A genetic algorithm approach for integrated production and distribution problem

A. J. Abraham\textsuperscript{a}, K. Ratna Kumar\textsuperscript{b}, R. Sridharan\textsuperscript{c}, D. Singh\textsuperscript{a}

\textsuperscript{a}Post Graduate Student, National Institute of Technology Calicut, Calicut 673601, India
\textsuperscript{b}Assistant Professor, National Institute of Technology, Calicut 673601, India
\textsuperscript{c}Professor, Department of Mechanical Engineering, National Institute of Technology Calicut – 673601, Kerala, India.

Abstract

In this paper we propose a mathematical model for a production-distribution problem with multiple plant and multiple products over a planning horizon with multiple time periods. The environment allows both stock out and backordering. In order to solve this problem we introduce a genetic algorithm approach.

Keywords: Supply chain management, genetic algorithm; stockout and backordering; production and distribution planning; plant side restriction.

1. Introduction

Supply chain management as a key activity that can play the part of deciding factor that dictates profit or loss in the modern business environment. Though supply chain is a key concept it is interpreted and defined by various authors in different ways. But all of these definitions define supply chain as an integrated effort to fulfill customer requirements in terms of quality and quantity. A supply chain in the simplest sense is just a buyer seller interaction involving various stages in between. Unless the supply chain performs in an integrated and coordinated fashion, all members of the supply chain will suffer loss. Therefore integrating various stages of the supply chain is a widely investigated topic in supply chain analytics. It has been pointed out that integrating the retailer to its previous stage can result in reduction of inventory holding cost. Logistics cost, which is a major component of the operating costs can also be decreased if we consider an integrated production distribution plan. Supply chain decisions can be made in an independent fashion or in an integrated fashion.

*corresponding Author: Email Address: ratna_kumar@nitc.ac.in
**Nomenclature**

\( i \) Plants, \( i \in \{1, 2, \ldots, I\} \)

\( j \) Retail outlets, \( j \in \{1, 2, \ldots, J\} \)

\( k \) Product items, \( k \in \{1, 2, \ldots, K\} \)

\( t \) Time periods, \( t \in \{1, 2, \ldots, T\} \)

\( A_P^{ikt} \) Ending on-hand inventory of item \( k \) at plant \( i \) in period \( t \).

\( A_P^{ik} \) Per period unit holding cost of item \( k \) at plant \( i \).

\( A_R^{jkt} \) Ending on-hand inventory of item \( k \) at retail outlet \( j \) in period \( t \).

\( A_R^{jk} \) Per period unit holding cost at retail outlet \( j \) for item \( k \).

\( B^{CAP} \) Vehicle available capacity

\( B_{jk}^{kt} \) Backorder quantity of item \( k \) at retail outlet \( j \) in period \( t \).

\( BC_{jk} \) Unit backordering cost of item \( k \) at retail outlet \( j \).

\( C_{ik} \) Unit processing cost of item \( k \) at plant \( i \).

\( D_{ij} \) Unit cost of transportation from plant \( i \) to retail outlet \( j \).

\( F_{jkt} \) Demand for item \( k \) at retail outlet \( j \) in period \( t \).

\( G \) Fixed cost per vehicle

\( L_i \) Available production capacity at plant \( i \).

\( O_{ik} \) Unit processing time of item \( k \) at plant \( i \).

\( P_{jk} \) Unit selling price of item \( k \) at retail outlet \( j \).

\( Q_{jkt}^{ik} \) Amount of item \( k \) delivered from plant \( i \) to retail outlet \( j \) in period \( t \).

\( S_{ik} \) Set-up cost for item \( k \) at plant \( i \).

\( SO_{jkt}^{ik} \) Stockout quantity of item \( k \) at retail outlet \( j \) in period \( t \).

\( ST_{i} \) Storage capacity at plant \( i \).

\( U_{ik} \) Set-up time for item \( k \) at plant \( i \).

\( V_{jk} \) Unit stockout cost for item \( k \) at retail outlet \( j \).

\( W_j \) Storage capacity at retailer \( j \).

\( x_{ikt} \) Amount of item \( k \) produced at plant \( i \) in period \( t \).

\( y_{ikt} \) Takes value 1 when product \( k \) is set-up at plant \( i \) in period \( t \).

\( Z_{jk}^{it} \) Number of vehicles required for delivering items from plant \( i \) to retail outlet \( j \) in period \( t \).

\( \tau_{jk} \) Critical time duration for product \( k \) at retailer \( j \).

**1.1. Literature review**

Researchers have investigated the effect of partial integration and found that integrating logistics with inventory control decisions can lead to major savings in logistics cost. Logistics cost accounts for nearly half the supply chain cost and an integrated approach can reduce this cost (Chandra and Fisher, 1994). They also said that total operating cost which includes inventory costs also can be reduce up to 30%. Though they introduced production restriction many realistic aspects like backordering, stock out, inventory restriction and setup time were not considered and analyzed in detail. Sarmiento and Nagi (1999) gives a good review of the works in this area. A critical review of the works in integrated production distribution planning is given by Erenguc et al. (2001).

Park (2005) investigated the effectiveness of the integrated approach over the decoupled one through an extensive computational study using heuristic for the decoupled production and distribution problem. Only stock outs were considered in this work and plant side restrictions were not considered. Genetic algorithm has been used to solve integrated production and distribution problems as in Abdelmaguid and Dessouky (2006). Moin et al. (2011) also used a hybrid genetic algorithm approach to solve a multi-product, multi-period inventory routing problem.

The content of this paper is divided into four sections. Section (2) has the problem definition, followed by mathematical model (3), solution procedure (4) and computational results (5).
2. Problem statement

This integrated production distribution problem describes a two echelon supply chain with multiple plants each capable of producing multiple products serving retailer demands over a given number of planning periods. Storage of materials is allowed at retailers as well as the plant side. The plant has limited production capacity and storage capacity. Storage capacities at retailers are unequal and limited. The problem environment allows for backordering and stock out at each period. Materials are transported from plants to retailers by an unlimited fleet of homogeneous vehicles with limited capacity. The objective of the problem is to maximize the profit which is the difference between revenue and cost. The cost includes production cost, inventory holding cost, transportation cost, backordering cost and stock out cost. Furthermore we consider a fixed cost to be incurred for each extra vehicle used and also set up cost is considered at the plant side. The per unit cost and sales price at each retailer for each product is different. Also we assume that size of each item is the same.

3. The mathematical model

The integrated production and distribution planning problem is proposed as an integer programming model. It is formulated as given below.

\[
\text{Max} \sum_{k} \sum_{i} P_{jk} \sum_{l} \left( AR_{jkt-1} + \sum_{i} Q_{jkt} - AR_{jkt} \right) - \left[ \sum_{k} \sum_{l} C_{ik} \sum_{t} x_{ikt} + \sum_{k} \sum_{i} S_{ik} \sum_{t} y_{ikt} + \sum_{i} APO_{ik} \sum_{k} AP_{i} + \sum_{k} AR_{ik} \sum_{t} \sum_{i} V_{jk} \sum_{t} SO_{jkt} + \sum_{k} BC_{jk} \sum_{t} B_{jkt} \right] + G \sum_{i} \sum_{t} Z_{ijt} + \sum_{i} \sum_{t} \sum_{k} D_{ij} \sum_{t} Q_{jkt} \right) \\
\text{Subject to} \\
\sum_{k} O_{ik} x_{ikt} + \sum_{k} U_{ik} y_{ikt} \leq L_{i} \quad \forall \, i, t \\
x_{ikt} \leq My_{ikt} \quad \forall \, i, k, t \\
AP_{ikt} = AP_{ikt-1} + x_{ikt} - \sum_{i} Q_{ijkt} \quad \forall \, i, k, t \\
\sum_{k} AP_{ikt} \leq ST_{i} \quad \forall \, j, t \\
B_{jkt} = F_{jkt} - AR_{jkt-1} - \sum_{i} Q_{jkt} + AR_{jkt} + B_{jkt-1} - SO_{jkt} \quad \forall \, j, k, t \\
\sum_{k} AR_{jkt} \leq W_{j} \quad \forall \, j, t \\
Z_{ijt} \geq \sum_{k} \frac{O_{ijkt}}{BC_{CAP}} \quad \forall \, i, j, t
\]
The objective function (1) denotes the net profit which is to be maximized. Net profit is revenue minus the total cost. The total cost is the sum of production cost, inventory holding costs, transportation cost, stock out cost, and backordering cost. Constraint (2) limits the production at each plant to be within a given number of hours. Constraint (3) enforces that set up has to take place in order for production to take place. The parameter $M$ is a sufficiently large positive number. Constraint (4) assures the material balance at each plant. Constraint (5) is the plant side storage restriction. Constraint (6) assures material balance at each retailer. Constraint (7) enforces the inventory at the retailer to be within a given limit. Constraint (8) relates the number of vehicles and the delivery quantity. Constraint (9) respectively states that the beginning on-hand inventories at plant and retailer sides are zero. Constraint (10) means that initial period and final period backorders are zero. Constraint (11), along with constraints (12) enforces the non-negativity on the decision variables. Constraint (13) represents binary nature of the related decision variables.

The production plan, delivery schedule, resulting inventory decision, and the stock out/backordering decisions are obtained by solving the problem.

4. Solution procedure

The model as initially solved using LINGO solver and the complexity associated with the problem can judged by the time taken by LINGO to solve the problem. These time values are given in table 1. The integrated production distribution planning problem has been reported in literature as an NP hard. The present problem is also a variant of integrated production distribution planning problem and it has some additional constraints. So the given problem is also NP hard. The complexity of the model can also be seen from the empirical results in table 1. So, genetic algorithm was used to solve the problem. The GA was coded in Scilab. The solutions in the GA are represented as a matrix. These solutions are called chromosomes. The algorithm should give a solution that answers to four decisions namely production plan, plant side inventory decision, dispatch plan, and retailer side inventory decision.

<table>
<thead>
<tr>
<th>Plants</th>
<th>Retail outlets</th>
<th>Products</th>
<th>Periods</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>00.02.40</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>00.19.24</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>00.10.30</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>09.19.30</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>Interrupted after 24hrs</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>Interrupted after 24 hrs</td>
</tr>
</tbody>
</table>
4.1 Chromosome representation

The chromosomes are represented as a four dimensional matrix. The order of the matrix is $I \times (J+1)$. There are $K \times T$ such matrices in a single chromosome. A sample chromosome is given below in table 2.

<table>
<thead>
<tr>
<th>$k=2$, $i=1$</th>
<th>$j=1$</th>
<th>$j=2$</th>
<th>$j=3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i=1$</td>
<td>20</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>$i=2$</td>
<td>110</td>
<td>0</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2. Chromosome representation

Table 2 shows a chromosome representation. Consider a case where there are 2 plants, 2 retailers, 2 products and 3 periods. The chromosome represents details of product 2 in period 1. Such matrices exist for every period and every product. The chromosome denotes a case where 20 units are transferred from plant 1 to retailer 1, 60 from plant 1 to retailer 2, 10 units are produced at plant 1 and added to existing inventory. Similarly the row corresponding to plant 2 can also be interpreted.

4.2. Initialization

Generate binary random numbers to fill the cells of the chromosome (Table 2). Leave the column of plant side inventory empty. For every cell with a 1, it is replaced by $Q_{ijkt}$, if $m_{ij}$ denotes entries in the binary mask. The plant side storage is initialized as zero.

$$Q_{ijkt} = \frac{F_{jkt} \times \sum_{i=1}^{I} m_{ij}}{I}$$ (14)

Table 3. Binary matrix

<table>
<thead>
<tr>
<th>$k=2$, $i=1$</th>
<th>$j=1$</th>
<th>$j=2$</th>
<th>$j=3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i=1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$i=2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

4.3. Fitness function

The profit value is taken as the fitness function value. If the quantity dispatched and inventory is known, we can calculate the production quantity of each product at each period at each of the plants. Retailer side inventory and backorder for each product in every period can be found if we know the initial inventory and demand at each retailer. Here we use a separate method to determine whether stock out or backordering is to be done if demand is not met. For this we define a quantity called critical time duration ($\tau_{jk}$). It is that time duration after which backlogging becomes more uneconomical in comparison with stock out.

$$\tau_{jk} = \frac{P_{jk} + V_{jk} - \left( \sum_{i=1}^{I} \sum_{k=1}^{K} C_{ik} \right) / \left( I \times K \right) - \left( \sum_{i=1}^{I} \sum_{j=1}^{J} C_{ij} \right) / \left( I \times J \right)}{BC_{jk}}$$ (15)

The critical period method is described below.

1. Find the backorder for all products for all periods at every retailer.
2. For every product and retailer do step 3 to 5
3. Calculate $r_{jk}$
4. Do 5 for all periods from 1 to $T$
5. At any period $t$ determine the consecutive number of subsequent periods over which backorder are maintained. If the backorder is maintained for a duration more than the critical time duration, then consider those backordered units as lost sales.

Any unit of backorder that is held until the last period should be treated as lost sales as the first period the backordering starts from. In this way all quantities related to the relevant cost of the problem can be identified from the chromosome and fitness function values can be calculated.

4.4. Selection operator

Here we used a selection procedure based on tournament selection. Tournament selection always screens out the worst solution as we select potential entries into the mating pool. In addition to the traditional tournament operator, in this algorithm the best solution is always included into the mating pool in each generation. In any generation, the first solution in our mating pool is the best solution from the previous generation only the rest of the positions are filled by tournament selection.

4.5. Crossover operator

To do the crossover we convert the entries of the chromosome into binary equivalent numbers and do a uniform crossover by selecting any two random entries from the mating pool table 5 and table 6. A binary mask is created randomly and children are formed using this as shown in table 6 to 11. The first child is formed by assigning positions with zeros entry with corresponding values from parent 1 and the other positions with values from parent 2. The second child is formed by assigning the positions with zero entry with corresponding values from parent 2 and other locations from parent 1. Crossover is carried out probabilistically.

<table>
<thead>
<tr>
<th>Table 4. Parent 1 with decimal entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k=3, t=1$</td>
</tr>
<tr>
<td>$i=1$</td>
</tr>
<tr>
<td>$i=2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5. Parent 1 with all entries converted to binary numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k=3, t=1$</td>
</tr>
<tr>
<td>$i=1$</td>
</tr>
<tr>
<td>$i=2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6. Parent 2 in binary from</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k=3, t=1$</td>
</tr>
<tr>
<td>$i=1$</td>
</tr>
<tr>
<td>$i=2$</td>
</tr>
</tbody>
</table>
Table 7. Binary mask

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Table 8. Child 1 in binary from

\[ k=3, t=1 \]

\[ j=1 \]

\[ j=2 \]

\[ j=3 \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

Table 9. Child 2 in binary form

\[ k=3, t=1 \]

\[ j=1 \]

\[ j=2 \]

\[ j=3 \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

Table 10. Child 1

\[ k=3, t=1 \]

\[ j=1 \]

\[ j=2 \]

\[ j=3 \] (inventory)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>7</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
</table>

Table 11. Child 2

\[ k=3, t=1 \]

\[ j=1 \]

\[ j=2 \]

\[ j=3 \] (inventory)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>4</th>
<th>1</th>
</tr>
</thead>
</table>

4.6. Mutation

A bitwise mutation operator is adopted in this algorithm. Initially the chromosomes are converted to the binary equivalent and for each entry we generate a random number. If this random number is found to be less than mutation probability then the entry at that location is changed. When a change is made, then a 1 will change to 0 and vice versa. The chromosome that is changed to binary entry is converted back to decimal equivalent only after mutation operation.

4.7. The repair strategy

During the crossover and mutation operations three possible infeasibilities might arise in the solution.
- Plant side production capacity violation
- Plant side inventory capacity violation
- Retailer side inventory capacity violation
An obvious way of removing this infeasibility is to select a random entry that has an effect on the infeasibility and decrease the entry as much as possible until feasibility is obtained. A more efficient procedure is to sort products on the basis of their average benefit to cost ratio. Since inventory is shared by all products, decrease the stock of those products which has low benefit to cost ratio until feasibility with regard to inventory is obtained. Stock of these products at a particular retailer can be decreased by reducing the dispatch amount from a plant from which saving in transportation cost is largest. The plant storage capacity violation is also tackled in a similar way. When production capacity constraint is violated then first identify the product with minimum average benefit to cost ratio. Decrease the amount of this product shipped until feasibility with respect to production is obtained. When plant side and retailer side inventory restrictions are not violated, then production capacity is less likely to be violated.

4.8. Termination condition

The algorithm is run for a specified number of iterations and then terminated. The choice of the number of iterations is made by the decision maker depending on the size of the problem and the nature of the problem, i.e. tactical or operational.

5. Computational results

For our analysis we took 6 cases from Park (2005) and used the same relationship to generate the input data. Crossover probability was taken as 0.9 and mutation probability as 0.1. 100 iterations was considered as the termination condition. The cases considered in this work are categorized as small size problems in literature. Apart from the relationships in literature (Park, 2005), we have developed relationships for plant side storage capacity and backordering cost per unit which are given as eqn. 16 and eqn. 17. The GA described in this work was coded in Scilab 5.4.1 on a 3 GHz core 2 duo processor. The production-distribution problem, which is formulated as mixed integer programming model, is solved using LINGO software. The computational results are as given in table 12.

Table 12. Computational results

<table>
<thead>
<tr>
<th>Problem</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>T</th>
<th>GA</th>
<th>LINGO</th>
<th>% difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>96157</td>
<td>96157</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>198429</td>
<td>197883</td>
<td>0.27</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>205610</td>
<td>205843</td>
<td>-0.11</td>
</tr>
<tr>
<td>P4</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>221898</td>
<td>223457</td>
<td>-0.69</td>
</tr>
<tr>
<td>P5</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>307447</td>
<td>307978</td>
<td>-0.17</td>
</tr>
<tr>
<td>P6</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>278016</td>
<td>288053</td>
<td>-3.48</td>
</tr>
</tbody>
</table>

\[ ST_i = \frac{\max_j W_j}{1.0}, \quad i = 1, \ldots, I. \quad (16) \]

\[ BC_{jk} = 0.1 \times P_{jk} \quad (17) \]

6. Conclusions and scope for future work

The production-distribution problem, which is formulated as an integer programming model, is solved using the solver software, LINGO. The CPU time taken for the solver was found to be largely dependent on number of
products. The input data was generated according to the relationships given in Park (2005). The complexity of the problem is due to the fact that there are a large number of infeasible solutions and identification of feasible solution is difficult. Therefore, even in the metaheuristic approach that we have followed, too much CPU time was consumed for repair operations. The problem can be solved in reasonable amount of time using GA, but as problem size increases a deviation in the solution given by GA and that found by LINGO exist.

The major limitation in the present work is the scheme of crossover where binary to decimal conversion and vice versa takes place. When the quantity to be transported becomes large this part of the program adds consumes too much CPU time. An improvement in this approach can be made by using crossover and mutation schemes that does not require repair or binary conversion.

As a future work, core demand and routing aspects can be considered and GA can be developed for solving that particular problem. Also much improvement can be done on the current algorithm to improve its accuracy. Solving the problem using other metaheuristics is also a good area of research. Non homogeneity of vehicles and space occupied by items were not considered in the current research for the sake of simplicity. This can be taken up as a future direction for research.

References