Variable Neighborhood Search Based Approach for Solving Multilevel Lot-Sizing Problems

YIYONG XIAO¹, IKOU KAKU²*

¹Department of System Engineering of Technology Engineering
Beihang University
Beijing, 100191, China
xiaoyiyong@263.net

²Department of Management Science and Engineering*
Akita Prefectural University
Yulihonjo, Akita, 015-0055, Japan
ikou_kaku@akita-pu.ac.jp

* Corresponding author

Abstract: In this paper, we present an effective approach based on the variable neighborhood search (VNS) for solving multilevel lot-sizing (MLLS) problems. Two kinds of neighborhood search strategies, i.e., move at first improvement (MAFI) and move at best improvement (MABI), are adopted to improve the performance of proposed algorithm. Computational experiments are carried out on 96 benchmark problems to test the optimality against genetic algorithm on identical problems, and also to analyze the mechanism of VNS while it solving MLLS problem. Experimental outcomes show that the VNS algorithm equipped with MABI and emendation by inner corner property enjoys good optimality and high computation effectiveness as well, which is quite competitive to the existing algorithms that have been studied on the MLLS problems.

Key-Words: multilevel lot-sizing, variable neighborhood search, meta-heuristic, genetic algorithm

1 Introduction
The multilevel lot-sizing (MLLS) problem concerns how to determine the lot size for producing or procuring an item at each level so as to minimize the total costs of set up and inventory holding. The problem plays an important role in the efficient operation of modern manufacturing and assembly processes. Optimal solution algorithms exist for MLLS problems; however, only small instances can be solved in reasonable computational time because the problem is NP-hard [1]. In the past decades, not only optimization formulations and algorithms ([2], [3], [4]) but also heuristic approaches by using single level decision rules ([5], [6]) and multi level nature of the problem ([7]-[9]) have been developed to solve variants of the MLLS problem. Recently, meta-algorithms have been proposed to solve the MLLS problem with a low computational load (for example, genetic algorithms [10] [11], simulated annealing [12], particle swarm optimization [13], soft optimization [14] and so on). It has been claimed that these algorithms can provide highly cost-efficient solutions with reasonable computation time. Unfortunately, due to their complicated procedures for constructing samples, they do not retain these characteristics when applied to problems of a realistic size.

In this paper, a variable neighborhood search (VNS) based approach is presented newly for solving the MLLS problems. The variable neighborhood search is a top-level methodology that had been applied for solving combinatorial optimization problems in many cases. The main reasoning of this meta-heuristic, in comparison to most other heuristics of past where only one neighborhood is used, is based on the idea of a systematic change of multi-leveled neighborhoods within a local search method (details can be found in [15] [16]). Therefore, it is expected to be useful in dealing with complicated MLLS problems with time-invariant cost structures and no restriction on product structure. We conduct simulation experiments to compare the new algorithm’s performance to that of a genetic algorithm developed in [10] [11], and show the superiority of the proposed approach.
The paper is organized as follows. Section 2 describes the MLLS problem. Section 3 explains heuristic considerations and presents the new algorithm. Section 4 describes the comparative experiments, discusses the presents. Section 5 summarizes our conclusions.

2 The MLLS Problem

The MLLS problem is considered as a discrete-time, multilevel production/inventory system with an assembly structure and one finished item. We assume that external demand of the finished item is known up to the planning horizon, that backlog is not allowed for any items and that we have zero lead-times for all production items. Suppose that there are $n$ items and the planning horizon is divided into $m$ periods. $x = \{x_{i,t}\}_{m \times n}$ represents the solution space of the MLLS problem, where $x_{i,t} \in \{0,1\}$ is a binary decision variable for item $i$ in period $t$. Calculation of the total cost function of set-up and inventory holding cost can be defined uniformly when $X$ is fixed. Our purpose is to find the lot sizes of all items so as to minimize the total costs, while ensuring that external demands for the end item are met over the $m$-period planning horizon. Then searching for an optimal solution of the MLLS problem, where $x_{i,t} \geq x_{k,j}$ if item $i$ creates internal demand for item $k$. Thus we need a constraint in order to guarantee that the matrix is feasible.

Moreover, we set the first column of the matrix to be ‘1’ in order to ensure that the initial production is feasible because backlog is not allowed for any items and the lead-times are zero. Since there is an inner corner property [12], we need to have $x_{i,t} \geq x_{k,j}$ if item $i$ creates internal demand for item $k$. Thus we need a constraint in order to guarantee that the matrix is feasible.

3 Description of the new algorithm

3.1 The method of solving MLLS by VNS

The difficulty of solving MLLS problem has been recognized for years. The size of its potential solution space, theoretically expected to be $2^{(n-1)\cdot m}$ for a problem with $n$ items and $m$ periods, will exponentially accelerate to considerable large as the problem’s scale increases. Although slight alleviation may be done to some extent by considering the inner corner property (see [12]) of product structure, it is still very difficult to find an optimal solution.

The VNS algorithms, by introducing a set of distance-leveled neighborhoods and correlated exploration order, e.g., from near to far, differs from most local search heuristics of the past where only one neighborhood is used. It is therefore promoted into the meta-heuristics and capable of finding optimal solution in large solution space. The basic VNS consists of two general phases. The first phase includes defining a neighborhood structure, selecting an exploring sequence, initiating a solution $x_0$ as incumbent $(x \leftarrow x_0)$, and defining a stop condition. The second is an iterated local search process whereby to explore a better/best solution from the neighborhoods of incumbent solution. In such process, the nearest neighborhood $N_i(x)$ is commonly explored at first($k \leftarrow 1$); if no better solution is found then changes to a farther neighborhood $N_i(x)(k \leftarrow k+1)$ until $k=k_{max}$. The shaking (or perturbation) operation at the end of each local search acts as a role to force the incumbent solution out of local optima, which features the VNS based algorithm to search better solution far from the incumbent.

Basically, two main aspects of the basic VNS are often reconfigured when applying to a new problem. One is neighborhood structure that should be redefined according to concrete problems; the other is the local search algorithm, which is used to explore the local optima from a neighborhood and should be properly selected to match to the problem’s inherent natures. Both of these two aspects are crucial for constructing an efficient VNS algorithm and also for good quality of finally solution found.

3.2 Neighborhood structure for MLLS

We use the similar definition used in [15] to define the neighborhoods for the MLLS problems, where the distance-leveled neighborhoods are based on distance metric measured by the number of different points between two solutions. The distance metric is as follows.

**Definition 1.** *Distance between two solutions:* For a set of feasible solution of a MLLS problem, i.e., $X = \{x\}$, the distance between any two solutions $x$ and $x'$ can be measured by

$$\rho(x, x') = \left| x \setminus x' \right| = \left| x' \setminus x \right| \quad \forall x, x' \in X$$

Where $\left| \setminus \right|$ denotes the number of different points between two solutions.

The Definition 1 can be further illustrated by the following example. For a MLLS problem with 3 items
and 3 periods, suppose \( x, x', \) and \( x'' \) to be three solutions.

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

According Definition 1, we can get their distances:

\[
\rho(x, x') = 1, \quad \rho(x', x'') = 1, \quad \text{and} \quad \rho(x, x'') = 2.
\]

Based on the distance metric of Definition 1, the distance-leveled neighborhood structure of incumbent solution can then be introduced as follows.

**Definition 2. Neighborhood of incumbent solution**: A solution \( x' \) belongs to the \( k \)-neighborhood, i.e., \( N_k(x) \), of incumbent solution \( x \) if and only if it satisfies

\[
\rho(x, x') = k.
\]

The Definition 2 can be simply expressed as

\[
x' \in N_k(x) \iff \rho(x, x') = k, \quad \text{where } k \text{ is a positive integer.}
\]

For the same example used to illustrate Definition 1, according to Definition 2, we can get that \( x' \in N_1(x) \) and \( x'' \in N_2(x) \), as shown in Fig. 1.

![Fig. 1 An example of changing \( x \) into its neighbors](image)

For example, to extract a solution from the first neighborhood, i.e., \( N_1(x) \), of incumbent solution, we just need to select an element in \( x \) randomly, i.e., \( x_{i_0} \in x \), and revise its value; namely, if \( x_{i_0} = 0 \) then set \( x'_{i_0} = 1 \), otherwise set \( x'_{i_0} = 0 \). Similarly, solutions in the \( k \)-neighborhood, i.e., \( N_k(x) \), of incumbent solution can also be quickly reached by revising the values of \( k \) different elements in incumbent solution.

Next, we give an illustration on the methodology of applying VNS algorithm to solve the MLLS problem. According to the principle of basic VNS, the nearby neighborhoods, e.g., \( N_k(x) \), are to be explored firstly for better solution, such that less changes, e.g., only one point when exploring \( N_k(x) \), are done to the incumbent solution. Therefore, it can avoid too much damage and well reserve the good properties of the incumbent solution, e.g., some elements are already in its optimal positions. However, when the nearby neighborhoods cannot produce any better solutions, VNS algorithm is then forced to explore farther neighborhood for better one, which indicates continuous changes are done to the incumbent solution, regardless of its middle states are better or worse.

### 3.3 The proposed VNS algorithm for MLLS

In this paper, we proposed two algorithms based on basic VNS scheme, as shown in Fig.2(a) and Fig.2(b), to solve the MLLS problem. Two different principles, under which the movement of incumbent solution is decided, are equipped to the two algorithms respectively. The two principles are: (1) move at first improvement (MAFI) and (2) move at best improvement (MABI). MAFI means that a movement of the incumbent solution occurs (\( x \leftarrow x' \)) immediately as a better solution is found, Whereas MABI means the incumbent solution moves only to the best solution found in a neighborhood after a number of searching loops. The differences are embodied in the local search step, i.e., step 2a, of two algorithms.

**Initialization.** Define the set of neighborhood structures \( N_k, k=1,.., k_{max} \), that will be used in the search; find an initial solution \( x \); choose a stopping condition. Repeat the following until the stopping condition is met: (1)Set \( k=1 \); (2)Until \( k=k_{max} \) Repeat (a), (b) steps:

(a)Local search. Search at random a better solution \( x' \in N_k(x) \);
(b)Move or not. If \( x' \) is better than \( x \), move there (\( x \leftarrow x' \)), and continue the search from \( N_k(x)(k+1) \); otherwise, set \( k \leftarrow k+1 \);
(c) Perturbing/Shaking the incumbent solution \( x \) to force it out of local optima.

**Fig. 2(a) The VNS algorithm with MAFI**

**Initialization.** Define the set of neighborhood structures \( N_k, k=1,.., k_{max} \), that will be used in the search; find an initial solution \( x \); choose a stopping condition. Repeat the following until the stopping condition is met: (1)Set \( k=1 \); (2)Until \( k=k_{max} \) Repeat (a), (b) steps:

(a)Local search. Search the best solution \( x' \in N_k(x) \);
(b)Move or not. If \( x' \) is better than \( x \), move there (\( x \leftarrow x' \)), and continue the search from \( N_k(x)(k+1) \); otherwise, set \( k \leftarrow k+1 \);
(c) Perturbing/Shaking the incumbent solution \( x \) to force it out of local optima.

**Fig. 2(b) The VNS algorithm with MABI**

The perturbing/shaking operation in step (3) is crucial for enhancing the uphill ability of the
algorithms which forces the incumbent out of local optima to relocate global optima. For simplicity, we just use a re-initiation as step (3) in our program code of the proposed algorithm.

The stop conditions may be various. Basically, two kinds of conditions are commonly used. One is a maximum computing time, and the other is a maximum number of perturbing/shaking as no better solution can be found. The second is the one chosen as the stop condition in our program, which is actually an adjustable parameter noted as $p$. If $p=5$ then the program will stop after 5 times of perturbing/shaking are executed but with no better solution having been found.

4 Experiments and discussion
4.1 Problem set
The problem set to which our experiments are applied is the well-known 96 benchmark problems used in [10] and [11], which are all small-sized MLLS problems with known optimal solutions. Four typical product structures are considered in the problem set and each of them is with five-item assembly structure and one-to-one production ratio. Lead time is zero for all items. For each product structure, 4 cost combinations are considered which assign each individual item with different set up cost and different unit holding cost. Six independent demand patterns with variations to reflecting low, medium and high demand are considered over 12-period planning horizon. Therefore, the combinations produce $4\times4\times6=96$ benchmark problems for testing.

We apply the VNS algorithms to the 96 benchmark problems. Two principles for the movement of incumbent solution, i.e., MAFI and MABI, are compared in same algorithm. The stop condition is an integer threshold, i.e., $p$, a maximum number of perturbing/shaking as no better solution has been found. For each benchmark problem, we repeatedly run the algorithm for 10 times with respect to different stop conditions, i.e., $p=5$, $p=10$ and $p=20$, respectively.

4.2 Results and comparison with GA algorithm
The computation result is shown in Table 1. It can be observed that the algorithm with MABI show its obvious superiority over that with MAFI on nearly all aspects. For example, on the optimality, MABI can find 95.31% of 960 optimal solutions while MAFI can only find 91.98% when $p=20$, similarly when $p=10$ and $p=5$. Next, we run the same calculations as above but with emendation from inner corner property in the algorithm, where the difference is that an emendation is always applied to any newly produced solution to force it in line with inner corner property. The computation results are shown in Table 2. For the purpose of comparison, the performance of genetic algorithm in [10] on the same benchmark problems is also appended to Table 2. It is similar to Table 1 that the algorithms with MABI continue to show its superiority to that with MAFI in Table 2. However, due to the effect of emendation from inner corner property, both of the two algorithms in Table 2 are much better than that in Table 1. Therefore, after compared the data in Table 1 and Table 2, we come to two conclusions about applying VNS algorithms to MLLS problem, which are: (1) the MABI is better than MAFI as a local search algorithm for the MLLS problems; (2) the consideration of inner corner property can well improve the performance of algorithm.

| Table 1. Run 10 times on 96 benchmark problems without emendation |
|-----------------|-------|--------|---------|--------|---------|---------|---------|-------|
|                 | Comp. | Mean   | Average | Optimal | Average | Maximum | Avg. Dev. |
|                | time(s) | Cost   | Index(%)| (n=960)%| Deviation| Deviation| When not optimal | |
| **MAFI**       |       |        |         |         |         |         |           |       |
| p=5            | 0.10  | 814.651| 0.491   | 66.67   | 0.493   | 15.737  | 1.478     |       |
| p=10           | 0.18  | 812.169| 0.185   | 82.08   | 0.178   | 7.040   | 0.993     |       |
| p=20           | 0.32  | 811.107| 0.054   | 91.98   | 0.049   | 5.710   | 0.608     |       |
| **MABI**       |       |        |         |         |         |         |           |       |
| p=5            | 0.13  | 813.022| 0.290   | 76.35   | 0.279   | 7.744   | 1.178     |       |
| p=10           | 0.22  | 811.220| 0.068   | 89.69   | 0.060   | 3.513   | 0.578     |       |
| p=20           | 0.39  | 810.847| 0.022   | 95.31   | 0.021   | 5.967   | 0.458     |       |
We compare the best result obtained by VNS algorithms (with emendation, MABI, \( p = 20 \)) to the best result obtained by genetic algorithm (GA50) of [10], appended at the balance of Table 2. It can be observed that on nearly all indexes, including mean cost, optimality, maximum deviation, and average deviation when not optimal, the VNS algorithm shows its superiority to that of genetic algorithm.

### 4.3 Mechanism analysis of VNS for solving MLLS

We use the experiment data to show the mechanism of VNS algorithm for MLLS problems.

To know about the extents to which different distance-leveled neighborhoods have contributed to improve the incumbent solution in the whole search progress, we run the algorithm 10 times on the identical 96 benchmark problems with four sets of parameter, to count how many times the improvement of incumbent solution has occurred within different neighborhoods. Fig.3 gives the distributions in percentages of the movements in different neighborhoods, i.e., \( N_1(x) \), \( N_2(x) \), \( N_3(x) \) and \( N_4(x) \) (where \( k_{\text{max}} = 5 \)). The results of four sets of parameters are compared in the graph. We can see that most of the movements (nearly 90%) are occurred in first neighborhood, i.e., the \( N_1(x) \), however, there are still many movements that occur in \( N_2(x) \), \( N_3(x) \) and even in \( N_4(x) \). The minority movements of later are in fact very crucial for the quality of final solution because each of them make a further improvement to the incumbent solution after its nearby neighborhoods cannot produce any better solution. We deem they have reflected the phenomena that the algorithm had successfully climbed over high “mountains” (the explored neighborhoods) and found better solution far from the incumbent solution.

![Fig.3 Percentage of the incumbent solution’s movements in different neighborhoods (10 runs on 96 problems, \( p = 10 \))](image)

A longer run (\( p = 40 \)) has been done to examine the effect of perturbing/shaking to the quality of solution. This run is still on the 96 benchmark problems. For each problem we have recorded the cost deviations of the incumbent to optimal (in percentage) before executing a perturbing/shaking. Thes deviations before each perturbing/shaking are shown in Fig. 4. There are four lines in Fig.4 represent the results of different parameters. It can be seen that the average deviations drop very quickly as the times of

<table>
<thead>
<tr>
<th>( MAFI )</th>
<th>Comp. time(s)</th>
<th>Mean Cost</th>
<th>Average Index(%)</th>
<th>Optimality (n=960)%</th>
<th>Average. Deviation</th>
<th>Maximum Deviation</th>
<th>Avg. Dev. When not optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P = 5 )</td>
<td>0.10</td>
<td>811.288</td>
<td>0.076</td>
<td>89.58</td>
<td>0.063</td>
<td>7.570</td>
<td>0.607</td>
</tr>
<tr>
<td>( P = 10 )</td>
<td>0.17</td>
<td>810.796</td>
<td>0.016</td>
<td>94.90</td>
<td>0.012</td>
<td>1.097</td>
<td>0.233</td>
</tr>
<tr>
<td>( P = 20 )</td>
<td>0.31</td>
<td>810.692</td>
<td>0.003</td>
<td>97.40</td>
<td>0.002</td>
<td>0.448</td>
<td>0.078</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( MABI )</th>
<th>Comp. time(s)</th>
<th>Mean Cost</th>
<th>Average Index(%)</th>
<th>Optimality (n=960)%</th>
<th>Average. Deviation</th>
<th>Maximum Deviation</th>
<th>Avg. Dev. When not optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P = 5 )</td>
<td>0.11</td>
<td>811.011</td>
<td>0.042</td>
<td>92.19</td>
<td>0.035</td>
<td>5.241</td>
<td>0.450</td>
</tr>
<tr>
<td>( P = 10 )</td>
<td>0.18</td>
<td>810.778</td>
<td>0.013</td>
<td>96.25</td>
<td>0.010</td>
<td>2.220</td>
<td>0.275</td>
</tr>
<tr>
<td>( P = 20 )</td>
<td>0.33</td>
<td>810.691</td>
<td>0.003</td>
<td>98.33</td>
<td>0.002</td>
<td>1.425</td>
<td>0.128</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>([13])</th>
<th>Comp. time(s)</th>
<th>Mean Cost</th>
<th>Average Index(%)</th>
<th>Optimality (n=96)%</th>
<th>Average. Deviation</th>
<th>Maximum Deviation</th>
<th>Avg. Dev. When not optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA50</td>
<td>--</td>
<td>810.74</td>
<td>0.01</td>
<td>96.88</td>
<td>--</td>
<td>6.00</td>
<td>0.26</td>
</tr>
<tr>
<td>GA5</td>
<td>--</td>
<td>811.01</td>
<td>0.04</td>
<td>89.58</td>
<td>--</td>
<td>9.00</td>
<td>0.35</td>
</tr>
<tr>
<td>GA0</td>
<td>--</td>
<td>811.53</td>
<td>0.11</td>
<td>83.33</td>
<td>--</td>
<td>14.00</td>
<td>0.65</td>
</tr>
<tr>
<td>Avg.</td>
<td>5.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
perturbing/shaking growing. In particular, the emendation of inner corner property and the MABI principle can fast accelerate the dropping speed at the beginning. However, all algorithms can reach a very lower level of average deviation when the computation time is long enough.

Fig. 7 Average deviation of 96 benchmark problems as the steps of Perturbing/Shaking growing (P=40)

5 Conclusion
This paper conducts a research on applying variable neighborhood search algorithm to solve the multi-level lot-sizing problem. Algorithms are presented and experiments are done over 96 small-sized benchmark problems. The commonly used principles: move at best improvement(MABI) and move at first improvement(MAFI) are compared in the experiment, which shows that MABI is superior over MAFI and is therefore recommended. Moreover, the inner corner property of MLLS problem is also confirmed in the experiments that it can make considerable contribution to the good quality of solution, as well as the efficiency of computation.

Future works on this topic are within our schedule that media-sized even large-sized MLLS problems will be focused on to further confirm the superiority of proposed VNS algorithm, and also we hope it can bring more surprise.

References: