INTEGRATION OF INVENTORY CONTROL AND SCHEDULING USING BINARY PARTICLE SWARM OPTIMIZATION ALGORITHM

Manash Dey
Assistant Professor, Mechanical Engineering Department, JIMS EMTC Greater Noida (India)

ABSTRACT

Production planning and scheduling are usually performed in a hierarchical manner, thus generating unfeasibility conflicts when comes to implementation. Moreover, solving these problems simultaneously in complex manufacturing systems is very challenging in production management. Production planning is first performed at the tactical decision level and, the different jobs are then supposed to be scheduled at the operational decision level. Therefore, the information about capacity planned at the tactical level is in aggregate manner, thus not guaranteeing that scheduling constraints are respected. Thereby, the production plans may be unfeasible.

An integrated approach for guaranteeing consistency to some extent between decisions taken at tactical and operational levels of production management was presented, thus avoiding the shortcomings of traditional approaches in which decisions are taken sequentially. Integrated problem are solved by using the exact capacity constraint from a standard scheduling problem to the lot sizing problem.

However this combinatorial optimization problem can be solved by using soft computing techniques in reasonable time. In the present work we have applied Binary Particle Swarm optimization (BPSO) technique to the Single item single level, multi-level and Multi item Lot sizing problems with and without applying the Scheduling constraint. We have tested the BPSO technique to the different types and sizes of problems by applying scheduling constraint. The obtained results are compared with Lot sizing problems without constraint and it is concluded that in all instances the results are improved compared to simple lot sizing problems.

Keywords: MRP, Binary Particle Swarm optimization (BPSO)

I. INTRODUCTION

Today’s business environment has become highly competitive. Manufacturing firms have started recognizing the importance of manufacturing strategy in their businesses. Firms are increasingly facing external pressures to improve customer response time, increase product offerings, manage demand variability and be price competitive. In order to meet these challenges, firms often find themselves in situations with critical shortages of some products and
excess inventories of other products. This raises the issue of finding the right balance between cutting costs and maintaining customer responsiveness. Previously, production specialists used multiple and sometimes contradictory or confusing databases, data gathered from machine operators, and past experience to gauge what was needed to meet production goals. Problems always take place on shop floor when generating MRP and production schedule are separately taken into account since both MRP and schedule aim for different objectives which are not synchronized. MRP is computer software based production planning and inventory control system used to ensure that all materials required are ready for production and requested products are available for delivery to customers with the lowest possible level of inventory. Using conventional MRP and classic shop floor scheduling separately cannot solve the problem. Integration of inventory control and scheduling is one of the solutions.

II. MATHEMATICAL FORMULA

1. Mathematical formulation to the Single level Lot sizing Problem (SISL)

The incapacitated single item no shortages allowed and single level lot sizing model is the simplest model in the inventory lot sizing problems. Lot sizing formulation for this kind of lot sizing problem takes the following form

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

subject to:

\[I_0 = 0 \quad \forall i\]

\[I_{i-1} + x_iQ_i - I_i = R_i \quad \forall i\]

\[I_i \geq 0 \quad \forall i\]

\[Q_i \geq 0 \quad \forall i\]

\[x_i \in \{0,1\} \quad \forall i\]

Where

n=number of periods, A=ordering/setup cost per period, c=holding cost per unit per period, Ri=net requirement for period i, Qi=Order quantity for period i, Ii=projected inventory balance for period i, Xi=1 if an order is placed in period i, Xi=0 otherwise.
2. Mathematical formulation to the Multi level Lot sizing Problem (MLLS)

\[
\min \sum_{i \in \Gamma(i)} \sum_{t=1}^{T}(s_i Y_{it} + h_i I_{it}) \rightarrow (1)
\]

\[
I_{it} = I_{i,t-1} + x_{it} - d_{it} \rightarrow (2)
\]

\[
d_{it} = \sum_{j \in \Gamma^{-1}(i)} c_{ij} x_{jt} \rightarrow (3)
\]

\[
x_{it} - M \gamma_{it} \leq 0, \gamma_{it} \in \{0, 1\} \rightarrow (4)
\]

\[
I_{it} \geq 0, x_{it} \geq 0 \rightarrow (5)
\]

Necessary notations:
\(\Gamma (i)\): set of immediate successors of items \(i\); \(\Gamma^{-1}(i)\): set of immediate predecessors of items \(i\); \(c_{ij}\): quantity of item \(i\) required to produce one unit of items \(j\); \(D_{it}\): external requirement for items \(i\) in period \(t\); \(h_i\): holding cost for items \(i\) (Following small instance standard); \(I_{i,0}\): initial inventory of product \(i\); \(S_{i}\): setup cost for items \(i\) (Following small instance standard); \(T\): total number of periods.

Decision and auxiliary variables:
\(d_{it}\): total requirement for item \(i\) in period \(t\); \(I_{it}\): Inventory level of item \(i\) at the end of period \(t\); \(X_{it}\): delivered quantity of items \(i\) at the beginning of the period \(t\); \(Y_{it}\): binary variable which indicates if an item \(i\) is produced in period \(t\), \((y_{it} = 1)\) or not \((y_{it} = 0)\).

3. Integrated formulation of planning and scheduling

The problem is formulated as
\[
\min \sum_{i \in \Gamma} \sum_{t=1}^{T} (c_{it} \cdot I_{it} + c_{it} \cdot I_{it} + c_{it} \cdot X_{it})
\]

\[
(I_{it} - I_{it}^-) - (I_{i,t-1}^- - I_{it-1}^-) - X_{it} + D_{it} = 0, i=1,...,n; t=1,...,T \rightarrow (1)
\]

\[
X_{it} \geq 0, \forall i, t \rightarrow (2)
\]

\[
I_{it}^+ \geq 0, \forall i, t \rightarrow (3)
\]

\[
I_{it}^- \geq 0, \forall i, t \rightarrow (4)
\]

\[
t_{ijk} - t_{ijk} - p_{ij}^w X_{it} \geq 0, \forall (o_{ij}, o_{ijk}) \in A \rightarrow (5)
\]

\[
t_{ijk} \geq 0, \forall o_{ijk} \in N \rightarrow (6)
\]

\[
t_{ijk} - t_{ijk'} - p_{ij}^w X_{ijk'} \geq 0, \forall (o_{ijk}, o_{ijk'}) \in S(y) \rightarrow (7)
\]

\[
t_{ijk} + p_{ijk}^w X_{it} \leq \sum_{j=1}^{J} c_j \forall o_{ijk} \in L \rightarrow (8)
\]

\[
t_{ijk} + p_{ijk}^w X_{it} \geq \sum_{j=1}^{J} c_j \forall o_{ijk} \in L \rightarrow (9)
\]
The objective function in the above problem is the minimization of sum of the Inventory surplus, backlog, and production cost of the products to be planned. (1) is the standard inventory balance equation. Constraints 2, 3, 4 presents that production items, inventory surplus, backlog quantities are always positive. Constraint 5 gives the conjunctive constraint relationship among the operation on the machines. (6) Gives that starting times of operation Oijt are always positive. Constraints 7 give disjunctive constraints relations among the operations. Constraints 8 & 9 state that the last operations of the Jit must be completed in period t and not before. Constraint 7 replaced with necessary conditions which does not involve Disjunctive constraints.

\[ \sum_{l=1}^{t} \left( \sum_{\epsilon_{il} \in O_{lt}} P_{ijkl} X_{il} \right) \leq \sum_{l=1}^{t} C_{l} \]

III. IMPLEMENTATION OF BPSO TO INTEGRATED PROBLEM

1. Binary Particle Swarm Optimization Algorithm (BPSO)

Pseudo code of the general PSO is given as

Begin
Step 1: Initialization
  - Initialize swarm, including swarm size, each particle’s position and velocity;
  - Evaluate the each particle fitness;
  - Initialize gbestposition with particle with the lowest fitness in the swarm;
  - Initialize pbest position with a copy of particle itself;
  - Give initial value: \( W_{\text{max}}, W_{\text{min}}, C_{1}, C_{2}\) and generation=0;
Step 2: Computation
  While (the maximum of generation is not met)
    Do {
      Generation++;
      Generate next swarm by equation (1a) and (1b);
      Evaluate Swarm {
        Find new gbest and pbest;
        Update gbest of the swarm and pbest of each particle;
      }
    }
Step 3: Output optimization results
End

The basic elements of PSO algorithm is summarized as follows:

**Particle:** 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 \( X_{ik} = [X_{i1}, X_{i2}, X_{i3}, \ldots, X_{id}] \), where \( x \)'s are the optimized parameters and \( X_{id} \) 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:** \( \text{pop}^{k} \) is the set of \( n \) particles in the swarm at iteration \( k \), i.e., \( \text{pop}^{k} = [X_{1}^{k}, X_{2}^{k}, X_{3}^{k}, \ldots, X_{n}^{k}] \).
Particle velocity: $V_{ik}^k$ is the velocity of particle $i$ at iteration $k$. It can be described as $V_{ik}^k = [V_{i1}^k, V_{i2}^k, V_{i3}^k, \ldots, V_{id}^k]$, where $V_{id}^k$ is the velocity with respect to $d^{th}$ dimension.

Particle best: $PB_{ik}^k$ 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 $PB_{ik}^k = [pb_{i1}^k, pb_{i2}^k, pb_{i3}^k, \ldots, pb_{id}^k]$, with the fitness function $f(PB_{ik}^k)$.

Global best: $GB_{ik}^k$ is the best position among all particles in the swarm, which is achieved so far and can be expressed as $GB_{ik}^k = [gb_{i1}^k, gb_{i2}^k, gb_{i3}^k, \ldots, gb_{id}^k]$, with the fitness function $f(GB_{ik}^k)$.

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.

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

**Step 1: Initialization**

a) Set $k=0, n=\text{twice the number of dimensions}$
b) Generate $n$ particles randomly as, $\{X_{i0}^0, i=0,1,2,\ldots,n\}$, where $X_{i0}^0 = [X_{i1}^0, X_{i2}^0, X_{i3}^0, \ldots, X_{id}^0]$.
c) Generate the initial velocities of all particles randomly, $\{V_{i0}^0, i=0,1,2,\ldots,n\}$, where $V_{i0}^0 = [v_{i1}^0, v_{i2}^0, v_{i3}^0, \ldots, v_{id}^0]$. $v_{id}^0$ is randomly generated with $v_{id}^0 = V_{min}^c + (V_{max}^c - V_{min}^c) \cdot \text{rand}()$.
d) Evaluate each particle in the swarm using the objective function, $f(X_{i0}^0)$.
e) For each particle $i$ in the swarm, set $PB_{i0}^0 = X_{i0}^0$, where $PB_{i0}^0 = [pb_{i1}^0, pb_{i2}^0, pb_{i3}^0, \ldots, pb_{id}^0] = X_{i0}^0$ along with its best fitness value, $f_{i\text{pbest}}(PB_{i0}^0, i=1,2,3,\ldots,n)$.
f) Set the global best to, $f_{i\text{gbest}}(GB_0^0) = \text{min}\{ f_{i\text{pbest}}(PB_{i0}^0, i=1,2,3,\ldots,n) \}$ with $GB_0^0 = [gb_1, gb_2,\ldots, gb_d]$.

**Step 2: Update iteration counter**

- $k=k+1$

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

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

- $C_1$ and $C_2$ are social and cognitive parameters and $r_1$ and $r_2$ uniform random numbers between $(0, 1)$.

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

- $X_{id}^k = \{1, \text{if } U(0,1) < \text{sigmoid}(v_{id}^k) \}$
- $0$, otherwise

**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,

If \( f_i^k(X_i^k, i=0,1,2,\ldots,n) < f_i^{pbest}(PB_i^{k-1}, i=0,1,2,\ldots,n) \)

then

\[ f_i^{pbest}(PB_i^k, i=0,1,2,\ldots,n) = f_i^k(X_i^k, i=0,1,2,\ldots,n) \]

else

\[ f_i^{pbest}(PB_i^k, i=0,1,2,\ldots,n) = f_i^{pbest}(PB_i^{k-1}, i=0,1,2,\ldots,n) \]

Step 6: Update global best

\[ f^{gbest}(GB^k) = \min \{ f_i^{pbest}(PB_i^k, i=1,2,\ldots,n) \} \]

If \( f^{gbest}(GB^k) < f^{gbest}(GB^{k-1}) \), then

\[ f^{gbest}(GB^k) = f^{gbest}(GB^k) \]

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.

IV. RESULT

1. Single item Multi level Problem

![Figure 5.3 BOM Structure of 7×6 problem](image)

Figure 5.3 BOM Structure of 7×6 problem
Table 1.1 Demand of products and cost involved in (7×6) problem

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 76</td>
<td>25</td>
<td>20</td>
<td>15</td>
<td>10</td>
<td>20</td>
<td>35</td>
</tr>
<tr>
<td>Item no</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>S.C</td>
<td>400</td>
<td>500</td>
<td>1000</td>
<td>300</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>H.C</td>
<td>12</td>
<td>0.6</td>
<td>1</td>
<td>0.04</td>
<td>0.03</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 1.2 Demand of products for three different problems

<table>
<thead>
<tr>
<th>period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand 25×12</td>
<td>15</td>
<td>5</td>
<td>15</td>
<td>110</td>
<td>65</td>
<td>165</td>
<td>125</td>
<td>25</td>
<td>90</td>
<td>15</td>
<td>140</td>
<td>115</td>
</tr>
<tr>
<td>Demand 40×12</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td>130</td>
<td>115</td>
<td>150</td>
<td>70</td>
<td>10</td>
<td>65</td>
<td>70</td>
<td>165</td>
<td>25</td>
</tr>
<tr>
<td>Demand 50×12</td>
<td>15</td>
<td>5</td>
<td>15</td>
<td>120</td>
<td>65</td>
<td>155</td>
<td>125</td>
<td>25</td>
<td>95</td>
<td>15</td>
<td>135</td>
<td>115</td>
</tr>
</tbody>
</table>

Comparison of results with and without scheduling constraint tested at different iterations

Table 1.3 SIML problem solution with four different sizes at different iterations

<table>
<thead>
<tr>
<th>iteration No</th>
<th>simple 50×12</th>
<th>Integrated 50×12</th>
<th>simple 40×12</th>
<th>Integrated 40×12</th>
<th>simple 25×12</th>
<th>Integrated 25×12</th>
<th>Simple 7×6</th>
<th>Integrated 7×6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>256564.83</td>
<td>240840</td>
<td>367011.844</td>
<td>366131.843</td>
<td>306744.94</td>
<td>299342.5</td>
<td>5235.25</td>
<td>5190</td>
</tr>
<tr>
<td>25</td>
<td>244064.84</td>
<td>220425</td>
<td>333017.781</td>
<td>332167.781</td>
<td>260425.02</td>
<td>229615</td>
<td>4323.5</td>
<td>4285.27</td>
</tr>
<tr>
<td>50</td>
<td>221292.95</td>
<td>214097.5</td>
<td>276022.781</td>
<td>274832.781</td>
<td>217487.42</td>
<td>187560</td>
<td>3500</td>
<td>3465.15</td>
</tr>
<tr>
<td>100</td>
<td>208054.88</td>
<td>208940</td>
<td>251291.906</td>
<td>250011.906</td>
<td>193371.08</td>
<td>166397.5</td>
<td>2965.32</td>
<td>2846.58</td>
</tr>
<tr>
<td>500</td>
<td>204272.45</td>
<td>197120</td>
<td>233231.188</td>
<td>231861.1875</td>
<td>166823.33</td>
<td>158520</td>
<td>2833.63</td>
<td>2742.88</td>
</tr>
<tr>
<td>1000</td>
<td>200755.48</td>
<td>196262.5</td>
<td>221354.688</td>
<td>220344.6875</td>
<td>162807.89</td>
<td>152022.5</td>
<td>2795.34</td>
<td>2731.85</td>
</tr>
<tr>
<td>2000</td>
<td>194767.62</td>
<td>193652</td>
<td>218257.954</td>
<td>217965.7056</td>
<td>159652</td>
<td>150712.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>193546.38</td>
<td>192758</td>
<td>214125.36</td>
<td>213924.1637</td>
<td>159242.85</td>
<td>150215.32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 1.1 Convergence of Four SIML problems solutions at different iterations

2. Multi item level Problem

Table 1.4 MIML problem solution with three different sizes at different iterations

<table>
<thead>
<tr>
<th>iteration no</th>
<th>Simple 39x12</th>
<th>Integrated 39x12</th>
<th>Simple 15x12</th>
<th>Integrated 15x12</th>
<th>Simple 25x12</th>
<th>Integrated 25x12</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>422388.1563</td>
<td>412894.2812</td>
<td>267876.5625</td>
<td>252053.1563</td>
<td>350988.8125</td>
<td>336092.2813</td>
</tr>
<tr>
<td>25</td>
<td>376876.25</td>
<td>365135.0937</td>
<td>204949.6563</td>
<td>194649.6563</td>
<td>293958.2188</td>
<td>283658.2188</td>
</tr>
<tr>
<td>50</td>
<td>340013.4688</td>
<td>328273.875</td>
<td>163244.7031</td>
<td>156844.7031</td>
<td>236183.5313</td>
<td>222883.5313</td>
</tr>
<tr>
<td>100</td>
<td>280418.8438</td>
<td>267891.5937</td>
<td>124069.1641</td>
<td>123669.1641</td>
<td>194280.3594</td>
<td>189580.3594</td>
</tr>
<tr>
<td>200</td>
<td>255216.375</td>
<td>244539.421</td>
<td>121145.0547</td>
<td>120225.8984</td>
<td>181030.5625</td>
<td>175430.5625</td>
</tr>
<tr>
<td>500</td>
<td>244271.6406</td>
<td>231291.8906</td>
<td>113352.3047</td>
<td>112148.213</td>
<td>170112.3906</td>
<td>166256.7813</td>
</tr>
<tr>
<td>1000</td>
<td>230098.312</td>
<td>223192.4062</td>
<td>103500.2031</td>
<td>101742.2361</td>
<td>156994.7188</td>
<td>154257.6875</td>
</tr>
<tr>
<td>2000</td>
<td>222510</td>
<td>213361.4687</td>
<td>100472.1862</td>
<td>94232.80469</td>
<td>155365.7031</td>
<td>150367.7969</td>
</tr>
</tbody>
</table>

Figure 5.8 Convergence of three MIML problems solutions at different iterations
Figure 5.9 Comparison of three MIML problems solutions at different iterations

Figure 5.6 Convergence of three SIML problems solutions at different Swarm sizes

Figure 5.10 Convergence of three MIML problems solutions at different Swarm sizes
V. CONCLUSIONS

To the best of knowledge no work related to the integration problem by using BPSO technique has been published so far in the contemporary literature. BPSO technique have been successfully applied to integrated model and tested for different lot sizing problems such as single item single level, single item multi-level and multi item problems with three different product structures. In all the problem instances we found the improvement in inventory cost by introducing the scheduling constraint in the lot sizing problems. We found that problem solutions are converging at higher number of iterations and Swarm sizes.

Computational experience of BPSO algorithm to the combinatorial optimization problems in manufacturing decision making problems is good and its implementation to manufacturing problems is easy as it is having few number of control parameters in algorithms compared to other evolutionary algorithms.

REFERENCES


