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

V.V.D.Sahithi1, P.Sai Krishna2, K.Lalithkumar3, C.S.P.Rao4

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.com1 lalith.143143@gmail.com2 Saikrishna.parikirala@gmail.com3 csp_rao@rediffmail.com4

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 total cost (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 chosen 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. Taşgetiren 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, \( b_{it} \) 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 \( \in \) 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.

\[
\begin{align*}
\text{Min } (f(x)) &= \sum_{i=1}^{N} \sum_{t=1}^{T} \left( C_{it} x_{it} + h_{it} I_{it} + S_{it} y_{it} \right) \tag{1} \\
\sum_{j \in S(i)} r_{ij} x_{jt} - d_{it} &\leq I_{it-1} + X_{it} - I_{it} = d_{it} + \sum_{i=1}^{N} (V_{ikt} x_{it} + f_{ikt} y_{it}) \tag{2} \\
b_{kt} &\geq \sum_{i=1}^{N} (V_{ikt} x_{it} + f_{ikt} y_{it}) \tag{3} \\
X_{it} &\leq M y_{it} \quad i=1,\ldots,N; \quad t=1,\ldots,T \tag{4} \\
x_{it}, I_{it} &\geq 0 \quad i=1,\ldots,N; \quad t=1,\ldots,T \tag{5} \\
y_{it} &\in \{0,1\} \quad i=1\ldots N; \quad t=1,\ldots,T \tag{6}
\end{align*}
\]

www.ijergs.org
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 $m \times n$. where $m$ represents the number of items involved in the problem, represents time buckets. And particle representation is $X_{id}^{t}$. 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 $7 \times 6$ matrix.
As it is initial generation, all dimensions of particle are assigned to “0” or “1” randomly.

If \( R > 0.5 \) then \( X_{pt, id}^{t} = 1 \).
Else \( X_{pt, id}^{t} = 0 \).

Here \( R \) represents a random number.

\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

**Figure. 1** Particle dimension representation

\( X_{pt, id}^{t} \) 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 \( (X_{pt, id}^{t}) \).

(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

\[
V_{id}^{p0} = V_{\text{mini}} + (V_{\text{maxi}} - V_{\text{mini}}) * R
\]

here \([V_{\text{maxi}}, V_{\text{mini}}]=[-x,x] \), here \( x \) is an integer.

Ex: let \([V_{\text{maxi}}, V_{\text{mini}}]=[-5,5] \)

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

**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 particle containing best fitness value. As it is the initial generation all particle best \( (PB_{pt, id}^{t}) \) values are equal to particle values.

Here \( GB_{t, id}^{t} \) represents global best dimensions of \( t^{th} \) iteration.

Here \( PB_{pt, id}^{t} \) represents particle best dimensions of \( p^{th} \) particle \( t^{th} \) iteration.

(c) Updating parameters for next generations

(i) Updating velocity of particle \( (V_{pt, id}^{t}) \):

New velocity = \( V_{pt, id}^{t} = P (V_{pt, t-1}^{t-1} + \Delta V_{pt, t-1}^{t-1}) \)

\[
\Delta V_{pt, t-1}^{t-1} = c1 \ R1 (PB_{t, id}^{t-1} - X_{pt, id}^{t-1}) + c2 \ R2 (GB_{t, id}^{t-1} - X_{pt, id}^{t-1})
\]

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

Here Piece wise linear function \([P (V_{pt, id}^{t})]\)

\[
P (V_{pt, id}^{t}) = V_{\text{maxi}} \text{ if } V_{pt, id}^{t} > V_{\text{maxi}}
\]

\[
= V_{pt, id}^{t} \text{ if } |V_{pt, id}^{t}| < V_{\text{maxi}}
\]

\[
= V_{\text{mini}} \text{ if } V_{pt, id}^{t} < V_{\text{mini}}
\]

(ii) Updating position \( (X_{pt, id}^{t}) \) by sigmoid function:

\[
X_{pt, id}^{t} = 1 \text{ if } R < S (V_{pt, id}^{t})
\]

\[
= 0 \text{ otherwise}
\]

Sigmoid function \( S (V_{pt, id}^{t}) \):

This function forces velocity values to be in the limits of ‘0’ to ‘1’. It helps to update next generation \( X_{pt, id}^{t} \) values.

1095  

www.ijergs.org
S (V_{id}^{pt}) = \frac{1}{1+e^{-\beta_{id}^{pt}}}

(iii) Updating particle best and global best (PB_{id}^{pt}, GB_{id}^{pt})

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, otherwise go to (c) and repeat the procedure.

Pseudo code of IIBPSO is given in Figure 3.

STEP 1: Initialization phase
- Initialize swarm
- Assign velocities to all particles
- Fitness calculation
- Particle best and global best

STEP 2: 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
}

STEP 3: 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 $7 \times 6$

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

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

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

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

\[
\begin{align*}
\text{PB}_1 = &\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix} \\
\text{PB}_2 = &\begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix} \\
\text{PB}_3 = &\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix} \\
\text{Global Best} = \text{GB} = &\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\end{align*}
\]

**Step3:**

Update Velocity using standard procedure of Binary particle swarm optimization

\[
\begin{align*}
\text{Particle1} = &\begin{bmatrix}
+4.3 & -2.4 & -3.8 & +2.5 & +4.3 & +0.4 \\
-3.6 & -1.4 & -1.4 & -0.9 & -3.6 & +4.4 \\
-1.2 & +0.6 & +0.2 & -1.4 & -1.2 & +1.7 \\
-2.0 & +0.5 & -3.5 & -1.9 & -2.0 & -0.9 \\
+3.6 & -2.5 & -3.9 & -1.3 & +3.6 & +4.0 \\
+4.3 & +0.0 & +1.2 & -1.6 & +4.3 & -1.1 \\
+1.2 & -4.86 & +1.66 & -4.2 & +1.2 & -3.16
\end{bmatrix} \rightarrow \begin{bmatrix}
0.98 & 0.08 & 0.02 & 0.92 & 0.98 & 0.50 \\
0.02 & 0.19 & 0.19 & 0.28 & 0.02 & 0.98 \\
0.23 & 0.64 & 0.54 & 0.19 & 0.23 & 0.85 \\
0.11 & 0.62 & 0.02 & 0.13 & 0.11 & 0.28 \\
0.97 & 0.07 & 0.21 & 0.97 & 0.98 & 0.00 \\
0.98 & 0.50 & 0.76 & 0.16 & 0.98 & 0.23 \\
0.76 & 0.00 & 0.84 & 0.01 & 0.76 & 0.04
\end{bmatrix}
\end{align*}
\]

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

Update particle dimension matrix according new velocity matrix of particle.
As particle 1 fitness value is improved, so first particles, particle best (PB₁) 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:**

```
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
```

Input Particle = New Particle 1 = → Fitness = 10300

```
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
```

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

<table>
<thead>
<tr>
<th>S.No</th>
<th>50*12</th>
<th>39*12</th>
<th>75*36</th>
<th>50*12</th>
<th>39*12</th>
<th>75*36</th>
<th>S.No</th>
<th>75*36</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97.83</td>
<td>780</td>
<td>40.08</td>
<td>490</td>
<td>50</td>
<td>410</td>
<td>26</td>
<td>7.53</td>
</tr>
<tr>
<td>2</td>
<td>45.19</td>
<td>200</td>
<td>35.27</td>
<td>450</td>
<td>49</td>
<td>450</td>
<td>27</td>
<td>4.36</td>
</tr>
<tr>
<td>3</td>
<td>43.82</td>
<td>590</td>
<td>59.66</td>
<td>90</td>
<td>50</td>
<td>430</td>
<td>28</td>
<td>18.52</td>
</tr>
<tr>
<td>4</td>
<td>5.82</td>
<td>710</td>
<td>25.42</td>
<td>140</td>
<td>48</td>
<td>420</td>
<td>29</td>
<td>5.81</td>
</tr>
<tr>
<td>5</td>
<td>26.04</td>
<td>890</td>
<td>10.42</td>
<td>880</td>
<td>47.2</td>
<td>250</td>
<td>30</td>
<td>1.93</td>
</tr>
<tr>
<td>6</td>
<td>18.87</td>
<td>610</td>
<td>22.64</td>
<td>440</td>
<td>46</td>
<td>300</td>
<td>31</td>
<td>6.71</td>
</tr>
<tr>
<td>7</td>
<td>27.03</td>
<td>920</td>
<td>22.31</td>
<td>70</td>
<td>42</td>
<td>500</td>
<td>32</td>
<td>15.35</td>
</tr>
<tr>
<td>8</td>
<td>15.64</td>
<td>210</td>
<td>19.53</td>
<td>430</td>
<td>24.2</td>
<td>800</td>
<td>33</td>
<td>4.36</td>
</tr>
<tr>
<td>9</td>
<td>2.67</td>
<td>490</td>
<td>1.34</td>
<td>930</td>
<td>40</td>
<td>400</td>
<td>34</td>
<td>3.28</td>
</tr>
<tr>
<td>10</td>
<td>1.86</td>
<td>920</td>
<td>25.12</td>
<td>650</td>
<td>40.5</td>
<td>500</td>
<td>35</td>
<td>6.38</td>
</tr>
<tr>
<td>11</td>
<td>23.3</td>
<td>520</td>
<td>9.46</td>
<td>740</td>
<td>37</td>
<td>200</td>
<td>36</td>
<td>3.47</td>
</tr>
<tr>
<td>12</td>
<td>12.59</td>
<td>540</td>
<td>17.48</td>
<td>680</td>
<td>36</td>
<td>330</td>
<td>37</td>
<td>1.97</td>
</tr>
<tr>
<td>Item1</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>70</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Item2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Item3</td>
<td>30</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Item4</td>
<td>40</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Period</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>Item1</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>60</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Item2</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>170</td>
<td>10</td>
<td>10</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>Item3</td>
<td>10</td>
<td>180</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>60</td>
<td>10</td>
</tr>
</tbody>
</table>

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

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

Table 2c Demand and Availability of end products in 75×36 problem
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 10 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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>386,785.09</td>
<td>380,765.30</td>
<td>280,295.00</td>
<td>250,295.00</td>
</tr>
<tr>
<td>25</td>
<td>380,891.31</td>
<td>352,114.59</td>
<td>243,797.00</td>
<td>241,009.15</td>
</tr>
<tr>
<td>50</td>
<td>350,503.75</td>
<td>330,138.87</td>
<td>203,956.09</td>
<td>200,397.17</td>
</tr>
<tr>
<td>100</td>
<td>322,136.16</td>
<td>321,424.15</td>
<td>193,128.11</td>
<td>199,121.89</td>
</tr>
<tr>
<td>200</td>
<td>279,484.72</td>
<td>290,477.29</td>
<td>192,017.59</td>
<td>195,192.04</td>
</tr>
<tr>
<td>500</td>
<td>249,875.41</td>
<td>250,132.65</td>
<td>189,013.95</td>
<td>185,013.09</td>
</tr>
<tr>
<td>1,000</td>
<td>234,587.08</td>
<td>230,513.19</td>
<td>186,579.11</td>
<td>182,599.11</td>
</tr>
<tr>
<td>2,000</td>
<td>234,587.08</td>
<td>232,187.12</td>
<td>186,543.84</td>
<td>183,450.08</td>
</tr>
<tr>
<td>5,000</td>
<td>234,489.03</td>
<td>223,154.89</td>
<td>185,042.16</td>
<td>174,057.32</td>
</tr>
<tr>
<td>10,000</td>
<td>229,484.64</td>
<td>219,803.29</td>
<td>184,629.19</td>
<td>173,753.29</td>
</tr>
<tr>
<td>15,000</td>
<td>229,484.64</td>
<td>214,040.12</td>
<td>181,685.31</td>
<td>173,753.29</td>
</tr>
<tr>
<td>20,000</td>
<td>204,240.90</td>
<td>213,108.00</td>
<td>181,685.31</td>
<td>173,753.29</td>
</tr>
<tr>
<td>30,000</td>
<td>204,140.90</td>
<td>191,617.40</td>
<td>181,685.31</td>
<td>173,753.29</td>
</tr>
</tbody>
</table>
Figure. 5 BGA, IIBGA, BPSO, IIBPSO comparison at different iterations

Table 4 comparison 50x12 problem optimum results among BGA, IIBGA, BPSO and IIBPSO

<table>
<thead>
<tr>
<th>50x12</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x12</td>
<td>204,140.90</td>
<td>191,617.40</td>
<td>181,685.31</td>
<td>173,753.29</td>
</tr>
</tbody>
</table>

Table 5 comparison 39x12 problem results among BGA, IIBGA, BPSO and IIBPSO

<table>
<thead>
<tr>
<th>Iteration</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>39x12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>377,421.19</td>
<td>350,605.65</td>
<td>246,901.17</td>
<td>246,901.17</td>
</tr>
<tr>
<td>25</td>
<td>327,867.12</td>
<td>239,426.79</td>
<td>217,583.65</td>
<td>213,605.76</td>
</tr>
<tr>
<td>50</td>
<td>242,463.20</td>
<td>204,744.77</td>
<td>204,084.98</td>
<td>197,578.04</td>
</tr>
<tr>
<td>100</td>
<td>221,525.29</td>
<td>178,650.31</td>
<td>202,884.17</td>
<td>191,770.14</td>
</tr>
<tr>
<td>200</td>
<td>199,022.79</td>
<td>178,346.06</td>
<td>194,724.84</td>
<td>191,770.14</td>
</tr>
<tr>
<td>500</td>
<td>197,410.34</td>
<td>178,244.00</td>
<td>193,219.70</td>
<td>186,117.70</td>
</tr>
<tr>
<td>1,000</td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>185,691.15</td>
<td>142,889.60</td>
</tr>
<tr>
<td>2,000</td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>185,691.15</td>
<td>142,889.60</td>
</tr>
<tr>
<td>5,000</td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>172,684.78</td>
<td>142,889.60</td>
</tr>
<tr>
<td>10,000</td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>172,682.56</td>
<td>142,889.60</td>
</tr>
<tr>
<td>15,000</td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>172,682.56</td>
<td>142,889.60</td>
</tr>
</tbody>
</table>
Figure. 6 BGA, IIBGA, BPSO, IIBPSO comparison at different iterations

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

<table>
<thead>
<tr>
<th>39×12</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>197,410.34</td>
<td>177,609.65</td>
<td>172,682.56</td>
<td>142,889.60</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Iter No.</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>152,174,44</td>
<td>151,074,134</td>
<td>89,866,320</td>
<td>89,866,320</td>
</tr>
<tr>
<td>25</td>
<td>145,240,592</td>
<td>143,150,594</td>
<td>86,317,160</td>
<td>80,226,251</td>
</tr>
<tr>
<td>50</td>
<td>131,999,600</td>
<td>128,899,511</td>
<td>79,341,128</td>
<td>77,312,117</td>
</tr>
<tr>
<td>100</td>
<td>108,485,416</td>
<td>106,374,426</td>
<td>71,873,080</td>
<td>61,752,171</td>
</tr>
<tr>
<td>200</td>
<td>99,614,824</td>
<td>99,919,883</td>
<td>60,409,328</td>
<td>60,409,328</td>
</tr>
<tr>
<td>500</td>
<td>89,866,320</td>
<td>99,614,824</td>
<td>50,344,516</td>
<td>47,817,140</td>
</tr>
<tr>
<td>1,000</td>
<td>86,317,160</td>
<td>54,844,216</td>
<td>47,819,130</td>
<td>39,071,648</td>
</tr>
<tr>
<td>5,000</td>
<td>65,511,652</td>
<td>50,344,516</td>
<td>43,816,120</td>
<td>36,459,912</td>
</tr>
<tr>
<td>10,000</td>
<td>54,344,516</td>
<td>50,344,516</td>
<td>43,816,120</td>
<td>36,291,480</td>
</tr>
<tr>
<td>20,000</td>
<td>54,344,516</td>
<td>47,444,516</td>
<td>41,817,140</td>
<td>36,205,080</td>
</tr>
<tr>
<td>30,000</td>
<td>54,344,516</td>
<td>47,344,516</td>
<td>41,817,140</td>
<td>36,205,080</td>
</tr>
</tbody>
</table>
Figure 7 BGA, IIBGA, BPSO, IIBPSO comparison at different iterations

Table 8 comparison 75×36 problem optimum results among BGA, IIBGA, BPSO and IIBPSO

<table>
<thead>
<tr>
<th>75x36</th>
<th>BGA</th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>54344516</td>
<td>47344516</td>
<td>41817140</td>
<td>36205080</td>
</tr>
</tbody>
</table>

Table 9 comparisons 7x6, 50×12, 39×12, 75×36 problems optimum results among BGA, IIBGA, BPSO and IIBPSO

<table>
<thead>
<tr>
<th></th>
<th>BGA total cost</th>
<th>IIBGA total cost</th>
<th>BPSO total cost</th>
<th>IIBPSO total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>7x6</td>
<td>9245</td>
<td>8320</td>
<td>8320</td>
<td>8320</td>
</tr>
<tr>
<td>50x12</td>
<td>204,140.90</td>
<td>191617.40</td>
<td>181,685.31</td>
<td>173753.29</td>
</tr>
<tr>
<td>39x12</td>
<td>197,410.34</td>
<td>177609.65</td>
<td>172,682.56</td>
<td>142889.60</td>
</tr>
<tr>
<td>75x36</td>
<td>54,344,516</td>
<td>47344516</td>
<td>41,817,140.0</td>
<td>36,205,080</td>
</tr>
</tbody>
</table>

Table 10 Percentage improvement in solution when compared to BGA Solution

<table>
<thead>
<tr>
<th></th>
<th>IIBGA</th>
<th>BPSO</th>
<th>IIBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7x6</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>50x12</td>
<td>6.13</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>39x12</td>
<td>10</td>
<td>12.5</td>
<td>28</td>
</tr>
<tr>
<td>75x36</td>
<td>12.8</td>
<td>23.06</td>
<td>33.38</td>
</tr>
</tbody>
</table>

REFERENCES:
