	40.08	490	50	410	26	7.53	540	1.45	580	30	580	51	18	800
2	45.19	200	35.27	450	49	450	27	4.36	160	3.63	650	31	620	52	17	410
3	43.82	590	59.66	90	50	430	28	18.52	480	4.35	450	30	610	53	16	350
4	5.82	710	25.42	140	48	420	29	5.81	410	3.29	820	30	490	54	15	320
5	26.04	890	10.42	880	47.2	250	30	1.93	140	5.04	620	30	300	55	14	280
6	18.87	610	22.64	440	46	300	31	6.71	390	2.53	580	29	200	56	13	280
7	27.03	920	22.31	70	42	500	32	15.35	370	3.3	340	29	200	57	12	180
8	15.64	210	19.53	430	42.5	800	33	4.36	520	0.61	340	25	100	58	11	680
9	2.67	490	1.34	930	40	400	34	3.28	700	2.52	80	25	120	59	10	190
10	1.86	920	25.12	650	40.5	500	35	6.38	160	4.83	690	25	300	60	9	100
11	23.5	520	9.46	740	37	200	36	3.47	290	3.44	430	27	400	61	8	480
12	12.59	540	17.48	680	36	330	37	1.97	420	0.91	60	27	200	62	7	200

13	25.13	510	4.32	800	45	480	38	1.76	160	2.64	760	25	800	63	6	270
14	16.42	500	14.28	220	40	450	39	6.41	450	2.65	180	25	100	64	5	600
15	0.84	300	2.56	850	37	380	40	7.17	340	-	-	25	250	65	4	210
16	1.02	450	10.07	400	40	200	41	2.97	750	-	-	27	450	66	3	700
17	0.62	440	4.59	650	36	100	42	0.25	140	-	-	28	100	67	3	100
18	23.71	510	7.13	860	35	100	43	3.22	430	-	-	26	200	68	3	200
19	15.32	910	8.82	850	35	120	44	1.85	890	-	-	25	800	69	3	100
20	20.58	830	10.6	670	34	280	45	3.84	610	-	-	26	100	70	3	150
21	8.71	730	6.02	370	33	270	46	0.41	860	-	-	24	500	71	3	200
22	3.14	850	2.78	360	35	290	47	0.37	860	-	-	24	480	72	2	100
23	0.94	450	2.95	310	35	320	48	3.84	350	-	-	22	250	73	2	200
24	13.02	370	9.32	440	33	380	49	3.95	610	-	-	21	600	74	2	100
25	7.34	390	0.31	590	30	560	50	1.63	350	-	-	19	100	75	1	100

H.C.=Holding Cost , S.C=Setup Cost

Table 2a Demand and Availability of end product in 50×12 problem

Period	1	2	3	4	5	6	7	8	9	10	11	12
Demand	15	5	15	110	65	165	125	25	90	15	140	115
Available	1000	2000	1000	0	5000	1000	0	500	800	500	1000	200

Table 2b Demand and Availability of end products in 39×12 problem

period	1	2	3	4	5	6	7	8	9	10	11	12
Item1	10	100	10	130	115	150	70	10	65	70	165	125
available	1500	2000	0	1000	800	5000	0	800	500	1000	2000	200
Item2	175	15	85	90	85	90	75	150	75	10	150	15
available	0	1000	2000	1000	900	0	800	1200	500	500	1000	100
Item3	135	165	15	105	25	120	50	60	5	140	60	10
available	1000	2000	900	800	0	1000	1200	300	500	800	100	100

Table 2c Demand and Availability of end products in 75×36 problem

period	1	2	3	4	5	6	7	8	9	10	11	12
Item1	10	100	10	10	70	10	20	10	10	50	10	70
available	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞
Item2	20	10	10	10	100	20	10	10	10	320	10	100
available	∞	∞	0	∞	5000	∞	∞	∞	∞	∞	∞	∞
Item3	30	10	10	100	10	10	20	10	40	100	10	10
available	∞	∞	∞	∞	∞	5000	∞	∞	∞	∞	∞	∞
Item4	40	10	10	30	10	10	10	10	100	10	10	120
available	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
period	13	14	15	16	17	18	19	20	21	22	23	24
Item1	10	100	10	60	10	10	50	10	10	10	30	10
available	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
Item2	10	20	10	170	10	10	50	10	10	10	210	10
available	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
Item3	10	180	10	10	10	10	60	10	10	10	10	10

available	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Item4	10	10	10	110	10	10	30	10	410	10	20	10
available	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
period	25	26	27	28	29	30	31	32	33	34	35	36
Item1	20	10	90	10	10	310	10	250	10	10	90	10
available	∞	∞	∞	∞	0	∞	∞	∞	∞	1000	∞	∞
Item2	10	10	10	1000	10	10	10	10	10	10	80	10
available	∞	∞	∞	∞	800	0	∞	∞	500	∞	∞	∞
Item3	600	10	100	10	10	10	10	10	600	10	10	10
available	∞	∞	∞	∞	0	∞	∞	∞	0	∞	∞	∞
Item4	50	10	10	10	800	10	10	10	90	10	10	10
available	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞

6. Results

All capacitated large size lot sizing problems are coded in c language and run on Intel® Core™ Duo processors 667 MHz Front Side Bus and 2M Smart L2 Cache with 2GB RAM.

The authors have solved all the test problems using BGA, IIBGA, and BPSO, IIBPSO, and results are compared among them. A lot sizing problem of 7 items and 6 periods which is taken from Jinxing Xie, Jiefang (2002), is also taken for the comparison.

Following tables 3, 5, 7 and figures 5, 6, 7 show the comparison of binary BGA, IIBGA, BPSO and IIBPSO algorithms at different iterations of different problems under consideration. Table 4,6,8,9 gives the information about the optimum values obtained for different test problems for different programming techniques. Table10 gives the percentage of improvement of solutions of BGA, BPSO, IIBPSO techniques when compared to BGA technique solution for different problems under consideration.

Table 3 comparison 50×12 problem results among BGA, IIBGA, BPSO and IIBPSO

50×12				
Iteration	**BGA**	**IIBGA**	**BPSO**	**IIBPSO**
5	386,785.09	380,765.30	280,295.00	250295.00
25	380,891.31	352114.59	243,797.00	241009.15
50	350,503.75	330138.87	203,956.09	200037.17
100	322,136.16	321142.15	193,128.11	199121.89
200	279,484.72	290477.29	192,017.59	195192.04
500	249,875.41	250132.65	189,013.95	185013.09
1,000	234,587.08	230513.19	186,579.11	182599.11
2,000	234,587.08	232187.12	186,543.84	183450.08
5,000	234,489.03	223154.89	185,042.16	174057.32
10,000	229,484.6	219803.29	184,629.19	173753.29
15,000	229,484.6	214040.12	181,685.31	173753.29
20,000	204,240.90	213108.00	181,685.31	173753.29
30,000	204,140.90	191617.40	181,685.31	173753.29

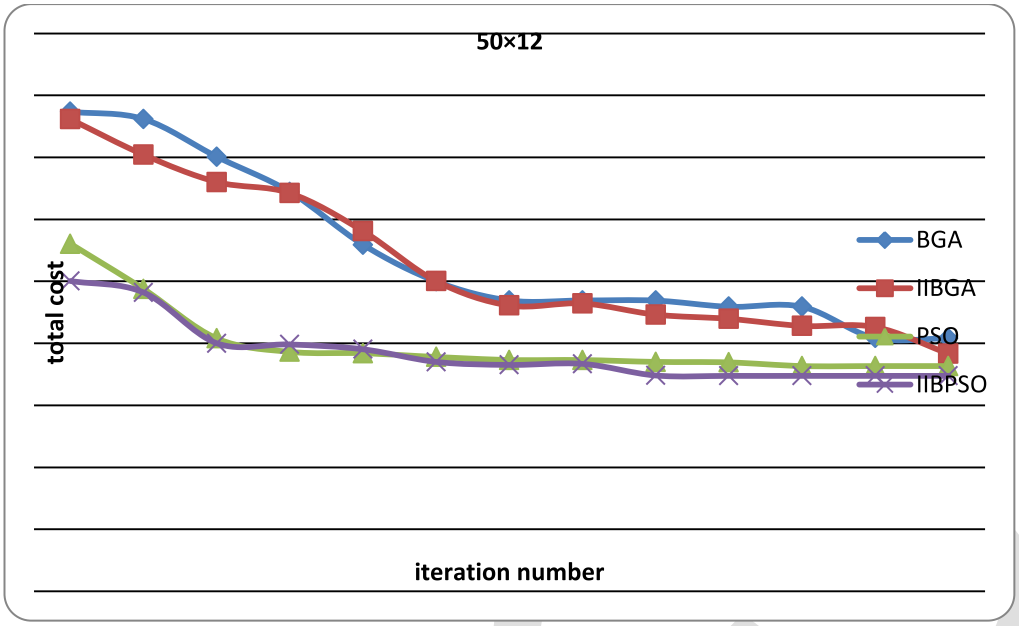

Figure. 5 BGA, IIBGA, BPSO, IIBPSO comparison at different iterations

Table 4 comparison 50×12 problem optimum results among BGA, IIBGA, BPSO and IIBPSO

50×12	BGA	IIBGA	BPSO	IIBPSO
	204,140.90	191617.40	181,685.31	173753.29

Table 5 comparison 39×12 problem results among BGA,IIBGA, BPSO and IIBPSO

39×12				
Iteration	**BGA**	**IIBGA**	**BPSO**	**IIBPSO**
5	377,421.19	350605.65	246,901.17	246,901.17
25	327,867.12	239426.79	217,583.65	213605.76
50	242,463.20	204744.77	204,084.98	197578.04
100	221,525.29	178650.31	202,884.17	191770.14
200	199,022.79	178346.06	194,724.84	191770.14
500	197,410.34	178244.00	193,219.70	186117.70
1,000	197,410.34	177609.65	185,691.15	142889.60
2,000	197,410.34	177609.65	185,691.15	142889.60
5,000	197,410.34	177609.65	172,684.78	142889.60
10,000	197,410.34	177609.65	172,682.56	142889.60
15,000	197,410.34	177609.65	172,682.56	142889.60

International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

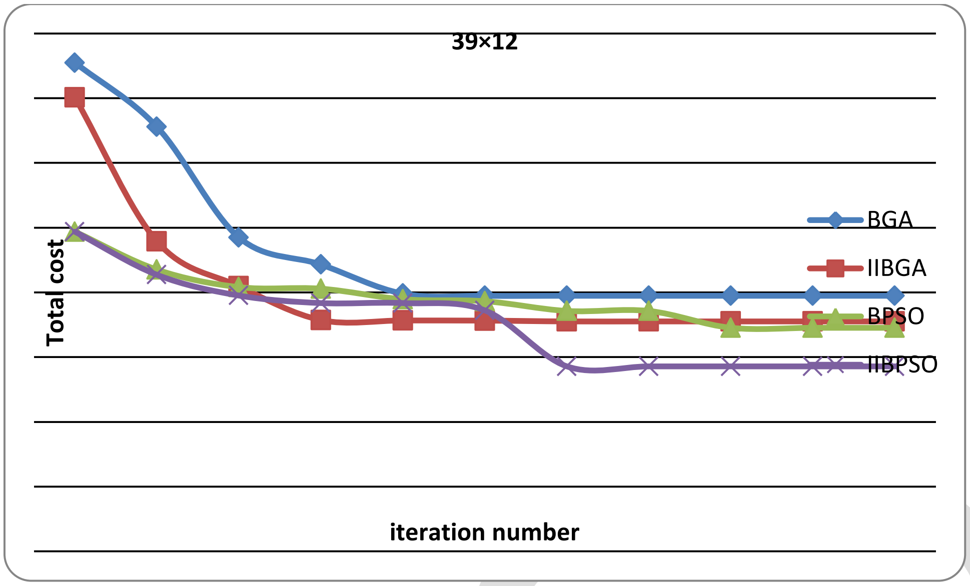

Figure. 6 BGA, IIBGA**,** BPSO, IIBPSO comparison at different iterations

Table 6 comparison 39×12 problem optimum results among BGA, IIBGA, BPSO and IIBPSO

39×12	BGA	IIBGA	BPSO	IIBPSO
	197,410.34	177609.65	172,682.56	142889.60

Table 7 comparison 75×36 problem results among BGA, IIBGA, BPSO and IIBPSO

75×36				
Iter No.	**BGA**	**IIBGA**	**BPSO**	**IIBPSO**
5	152174144	151074134	89866320	89866320
25	145240592	143150594	86317160	80226251
50	131999600	128899511	79341128	77312117
100	108485416	106374426	71873080	61752171
200	99614824	99919883	60409328	60409328
500	89866320	99614824	50344516	47817140
1,000	86317160	54844216	47819130	39071648
5,000	65511652	50344516	43816120	36459912
10,000	54344516	50344516	43816120	36291480
20,000	54344516	47444516	41817140	36205080
30,000	54344516	47344516	41817140	36205080

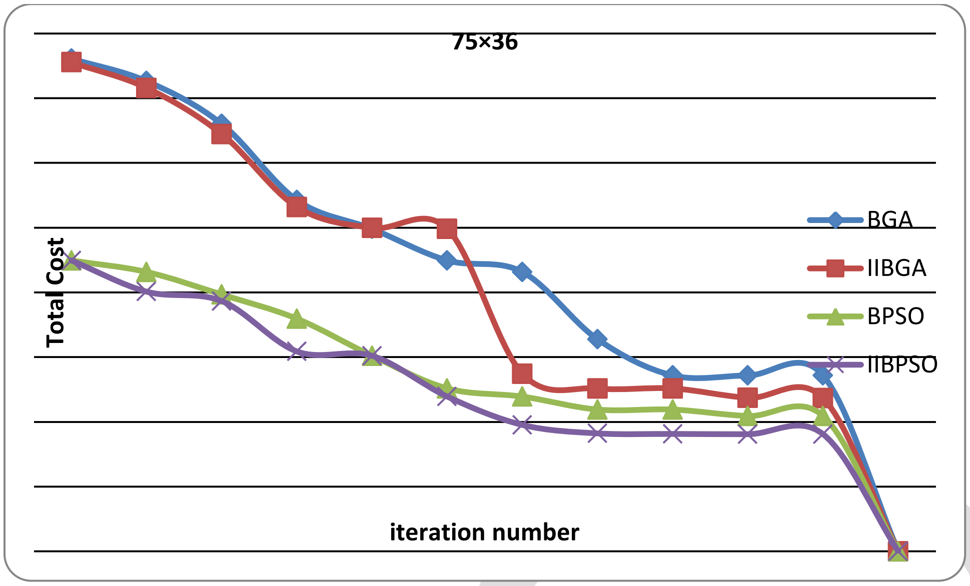

Figure 7 BGA, IIBGA, BPSO, IIBPSO comparison at different iterations

Table8 comparison 75×36 problem optimum results among BGA, IIBGA, BPSO and IIBPSO

	BGA	IIBGA	BPSO	IIBPSO
75×36	54344516	47344516	41817140	36205080

Table 9 comparisons 7×6, 50×12, 39×12, 75×36 problems optimum results among BGA, IIBGA, BPSO and IIBPSO

	BGA total cost	**IIBGA total cost**	**BPSO total cost**	**IIBPSO total cost**
7×6	9245	8320	8320	8320
50 × 12	204,140.90	191617.40	181,685.31	173753.29
39 × 12	197,410.34	177609.65	172,682.56	142889.60
75 × 36	54,344,516	47344516	41,817,140.0	36,205,080

Table 10 Percentage improvement in solution when compared to BGA Solution

	IIBGA	**BPSO**	**IIBPSO**
7×6	10	10	10
50 × 12	6.13	11	16
39 × 12	10	12.5	28
75 × 36	12.8	23.06	33.38

REFERENCES:

[1].N.P. Dellaert a, J. Jeunet b,*, Randomized multi-level lot-sizing heuristics for general product structures, European Journal of Operational Research 148 (2003) 211–228

[2]B.Karimi, S.M.T. Fatemi Ghomi, J.M.Wilson," The capacitated lot sizing problem: a review of models and algorithms", the international journal of Management science, Omega 31(2003) 365-378.

[3] Wagner, H. and Whitin, T. (2004). Dynamic version of the economic lot size model. Management Science, 50(12):1770–1774.

[4] Silver, E. and Meal, H. (1973). 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, 14(2):64–74.

[5] Coleman, B.J., McKnew, M.A., 1991. An improved heuristic for multilevel lot sizing in material requirements planning. Decision Sciences 22, 136–156.

[6] Hernández, W. and G. Süer, "Genetic Algorithms in Lot Sizing Decisions", Proceedings of the Congress on Evolutionary Computing (CEC99), Washington DC, July 6-9, 1999.

[7] N.Dellart, J.Jeunet, A genetic algorithm to solve the general multi-level lot sizing problem with time varying costs. International journal of production Economics68 (2000)241-257.

[8].Regina Berretta, Luiz Fernando Rodriguez, A memetic algorithm for a multistage capacitated lot sizing problem, Int.j. Production Economics 87(2004)67-81.

[9]M.Fatih Tasgetiren and Yun-Chia Liang(2003),A binary particle swarm optimization algorithm for lot sizing problem, journal of Economic and Social Research5(2),1-20.

[10]Yi Hana, Jiafu Tanga, Iko Kakub, Lifeng Mua, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight, Computers and Mathematics with Applications 57 (2009) 1748_1755

[11] Klorklear Wajanawichakon, Rapeepan Pitakaso, Solving large unconstrainted multi level lot-sizing problem by a binary particle swarm optimization. International Journal of Management Science and Engineering Management, 6(2): 134-141, 2011.

[12] JINXING XIE AND JIEFANG DONG, Heuristic Genetic Algorithm for General Capacitated Lot Sizing problem, Computers and Mathematics with Applications 44(2002) 263-276