EFFICIENT LOT-SIZING PROCEDURES FOR MATERIAL REQUIREMENTS PLANNING SYSTEMS

Johnny C. Ho, Abbott Turner College of Business, Columbus State University, Columbus, GA 31907, ho_johnny@colstate.edu, Tel: (706) 562-1668
Yih-Long Chang, Dupree College of Management, Georgia Institute of Technology, Atlanta, GA 30332, yihlong.chang@mgt.gatech.edu, Tel: (404) 894-4334
Adriano O. Solis, Dept. of Information and Decision Sciences, University of Texas at El Paso, El Paso, TX 79968, solis@utep.edu, Tel: (915) 747-7757

ABSTRACT

This paper proposes two constructive heuristics for the well-known single-level uncapacitated dynamic lot-sizing problem. The proposed heuristics, called nLPC and nLPC(i), are developed by modifying the average period cost concept from Silver and Meal’s heuristic, commonly known as Least Period Cost (LPC). An improved tie-breaking stopping rule is also proposed in this paper. The computational results from twenty benchmarking test problems show that both proposed heuristics outperform seven existing heuristics.

Keywords: Heuristics; Lot Sizing; Material Requirements Planning (MRP)

1. INTRODUCTION

The lot-sizing problem is one of the most important determinants on the cost performance of any dependent demand inventory system, such as Material Requirements Planning (MRP). Developing techniques for solving the single-level uncapacitated dynamic lot-sizing problem is significant, because they may be used as building blocks in the solution of larger and more complex problems [2]. Hence, much research has been conducted on this problem in the past four decades. Wagner and Whitin [19] propose a dynamic programming approach to determine the optimal solution for the problem. Due to the significant computational requirements and mathematical complexity of the Wagner-Whitin (WW) algorithm, it has not been extensively applied in practice [1] [5]. Haddock and Hubicki [11] carry out a survey of 263 manufacturing companies on the frequency of usage of 11 lot-sizing methods in MRP software and conclude that lot-sizing heuristics are used exclusively while the WW algorithm is not applied at all. Therefore, the majority of the research papers have been geared towards developing heuristics to solve the lot-sizing problem [7].

The heuristic by Silver and Meal [15], also known as Least Period Cost (LPC), is widely regarded as one of the most well known and effective methods. The LPC heuristic is discussed in detail in most, if not all, popular production/inventory management textbooks [12] [17] [18] that deal directly with the lot-sizing topic. For example, Silver and Meal’s LPC is one of only four heuristics discussed in the introductory operations management textbook by Evans [8], with the other three being Economic Order Quantity (EOQ), Part-Period Balancing (PPB), and Periodic Order Quantity (POQ).
Saydam and Evans [14] perform an extensive and comprehensive computational study of the relative performance of the WW algorithm against four popular heuristics: LPC, EOQ, PPB, and Incremental Part-Period Algorithm (IPPA). Their results show that LPC has the best overall performance, in terms of both speed and quality of solution, among all four heuristics. The average performance of LPC deviates by 1.6% from the optimal cost found by WW algorithm. Pan [13] provides a detailed sensitivity analysis of six popular heuristics using demand pattern, setup cost, and holding cost factors. The six heuristics evaluated are Least Unit Cost (LUC), Part-Period Algorithm (PPA), POQ, EOQ, PPB, and LPC. He concludes that LPC is not only the best in generating accurate solutions, as in Saydam and Evans [14], but it is also the most insensitive to uncertainty in parameter estimation. Wee and Shum [20] study the dynamic lot-sizing problem with deteriorating inventory, i.e., a deterioration cost of inventory is added. The LPC is one of only two heuristics that they employ to solve the deteriorating inventory dynamic lot-sizing problem. The foregoing discussion of earlier studies confirms the importance and broad acceptance of LPC.

Blackburn and Millen [4] test a number of heuristics under a rolling-horizon environment and their results reveal that LPC outperforms the dynamic programming approach, because the latter in this case gives the optimal solution to the wrong problem. Balckburn and Millen [5] conclude that Groff’s Marginal Cost Difference (MCD) [9] and Silver and Meal’s LPC are very similar in cost performance. Both MCD and LPC outperform three other heuristics evaluated in this study. Baker [2] summarizes this literature and reconciles the differences among different papers. He finds that LPC incurs an average cost of less than 1% higher than that of WW algorithm. For a comprehensive review on production lot-sizing, one may refer to Eftekharzadeh [7].

In this paper, we propose two heuristics for the well-known single-level uncapacitated dynamic lot-sizing problem. The proposed heuristics are developed by modifying the average period cost criterion employed by the heuristic from Silver and Meal [15] and by employing an improved tie-breaking stopping rule. We examine the effectiveness of the proposed heuristics by using a set of 20 benchmarking test problems from Baker [2] and Berry [3]. The next Section briefly reviews seven popular heuristics which will be compared with our two proposed heuristics later in this paper. Section 3 presents the two proposed heuristics. Section 4 presents some computational results obtained from solving the 20 benchmarking test problems in the literature. Finally, we conclude this paper in Section 5.

2. REVIEW OF HEURISTICS

In this Section we review seven well-known heuristics for the dynamic lot-sizing problem, which will be used in Section 4 for comparison with the proposed heuristics. Silver and Meal [15] propose a forward heuristic procedure which is based on the Least Period Cost (LPC) concept. LPC sets order quantities in multiples of single-period demands. It decides the number of future periods to supply in a given period by continuing to combine production until the average cost per period begins to increase. The average cost per period is defined as the total cost (sum of inventory holding and setup costs) divided by the number of periods the current order is to span, say k. The Least Unit Cost (LUC) heuristic is similar to LPC except that instead of dividing the total cost by k, we divide it by the total number of units demanded in the k period(s).
The Periodic Order Quantity (POQ) heuristic involves determining an economic time interval. The interval is obtained by dividing the economic order quantity by the average demand per period, then rounding the quotient to the nearest integer. Economic order quantity is defined as \((2DS/h)^{0.5}\), where \(D\) is the total demand, \(S\) is the fixed setup cost, and \(h\) is unit holding cost per period. Orders are placed at these intervals for all future demand requirements.

DeMatteis [6] is credited with the Part-Period Algorithm (PPA). The PPA determines the order quantity in time \(i\), \(Q(i)\), by adding the demand quantity in time \(t = i + 1, i + 2, ..., n\) to \(Q(i)\) as long as the following stopping condition holds true: the total holding cost in the lot remains less than or equal to the fixed setup cost. The term “part-period” refers to the equivalent of a part held for one period. Hence, six part-periods could be from one part held for six periods or from three parts held for two periods, etc. An alternative heuristic can be created by modifying the stopping rule of PPA as follows: the total holding cost in the lot is strictly less than the setup cost. We will denote the alternative PPA as PPA(-) employing the same notation as in Baker [2].

Gorham [10] proposes the Least Total Cost (LTC) heuristic which resembles PPA. LTC sets the order horizon equal to the number of periods that most closely matches the total holding cost with the fixed setup cost over that period. In case two or more consecutive periods show identical difference between the total holding cost and setup cost, LTC selects the larger number of periods. Therefore, it attempts to balance the total holding cost and the setup cost in the lot as much as possible. The “balancing” feature probably gave rise to the name Part-Period Balancing (PPB), but we choose to label it as LTC consistent with Gorham [10] and Baker [2]. A variation of the LTC heuristic, called LTC(-), chooses the smaller number of periods in case two or more consecutive periods have equal differences between the total holding cost and setup cost.

3. THE PROPOSED HEURISTICS

The proposed heuristics may be viewed as variations of the Least Period Cost (LPC) method by Silver and Meal [15]. The LPC is generally considered as the most effective method among the popular lot-sizing heuristics discussed in many textbooks (e.g., [18] [8] [17] [12]). Unlike some other modifications applied to LPC [16], which are essentially improvement methods with LPC as seed solution, the proposed heuristics are constructive LPC-based heuristics.

Let \(d(i)\) be the demand requirement in period \(i\) of a \(n\)-period scheduling horizon and \(Q(i)\) denote the lot-sizing quantity in period \(i\). Furthermore, let \(TC(i, j)\) be the total cost, i.e., sum of holding and setup costs, for lot sizing quantity \(Q(i)\), such that \(Q(i) = d(i) + d(i + 1) + ... + d(j)\). Hence, \(TC(i, j) = S + [h \cdot d(i + 1) + 2 \cdot d(i + 2) + ... + (j - i) \cdot d(j)]\).

We define net average period cost, denoted by \(nAPC(i, j)\), as the ratio of total cost to the number of non-zero demand period(s). Therefore,

\[
nAPC(i, j) = \frac{TC(i, j)}{(j - i + 1 - z)}
\]

where \(z\) is the number of periods between periods \(i\) and \(j\) (inclusive) with zero demand.
We will call the first proposed heuristic as net Least Period Cost and denote it as nLPC, since it is based on the nAPC concept. Without loss of generality, both periods 1 and n (n \geq 2) are assumed to have non-zero demand requirement. The steps of nLPC are given below.

**Algorithm nLPC**

Step 0: Initialize i = 1, j = 1, and Q(k) = 0, for k = 1, 2, ..., n.

Step 1: If j = n, then set Q(i) = d(i) + ... + d(n) and go to Step 5; else set j = j + 1 and enter Step 2.

Step 2: If d(j) \neq 0, then enter Step 3; else go to Step 1.

Step 3: Find nAPC(i, j) and nAPC(i, j – 1) using Eq. (1). If nAPC(i, j) > nAPC(i, j – 1), then set Q(i) = d(i) + ... + d(j–1) and update i = j.

Step 4: Go to Step 1.

Step 5: Output Q(k), for k = 1, 2, ..., n.

Step 0 is an initialization step. It sets indexes i and j equal to one and all lot-sizing quantities equal to zero. Step 1 checks whether j is equal to n. If the condition is met, we reach the end of the scheduling horizon. So we compute Q(i) = d(i) + ... + d(n) and terminate by outputting Q(k) \forall k in Step 5. If not, we increment j by 1 and check whether or not demand requirement in period j is zero. If d(j) = 0, we go back to Step 1 as period j will definitely be included in the current lot-sizing determination. If d(j) > 0, we check the stopping condition, nAPC(i, j) > nAPC(i, j – 1). If the condition is true, then we conclude that Q(i) = d(i) + ... + d(j–1) and update i = j. We now go back to Step 1. This process continues until j = n. The computational complexity of nLPC, as with LPC, is O(n^2).

The major difference between nLPC and LPC has to do with how they handle demand periods with zero demand requirements. When computing nAPC(i, j), nLPC divides the total cost associated with lot sizing quantity Q(i) by the total number of non-zero demand periods between periods i and j (inclusive). On the other hand, LPC divides the total cost by the total number of demand periods between periods i and j (inclusive), regardless of whether or not demand in a period is non-zero, to derive APC(i, j). The rationale for nLPC is that a zero demand period does not require a setup and does not inflate holding cost when it is evaluated in isolation. Nonetheless, it will increase the holding cost for periods following it when it is considered along with the other periods. Hence, the nAPC addresses the aim of improving cost performance under scenarios where zero demands are not uncommon. The nAPC accomplishes this aim by generally lengthening the order coverage or reducing the total number of orders in the planning horizon.

It is important to note that in Step 3 nLPC stops increasing the lot size quantity, Q(i), only if nAPC(i, j) is strictly larger than nAPC(i, j – 1). This guarantees that nLPC always includes the period(s) with zero demand requirement in its current lot-sizing quantity. Indeed, when d(j) = 0, nLPC skips evaluating the condition in Step 3, i.e., comparing nAPC(i, j) and nAPC(i, j – 1). Since nLPC would then return directly back to Step 1 to consider whether it should incorporate d(j + 1) to its current lot size quantity. Furthermore, nLPC is identical to LPC in terms of
computational complexity as the only difference between LPC and nLPC lies in the definition of average period cost. Since practitioners are reported to be familiar with LPC [11], nLPC should be fairly easy for practitioners to understand and thus to apply in practice.

The second proposed heuristic, similar to nLPC, is also based on the nAPC concept and will be labeled as nLPC(i). The ‘(i)’ refers to the employment of an improved tie-breaking stopping rule for the nLPC heuristic. The steps of nLPC(i) are given below.

Algorithm nLPC(i)
Step 0: Initialize \( i = 1, j = 1, \) and \( Q(k) = 0, \) for \( k = 1, 2, \ldots, n. \)
Step 1: If \( j = n, \) then set \( Q(i) = d(i) + \ldots + d(n) \) and go to Step 5; else set \( j = j + 1 \) and enter Step 2.
Step 2: If \( d(j) \neq 0, \) then enter Step 3; else go to Step 1.
Step 3: Find \( nAPC(i, j) \) and \( nAPC(i, j-1) \) using Eq. (1). If \( nAPC(i, j) > nAPC(i, j-1) \) or \( nAPC(i, j) = nAPC(i, j-1) = S, \) then set \( Q(i) = d(i) + \ldots + d(j-1) \) and update \( i = j. \)
Step 4: Go to Step 1.
Step 5: Output \( Q(k), \) for \( k = 1, 2, \ldots, n. \)

As shown in Step 3 of nLPC(i), the proposed tie-breaking stopping rule consists of two conditions: \( nAPC(i, j) > nAPC(i, j-1) \) and \( nAPC(i, j) = nAPC(i, j-1) = S. \) The first stopping condition provides a tie-breaking rule identical to that of LPC or nLPC. The second stopping condition is proposed as part of the tie-breaking rule of nLPC(i). The logic behind the second stopping condition is given as follows. Suppose that \( APC(i, i+1) = S \) and \( d(j) > 0, j = i, i+1, \ldots, n, \) then the cost of setting \( Q(i) = d(i) \) and \( Q(i+1) = d(i+1) \) is identical to that of setting \( Q(i) = d(i) + d(i+1). \) This is because the additional setup cost required in period \( i+1 \) is equal to the holding cost of carrying \( d(i) \) units of inventory from period \( i \) to period \( i+1. \) Now consider the following two cases given the second stopping condition is met: (1) \( APC(i, i+2) \leq APC(i, i+1) \) and (2) \( APC(i, i+2) > APC(i, i+1). \) In case (1), LPC yields \( Q(i) \geq d(i) + d(i+1) + d(i+2), \) which is clearly suboptimal. For example, if \( APC(i, i+3) > APC(i, i+2) \), then LPC produces \( Q(i) = d(i) + d(i+1) + d(i+2), \) which is dominated by \( Q(i) = d(i) \) and \( Q(i+1) = d(i+1) + d(i+2). \) The second stopping rule helps prevent this suboptimal situation from happening. The nAPC(i) may of course yield a solution with a larger lot-sizing quantity than \( d(i+1) + d(i+2) \) in period \( i+1 \) for the example. If \( Q(i+1) > d(i+1) + d(i+2) \), then there is no guarantee that nLPC(i) will ultimately outperform LPC. But we conjecture that nLPC(i) is more likely to produce a better solution than that of LPC in situations like this. In case (2), LPC yields \( Q(i) = d(i) + d(i+1), \) while nLPC(i) yields \( Q(i) = d(i) \) and \( Q(i+1) \geq d(i+1). \) Again, we conjecture that nLPC(i) is more likely to produce a better solution than that of LPC in situations like this. Furthermore, one positive side effect of employing the proposed tie-breaking stopping rule by nLPC(i) is that it always yields a larger or an equal number of orders than nLPC for a given problem that could allow for greater scheduling flexibility. Lastly, the computational complexity of nLPC(i), similar to nLPC, is also \( O(n^2). \)
4. RESULTS FROM TWENTY BENCHMARKING TEST PROBLEMS

We evaluate the performance of nLPC and nLPC(i) and the seven popular heuristics described in Section 2 using the 20 test problems given in Baker [2]. The data set is important because it is precisely specified and has been able to highlight many differences in performance among heuristics. Table 1 provides the cost performance results for all nine heuristics arising from the data set. They are sequenced by ascending order of the average percent deviation from the optimal solution of all 20 test problems. As shown in the Table, nLPC(i) possesses the lowest mean percent suboptimal (0.626%). It is followed by nLPC (0.669%), the third best heuristic being LPC (0.943%). Hence, nLPC(i) and nLPC are able to reduce the mean percent suboptimal of LPC by 33.6% and 29.1%, respectively. The next two best heuristics are PPA(-) and PPA with mean percent suboptimal of 0.953% and 1.339%, respectively. In terms of the number of suboptimal problems, PPA(-) produces the best performance of only three suboptimal problems. The nLPC(i), nLPC, and LPC heuristics produce four, five, and six suboptimal problems, respectively. The LTC heuristic has both the highest mean percent suboptimal (10.683%) and the largest number of suboptimal problems (15). The last two columns give the minimum and maximum percent sub-optimal among problems solved sub-optimally for each of the heuristics.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Average percent sub-optimal</th>
<th>Number of sub-optimal problems</th>
<th>Min % sub-optimal if sub-optimal</th>
<th>Max % sub-optimal if sub-optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLPC(i)</td>
<td>0.626</td>
<td>4</td>
<td>0.957</td>
<td>8.363</td>
</tr>
<tr>
<td>nLPC</td>
<td>0.669</td>
<td>5</td>
<td>0.858</td>
<td>8.363</td>
</tr>
<tr>
<td>LPC</td>
<td>0.943</td>
<td>6</td>
<td>0.858</td>
<td>8.363</td>
</tr>
<tr>
<td>PPA(-)</td>
<td>0.953</td>
<td>3</td>
<td>4.181</td>
<td>7.692</td>
</tr>
<tr>
<td>PPA</td>
<td>1.339</td>
<td>4</td>
<td>4.181</td>
<td>7.725</td>
</tr>
<tr>
<td>LTC(-)</td>
<td>8.014</td>
<td>13</td>
<td>1.449</td>
<td>26.667</td>
</tr>
<tr>
<td>LUC</td>
<td>9.303</td>
<td>12</td>
<td>1.653</td>
<td>39.823</td>
</tr>
<tr>
<td>POQ</td>
<td>10.009</td>
<td>13</td>
<td>2.857</td>
<td>43.872</td>
</tr>
<tr>
<td>LTC</td>
<td>10.683</td>
<td>15</td>
<td>1.449</td>
<td>50.000</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

This paper proposes two efficient constructive heuristics, called nLPC and nLPC(i), for the well-known single-level uncapacitated dynamic lot-sizing problem. Both heuristics are based on the proposed net average period cost concept, which is derived from modifying the average period cost idea from the Silver-Meal heuristic [15], also known as Least Period Cost (LPC). The second proposed heuristic, nLPC(i), employs an improved tie-breaking stopping rule to enhance the heuristic’s performance. Also, we examine the effectiveness of nLPC and nLPC(i) against seven popular heuristics in the literature by using the well-known set of 20 test problems from Baker [2] and Berry [3].

REFERENCES

References available upon request from Johnny C. Ho, (706) 562-1668.