**Submitted to**
**manuscript (Please, provide the mansucript number!)**

# An Optimization Framework for Solving Capacitated Multi-level Lot-sizing Problems with Backlogging

Tao Wu[†], Leyuan Shi[†], Joseph Geunes[‡], Kerem Akartunalı[*]
[†] Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706
[‡] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611
[*] Dept. of Management Science, University of Strathclyde, Glasgow G1 1QE, UK

This paper proposes two new mixed integer programming models for capacitated multi-level lot-sizing problems with backlogging, whose linear programming relaxations provide good lower bounds on the optimal solution value. We show that both of these strong formulations yield the same lower bounds. In addition to these theoretical results, we propose a new, effective optimization framework that achieves high quality solutions in reasonable computational time. Computational results show that the proposed optimization framework is superior to other well-known approaches on several important performance dimensions.

*Key words*: Capacitated, Multi-level, Lot-sizing, Backlogging, Lower and Upper Bound Guided Nested Partitions

## 1. Introduction

This paper considers the Multi-Level Capacitated Lot-Sizing problem with Backlogging (ML-CLSB), a class of problems often faced in practical production planning settings. The goal of this problem is to schedule production at each level of a complex bill-of-materials structure over a finite horizon, while minimizing total cost and ultimately meeting all customer demands by the end of the horizon. The total relevant costs include setup costs, inventory holding costs, and backlogging costs. The output of this model consists of the setup periods and the time-phased production, inventory, and backlog quantities at each stage of the bill-of-materials over the planning horizon. Effective solution of this problem is one of the most important determinants of cost performance in any dependent-demand production and inventory control system, which includes the well-known material requirements planning systems prevalent in manufacturing practice. Research on models and solution methods that facilitate more efficient solution of problems in this class thus has the potential to provide tangible benefits to practitioners in the form of lower total production-related costs.

Past literature on the ML-CLSB is reasonably sparse due to the problem's complexity. Most of the past research considered simpler problems, such as the uncapacitated and single-level lot-sizing problems with backlogging. For instance, Zangwill (1969) investigated single level lot-sizing problems with backlogging. Pochet and Wolsey (1988) studied extended reformulations of uncapacitated lot-sizing problems with backlogging and explored their relationships. Federgruen and Tzur (1993) developed an $O(n \log n)$ dynamic programming algorithm for a standard $n$-period single-level lot-sizing problem with backlogging. Millar and Yang (1994) proposed two Lagrangian decomposition and Lagrangian relaxation algorithms for solving the capacitated single level multi-item lot-sizing problem with backlogging. Agra and Constantino (1999) examined the uncapacitated single item lot-sizing problem with backlogging and start-up costs, when Wagner-Whitin costs are assumed. Cheng et al. (2001) formulated single-level lot-sizing problems with provisions

for backorders using a fixed-charge transportation model and proposed a heuristic solution method. Ganas and Papachristos (2005) proposed a polynomial-time algorithm for the single-item lot-sizing problem with backlogging. Song and Chan (2005) proposed a dynamic programming algorithm for solving a single-item lot-sizing problem with backlogging on a single machine at a finite production rate. Mathieu (2006) proposed two extended linear programming (LP) reformulations of single-item lot-sizing problems with backlogging and constant capacities. In a recent study, Küçükyavuz and Pochet (2009) provided the full description of the convex hull for the single-level uncapacitated problem with backlogging. We refer the interested reader to Pochet and Wolsey (2006) for a detailed general review of different lot-sizing problems. We note that the term *backlog* is used interchangeably with *backorder* in the lot-sizing literature, referring to any demand that is not satisfied on time but in a later time period, no matter what type of manufacturing environment. In our context, we consider a model that is flexible enough to apply to both MTO (Make-To-Order) and MTS (Make-To-Stock) environments when production is planned based on fixed demands or forecasts.

The past research has also considered other classes of lot sizing problems. For example, Thizy and van Wassenhove (1985) designed a Lagrangian relaxation (LR) approach, in which capacity constraints are relaxed, in an attempt to decompose the problem into $N$ uncapacitated single item lot-sizing subproblems, solvable by the Wagner-Whitin algorithm. Trigeiro (1987) developed a similar approach for solving the deterministic, multi-item, single-operation lot-sizing problem. Trigeiro et al. (1989) also proposed LR based methods for large-scale lot-sizing problems. Kuik and Salomon (1990) evaluated a simulated annealing heuristic for solving multi-level lot-sizing problem. Pochet and Wolsey (1991) applied strong cutting planes to solve two classes of multi-item lot-sizing problems, a class of single-stage problems involving joint machine capacity constraints and/or start up costs and a class of multistage problems with general product structure. Salomon (1991) developed a tabu search procedure where the solution set is searched quickly in a greedy fashion. Kuik et al. (1993) proposed an LP-based heuristic, and compared the performance of the heuristic with the performance of approaches based on simulated annealing and tabu search techniques. To solve the capacitated single-item lot-sizing problem with setup and reservation costs (a separate reservation cost is incurred for keeping the resource on, whether or not it is used for production), Hindi (1995) developed a tabu search scheme that was shown to be capable of reaching an optimal solution for a large number of varied problem instances tested. Also, Hindi (1996) developed a tabu search procedure for solving the capacitated single-level multi-item lot-sizing problem using a combination of other techniques, such as column generation. Özdamar and Bozyel (2000) proposed a heuristic that combines genetic algorithms and simulated annealing to solve the capacitated multi-item lot-sizing problem. Barbarosoglu and Özdamar (2000) described an analysis of different neighborhood transition schemes and their effects on the performance of a general purpose simulated annealing procedure for solving the capacitated dynamic multi-level lot-sizing problem with general product structures. Miller et al. (2003a,b) studied the polyhedral structure of an MIP model that occurs as a relaxation of a number of structured MIP problems (e.g., capacitated lot-sizing problems with setup times and fixed-charge network flow problems), characterized the extreme points of the convex hull, and defined cover inequalities and reverse cover inequalities for lot-sizing problems with preceding inventory. Hung et al. (2003) proposed a tabu search heuristic for lot-sizing problems involving

multiple products, multiple resources, multiple periods, setup times, and setup costs. Tang (2004) provided a brief presentation of simulated annealing techniques and their applications in lot-sizing problems. Simpson and Erenguc (2005) developed a neighborhood search heuristic method for capacitated multi-stage problems. Karimi et al. (2006) proposed a tabu search heuristic for the capacitated lot-sizing problem with setup carryover. Almada-Lobo and James (2010) proposed a meta-heuristic that incorporates a tabu search and a neighborhood search meta-heuristic to solve the capacitated multi-item lot-sizing and scheduling problem with sequence-dependent setup times and costs.

Akartunalı and Miller (2009) provided one of the few past works that directly considers the ML-CLSB. They generalized the traditional inventory and lot-sizing model originally proposed by Billington et al. (1983), and they added classical lot-sizing $(\ell, S)$ inequalities, which are quite effective in improving LP relaxation lower bounds. They also proposed a heuristic framework for solving problems within this class. Their research was significant in advancing the techniques for solving the ML-CLSB; however, computational results of their framework leave significant room for improvement. In order to further advance the techniques available for solving such problems, we propose two new mixed integer programming (MIP) models that are able to yield strong LP relaxation lower bounds. A new optimization framework is also proposed for this problem class, which can significantly improve solution quality when compared with the prior heuristic methods.

We refer to our new optimization framework as LugNP−Lower and upper bound guided Nested Partitions. LugNP is a method based on partitioning and sampling, which focuses computational effort on the most promising region of the solution space. The basic premise of the framework is that an efficient partitioning and sampling strategy can be achieved by combining domain knowledge from exact and heuristic methods to leverage the strengths of both of these approaches. In this framework, exact methods are used to generate lower bound solutions, while heuristic methods are used to achieve feasible upper bound solutions. The optimization framework effectively utilizes both lower and upper bound solutions, and then provides an efficient partitioning and sampling strategy. The search is largely restricted to the most promising region(s) where good solutions are likely clustered.

LugNP is an extended version of the nested partitions (NP) approach proposed by Shi and Ólafsson (2000, 2007). NP has been efficiently applied to a wide range of MIP problems, such as the traveling salesman problem (Shi et al. 1999), the product design problem (Shi et al. 2001), and the local pickup and delivery problem (Pi et al. 2008). However, the generic NP method is not able to partition promising regions in the case of the complex ML-CLSB sampling problems. LugNP has similarities with the hybrid NP and mathematical programming (HNP-MP) approach proposed previously (Wu et al. 2010). LugNP is superior to HNP-MP in that (i) its method for identifying promising regions can cluster good solutions together, (ii) its partitioning and sampling approach can lead to higher-quality sampled problems or subproblems, and (iii) it continuously refines upper bound solutions, which helps improve the efficiency of partitioning and sampling.

In order to convey the advantages of the proposed approach, it is necessary to explain its similarities to and differences from the classical branch-and-bound approach. The branch-and-bound method creates numerous branches in which parts of integer variable vectors are fixed; all branches must be searched until

fathomed or eliminated (i.e., when their lower bounds are larger than the current best upper bound). When the number of branches is particularly large, it is impossible to search all branches because of computational resource constraints. In LugNP, a subset of integer variables is fixed using branches as well. However, some branches are deemed to fall within a so-called promising region, while other parts are considered to be in the surrounding region. To avoid the disadvantages associated with the standard branch-and-bound approach, LugNP does not search all branches with equal focus. Instead, using domain knowledge provided by lower and upper bound solution values, it effectively places greater effort on promising branches that can possibly lead to near-optimal solutions. Both LugNP and NP are similar to beam search methods in placing greater focus on promising regions (Pinedo 2008). However, LugNP is significantly different from beam search methods in the way it identifies promising regions, the way it partitions promising regions into several subregions, and in obtaining good samples of subproblems, in addition to its ability to perform backtracking.

The remainder of this paper is organized in five sections. Section 2 proposes two MIP formulations, which are capable of generating tight lower bounds. This section also theoretically demonstrates the relationship between these two formulations. Section 3 first details our optimization framework as applied to generic mixed integer programs, and then discusses its application to the ML-CLSB. Section 4 discusses the results of computational experiments conducted on a number of previously published data sets via a comparison with other state-of-the-art techniques. Finally, Section 5 concludes and suggests directions for future research.

## 2. Problem Formulations

We make the following assumptions in defining and formulating the ML-CLSB. First, we assume that setup times and costs are non-sequence dependent, setup carryover between periods is not permitted, and all initial inventories are zero. Second, all production costs are assumed to be linear in production output and do not vary over time; hence, they can be dropped from the model for simplicity. Setup and holding costs also are assumed not to vary over time. Furthermore, end items are assumed to have no successors, and only end items have external demands and backlogging costs. Finally, we assume zero lead times and no lost sales. It is important to note that all these assumptions (except setup carryover) are made for ease of exposition only and without loss of generality, i.e., the theoretical results remain valid even when they are removed. See Oztürk and Ornek (2010) for the lot-sizing problem with setup carryover as well as with external demands for component items.

To present the problem formulations, we define the following notation:

**Indices and index sets**:

| | |
|---|---|
| $T$ | number of time periods in the planning horizon. |
| $M$ | number of production resources or machines. |
| $I$ | number of items (subassemblies and/or end items). |
| $endp$ | set of end items. |
| $i, j$ | item indices, $i, j \in [1, I]$. |
| $endp_i$ | set of end items that utilize subassembly item $i$. |
| $q, p, t, \ell$ | time period indices, $q, p, t, \ell \in \{1, \ldots, T\}$. |
| $m$ | machine index, $m = 1, \ldots, M$. |
| $\eta_i$ | set of immediate successors of item $i$. |

**Parameters**:

$sc_i$ setup cost for producing a lot of item $i$.
$bc_i$ backlogging cost for one unit of item $i$ for one period.
$hc_i$ inventory holding cost for one unit of item $i$ remaining at the end of a period.
$ec_i$ echelon inventory holding cost for one unit of item $i$ remaining at the end of a period.
$st_{im}$ setup time required for producing item $i$ on machine $m$.
$a_{im}$ production time required to produce one unit of item $i$ on machine $m$.
$gd_{it}$ gross demand for item $i$ in period $t$.
$gd_{itp}$ total gross demand for item $i$ from period $t$ to period $p$.
$ed_{it}$ echelon demand for item $i$ in period $t$.
$ed_{itp}$ total echelon demand for item $i$ from period $t$ to period $p$.
$r_{ij}$ number of units of item $i$ needed to produce one unit of item $j$, where item $j$ is one of the successors of item $i$, but not necessarily an immediate successor.
$C_{mt}$ available capacity of machine $m$ in period $t$.

**Variables**:

$x_{it}$ number of units of item $i$ produced in period $t$.
$s_{it}$ inventory of item $i$ at the end of period $t$.
$e_{it}$ echelon inventory of item $i$ at the end of period $t$.
$b_{it}$ backlogging level for item $i$ in period $t$.
$y_{it}$ binary setup decision variables, ($y_{it} = 1$ if production is setup for item $i$ in period $t$ and 0 otherwise).
$u_{itp}$ number of units of item $i$ produced in period $t$ to satisfy demand in period $p$.
$w_{itp}$ percentage of item $i$ production in period $t$ used to satisfy the accumulated demand for item $i$ from period $t$ to period $p$.

## 2.1. Inventory and Lot-sizing Formulation

The inventory and lot-sizing formulation for lot-sizing problems was originally proposed by Billington et al. (1983), and was presented by Akartunalı and Miller (2009) for the ML-CLSB. In this paper, we refer to this formulation as the inventory and lot sizing (ILS) formulation, which is written as follows.

**ILS:**

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T} sc_i \cdot y_{it} + \sum_{i=1}^{I}\sum_{t=1}^{T} hc_i \cdot s_{it} + \sum_{i\in endp}\sum_{t=1}^{T} bc_i \cdot b_{it} \tag{1}$$

Subject to:

$$x_{it} + s_{i,t-1} + b_{it} - b_{i,t-1} = gd_{it} + s_{it} \qquad \forall\, i \in endp,\ t \in [1,T]. \tag{2}$$

$$x_{it} + s_{i,t-1} = gd_{it} + \sum_{j\in\eta_i} r_{ij} \cdot x_{jt} + s_{it} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{3}$$

$$\sum_{i=1}^{I} a_{im} \cdot x_{it} + \sum_{i=1}^{I} st_{im} \cdot y_{it} \le C_{mt} \qquad \forall\, m \in [1,M],\ t \in [1,T]. \tag{4}$$

$$x_{it} \le \min\left\{\frac{C_{mt} - st_{im}}{a_{im}}, \sum_{j\in endp_i} r_{ij} \cdot gd_{j1T}\right\} \cdot y_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{5}$$

$$s_{i0} = 0,\ s_{it} \ge 0,\ b_{it} \ge 0,\ b_{iT} = 0,\ x_{it} \ge 0,\ y_{it} \in \{0,1\} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{6}$$

In this formulation, the objective function minimizes total setup, backlogging, and holding costs over the planning horizon. Constraints (2) and (3) ensure demand satisfaction in all periods for end and non-end

items, respectively, where the bill-of-materials (BOM) structure dictates the relationships among different levels of items via the definition of immediate successor and predecessor sets. Constraints (4) enforce obeying capacity restrictions. Constraint set (5) ensures that no production occurs for item $i$ in period $t$ unless the corresponding binary setup variable, $y_{it}$, takes a value of 1, in which case the amount of production is limited by a large value depending on the minimum of production time capacity and total demand. Constraints (6) enforce the binary and nonnegativity requirements for variables. In this constraint set, $b_{iT} = 0$ indicates that no backlogging is allowed in the final period. However, a minor change can be applied here to allow such backlogging by requiring $b_{iT} \geq 0$.

### 2.2. Echelon Inventory and Lot-sizing Formulation

Akartunalı and Miller (2009) presented an alternative formulation for this problem. In this formulation, three new sets of echelon variables and parameters, $ed_{it}$, $ec_{it}$, and $e_{it}$ are introduced, where $ed_{it}$ are echelon demands, $ec_i$ are echelon inventory holding cost coefficients, and $e_{it}$ are echelon stock variables for all $i$ and $t$. The relationships between these echelon variables and parameters and their original counterparts can be defined as follows.

$$ed_{it} = gd_{it},\ ec_i = hc_i,\ e_{it} = s_{it} \qquad \forall\, i \in endp,\ t \in [1,T].$$

$$ed_{it} = gd_{it} + \sum_{j \in \eta_i} r_{ij} \cdot ed_{jt} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T].$$

$$ec_i = hc_i - \sum_{i \in \eta_j} r_{ji} \cdot hc_j \qquad \forall\, i \in [1,I] \setminus endp.$$

$$e_{it} = s_{it} + \sum_{j \in \eta_i} r_{ij} \cdot e_{jt} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T].$$

Using these relationships, the alterative formulation, referred to as the echelon inventory and lot-sizing (EILS) model formulation, can be written as follows.

**EILS:**

$$\min \sum_{i=1}^{I} \sum_{t=1}^{T} sc_i \cdot y_{it} + \sum_{i=1}^{I} \sum_{t=1}^{T} ec_i \cdot e_{it} + \sum_{i \in endp} \sum_{t=1}^{T} bc_i \cdot b_{it} \tag{7}$$

Subject to:

$$x_{it} + e_{i,t-1} + b_{it} - b_{i,t-1} = ed_{it} + e_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{8}$$

$$e_{it} \geq \sum_{j \in \eta_i} r_{ij} \cdot e_{jt} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{9}$$

$$\sum_{i=1}^{I} a_{im} \cdot x_{it} + \sum_{i=1}^{I} st_{im} \cdot y_{it} \leq C_{mt} \qquad \forall\, m \in [1,M],\ t \in [1,T]. \tag{10}$$

$$x_{it} \leq \min \left\{ \frac{C_{mt} - st_{im}}{a_{im}}, \sum_{j \in endp_i} r_{ij} \cdot gd_{j1T} \right\} \cdot y_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{11}$$

$$e_{i0} = 0,\ e_{it} \geq 0,\ b_{it} \geq 0,\ b_{iT} = 0,\ x_{it} \geq 0,\ y_{it} \in \{0,1\} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{12}$$

The objective function here is slightly different when compared to the ILS objective function. That is, the second term now uses echelon inventory costs. Using echelon inventory variables, constraints (8) ensure

demand satisfaction for all items over the entire horizon. Constraints (9) require that echelon inventories for non-end items are at least as great as the echelon inventories of their corresponding successor items. Constraints (10) and (11) are the same as constraints (4) and (5) in ILS, while constraints (12) enforce the binary and nonnegativity requirements for variables.

The required model sizes corresponding to the ILS and EILS formulations are relatively modest. However, the time required for proving the optimality of a given solution is often prohibitive, because the integrality gap associated with the LP relaxation is typically large. Moreover, the relatively poor lower bounds provided by the LP relaxation are typically not adequate to guide the search for good feasible solutions in the branch-and-bound tree. Consequently, we propose two new formulations that are able to provide stronger lower bounds.

### 2.3. The Simplified Facility Location Reformulation

The first formulation we propose is called the simplified facility location formulation (SFL), based on the classical facility location (FL) problem formulation. Such a formulation approach for single-stage lot-sizing problems was originally proposed by Krarup and Bilde (1977). Stadtler (2003) later augmented the formulation, and then proposed a simple plant location (SPL) formulation for capacitated multi-level lot-sizing problems with setup times. Formulating SPL requires introducing a new set of variables, $u_{itp}$ (the number of units of item $i$ produced in period $t$ to satisfy demand in period $p$). While SFL is similar to the SPL formulation (Stadtler 2003), there are three significant differences between these formulations.

First, the inventory variables used in SPL can be eliminated by taking advantage of the relationships between the $u$ and $e$ variables; SFL therefore uses fewer variables, because the inventory variables are not explicitly required. Second, because backlogging is permitted in the SFL, the period index $p$ for the variables $u_{itp}$ may run from 1 to $T$ as opposed to the SPL formulation, which contains only $u_{itp}$ variables such that $p \geq t$. Finally, a new constraint set must be added to enforce that the echelon backlog levels for non-end-items correspond to the backlog level of relevant end-items. In SFL, the relationships between the variables $u_{iqp}$, $x_{it}$, $e_{it}$, and $b_{it}$ are written as follows:

$$x_{it} = \sum_{p=1}^{T} u_{itp} \qquad \forall\, i \in [1, I],\ t \in [1, T].$$

$$e_{it} = \sum_{q=1}^{t} \sum_{p=t+1}^{T} u_{iqp} \qquad \forall\, i \in [1, I],\ t \in [1, T].$$

$$b_{it} = \sum_{q=t+1}^{T} \sum_{p=1}^{t} u_{iqp} \qquad \forall\, i \in endp,\ t \in [1, T].$$

$$\sum_{j \in endp_i} r_{ij} \cdot b_{jt} = \sum_{q=t+1}^{T} \sum_{p=1}^{t} u_{iqp} \qquad \forall\, i \in [1, I] \setminus endp,\ t \in [1, T].$$

Here, the first three equations are quite straightforward, and the last equation ensures that echelon backlog levels of non-end-items are correctly related to the backlog levels of corresponding end-item. Using these relationships, we can substitute $x$, $e$, and $b$ with $u$ into (7), (8), (9), (10), and (12) in the EILS formulation. The resulting SFL formulation is then as follows:

**SFL:**

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T} sc_i \cdot y_{it} + \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{p=t}^{T} ec_i \cdot (p-t) \cdot u_{itp} + \sum_{i\in endp}\sum_{t=1}^{T}\sum_{p=1}^{t-1} bc_i \cdot (t-p) \cdot u_{itp} \tag{13}$$

Subject to:

$$\sum_{p=1}^{T} u_{ipt} = ed_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{14}$$

$$\sum_{q=1}^{t}\sum_{p=t+1}^{T} u_{iqp} \geq \sum_{j\in\eta_i} r_{ij} \cdot \sum_{q=1}^{t}\sum_{p=t+1}^{T} u_{jqp} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{15}$$

$$\sum_{j\in endp_i} r_{ij} \cdot \sum_{q=t+1}^{T}\sum_{p=1}^{t} u_{jqp} = \sum_{q=t+1}^{T}\sum_{p=1}^{t} u_{iqp} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{16}$$

$$u_{itp} \leq ed_{ip} \cdot y_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T],\ p \in [1,T]. \tag{17}$$

$$\sum_{i=1}^{T}\sum_{p=1}^{T} a_{im} \cdot u_{itp} + \sum_{i=1}^{I} st_{im} \cdot y_{it} \leq C_{mt} \qquad \forall\, m \in [1,M],\ t \in [1,T]. \tag{18}$$

$$u_{itp} \geq 0,\ y_{it} \in \{0,1\} \qquad \forall\, i \in [1,I],\ t \in [1,T],\ p \in [1,T]. \tag{19}$$

In the above SFL formulation, constraints (14) ensure demand satisfaction in all periods for all items. Constraints (15) ensure that echelon inventories for non-end-items are as large as those of their corresponding successors. Constraints (16) require that echelon backlog levels for non-end-items are consistent with the backlog levels of their corresponding end items. Constraints (17) correspond to setup forcing constraints, while constraint set (18) enforces production capacity limits. Finally, constraints (19) enforce the binary and nonnegativity requirements for variables.

Obviously there is a strong relationship between the EILS formulation and the SFL formulation. The significant advantage of the SFL formulation lies in the fact that the new variables $u_{itp}$ can enforce a stronger relationship between production levels and setups. This is because the disaggregated constraints (17) in SFL are stronger than constraints (11) in the EILS with respect to their LP relaxations. Consequently, SFL can provide better LP relaxation lower bounds than EILS. To illustrate this, observe that if we replace constraints (17) in SFL with another set of constraints obtained by substituting $x$ with the proper sum of $u$ variables in (11), we can easily show that the EILS formulation and the resulting SFL formulation provide exactly the same lower bounds.

### 2.4. The Simplified Shortest Path Reformulation

The second formulation we propose is related to a shortest path (SP) formulation approach. This new formulation introduces variables $w_{itp}$ (the percentage of item $i$ production in period $t$ used to satisfy the accumulated demand for item $i$ from period $t$ to period $p$). Such a formulation was originally proposed by Eppen and Martin (1987) for capacitated multi-item problems without backlogging. We extend this approach to formulate the ML-CLSB. As before, due to the existence of backlogging, the period index $p$ is not constrained to be greater than or equal to the period index $t$ for variables $w_{itp}$. The respective $w_{itp}$ and $u_{itp}$ variables, used in the SFL and simplified shortest path (SSP) formulations, obey the following relationships:

$$u_{ipt} = \sum_{q=t}^{T} w_{ipq} \cdot ed_{it} \qquad \forall\, i \in [1,I],\ p \in [1,T],\ t \in [p,T].$$

$$u_{ipt} = \sum_{q=1}^{t} w_{ipq} \cdot ed_{it} \qquad \forall\, i \in [1,I],\ p \in [1,T],\ t \in [1,p-1].$$

$$\sum_{p=t}^{T} u_{itp} = \sum_{p=t}^{T} w_{itp} \cdot ed_{itp} \qquad \forall\, i \in [1,I],\ t \in [1,T].$$

$$\sum_{p=1}^{t-1} u_{itp} = \sum_{p=1}^{t-1} w_{itp} \cdot ed_{i,t-1,p} \qquad \forall\, i \in [1,I],\ t \in [1,T].$$

Note that the first two equations here can be combined and written as a single equation, simply as $u_{ipt} = \sum_{q=1}^{T} w_{ipq} \cdot ed_{it}$. By substituting for the $u_{itp}$ variables in the SFL reformulation using the corresponding $w_{itp}$ variables, we obtain the following SSP reformulation.

**SSP:**

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T} sc_i \cdot y_{it} + \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{p=t}^{T}\left\{\sum_{q=t}^{p} ec_i \cdot (q-t) \cdot ed_{iq}\right\} \cdot w_{itp} + \sum_{i \in endp}\sum_{t=1}^{T}\sum_{p=1}^{t-1}\left\{\sum_{q=p}^{t} bc_i \cdot (t-q) \cdot ed_{iq}\right\} \cdot w_{itp} \tag{20}$$

Subject to:

$$\sum_{p=1}^{t}\sum_{q=t}^{T} w_{ipq} + \sum_{p=t+1}^{T}\sum_{q=1}^{t} w_{ipq} = 1 \qquad \forall\, i \in [1,I],\ t \in [1,T]. \tag{21}$$

$$\sum_{q=1}^{t}\sum_{p=t+1}^{T}\sum_{\ell=p}^{T} w_{iq\ell} \cdot ed_{ip} \geq \sum_{j \in \eta_i} r_{ij} \cdot \sum_{q=1}^{t}\sum_{p=t+1}^{T}\sum_{\ell=p}^{T} w_{jq\ell} \cdot ed_{jp} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{22}$$

$$\sum_{j \in endp_i} r_{ij} \cdot \sum_{q=t+1}^{T}\sum_{p=1}^{t}\sum_{\ell=1}^{p} w_{jq\ell} \cdot ed_{jp} = \sum_{q=t+1}^{T}\sum_{p=1}^{t}\sum_{\ell=1}^{p} w_{iq\ell} \cdot ed_{ip} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T]. \tag{23}$$

$$\sum_{q=p}^{T} w_{itq} \leq y_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T],\ p \in [t,T]. \tag{24}$$

$$\sum_{q=1}^{p} w_{itq} \leq y_{it} \qquad \forall\, i \in [1,I],\ t \in [1,T],\ p \in [1,t-1]. \tag{25}$$

$$\sum_{i=1}^{I} a_{im} \cdot \left\{\sum_{p=1}^{t-1} w_{itp} \cdot ed_{i,t-1,p} + \sum_{p=t}^{T} w_{itp} \cdot ed_{itp}\right\} + \sum_{i=1}^{I} st_{im} \cdot y_{it} \leq C_{mt} \qquad \forall\, m \in [1,M],\ t \in [1,T]. \tag{26}$$

$$w_{itp} \geq 0,\ y_{it} \in \{0,1\} \qquad \forall\, i \in [1,I],\ t \in [1,T],\ p \in [1,T]. \tag{27}$$

In this formulation, constraints (21) ensure demand satisfaction for all items over the entire horizon. Constraints (22) ensure that echelon inventories for non-end items are at least as large as those of their corresponding successor items. Constraints (23) enforce the relationship between the echelon backlog levels for non-end items and those of their corresponding end items. Constraints (24) and (25) are setup forcing constraints, and constraints (26) enforce capacity limits. Finally, constraints (27) enforce the binary and nonnegativity requirements for different variables.

## 2.5. Equivalence of SFL and SSP

In this section, we will show that SFL and SSP yield the same LP relaxation lower bounds. Before presenting the main result of this section, we first provide some important definitions for clarity of presentation.

**Definition 1** *The notation* $\{(y,u)|(14)-(19)\}$ *corresponds to the set of vectors* $(y,u)$ *that satisfy the constraints* $(14)-(19)$. *The decision problem* $\min\{(13)|(y,u)\in X\}$ *represents the minimization of the function* (13) *among all the vectors* $(y,u)$ *of the set* $X$. *The superscript LP denotes the LP relaxation of a feasible region or a problem, e.g.,* $X^{LP}$ *refers to the set* $X$ *with integrality requirements relaxed.*

Next, we define the feasible sets associated with each formulation as well as the corresponding decision problem:

**Definition 2** *The feasible sets and decision problems considered are as follows:*

$$X_{SFL}=\{(y,u)|(14)-(19)\} \qquad Z_{SFL}=\min\{(13)|(y,u)\in X_{SFL}\}$$
$$X_{SSP}=\{(y,w)|(21)-(27)\} \qquad Z_{SSP}=\min\{(20)|(y,w)\in X_{SSP}\}$$

**Definition 3** Let $\mathbb{A}\subseteq\{(x,z)\in\mathbb{R}^n\times\mathbb{R}^m\}$; the projection of $\mathbb{A}$ onto the space $x\in\mathbb{R}^n$, denoted by $proj_x(\mathbb{A})$, is $\{x\in\mathbb{R}^n:\exists z\in\mathbb{R}^m,(x,z)\in\mathbb{A}\}$.

In the above definition, $\mathbb{A}$ is a subset of the Euclidean space, including polyhedra and mixed-integer sets, and $\mathbb{A}$ may correspond to any of the previously defined feasible regions, e.g., $\mathbb{A}$ might correspond to $X_{SFL}$. Then, the projection of $X_{SFL}$ onto the space of the SSP formulation is denoted by $proj_{y,w}(X_{SFL})$.

**Definition 4** Letting $(\kappa)$ denote an index corresponding to a set of constraints, we let c.$(\kappa)$ denote the corresponding constraints $(\kappa)$.

Next, we present the main result of this section. Note that $Z^{LP}_{SFL}$ and $Z^{LP}_{SSP}$ are the LP relaxations of $Z_{SFL}$ and $Z_{SSP}$, respectively.

**Theorem 1** $Z^{LP}_{SFL}=Z^{LP}_{SSP}$.

*Proof.* This theorem can be proved by showing that $proj_{y,w}(X^{LP}_{SFL})\subseteq X^{LP}_{SSP}$ and $proj_{y,u}(X^{LP}_{SSP})\subseteq X^{LP}_{SFL}$.

In order to show that $proj_{y,w}(X^{LP}_{SFL})\subseteq X^{LP}_{SSP}$, we first consider any point $(y^*,u^*)\in X^{LP}_{SFL}$. Using the defined relationships between $u^*$ and $w^*$, we know the following equations are valid.

$$u^*_{ipt}=\sum_{q=t}^{T}w^*_{ipq}\cdot ed_{it} \qquad \forall\, i\in[1,I],\ p\in[1,T],\ t\in[p,T].$$

$$u^*_{ipt}=\sum_{q=1}^{t}w^*_{ipq}\cdot ed_{it} \qquad \forall\, i\in[1,I],\ p\in[1,T],\ t\in[1,p-1].$$

Given that $(y^*,u^*)\in X^{LP}_{SFL}$, the following constraints are valid from constraints (14).

$$\sum_{p=1}^{T}u^*_{ipt}=ed_{it} \qquad \forall\, i\in endp,\ t\in[1,T].$$

By projecting these constraints onto the space of SSP, we have:

$$\sum_{p=1}^{t}\sum_{q=t}^{T}w^*_{ipq}\cdot ed_{it}+\sum_{p=t+1}^{T}\sum_{q=1}^{t}w^*_{ipq}\cdot ed_{it}=ed_{it} \qquad \forall\, i\in[1,I],\ t\in[1,T].$$

As a consequence, we know $(y^*, w^*) \in c.(21)$ when $(y^*, u^*) \in X^{LP}_{SFL}$. Similarly, we know the following constraints are true from constraints (15):

$$\sum_{q=1}^{t}\sum_{p=t+1}^{T} u^*_{iqp} \geq \sum_{j\in\eta_i} r_{ij} \cdot \sum_{q=1}^{t}\sum_{p=t+1}^{T} u^*_{jqp} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T].$$

By projecting these constraints onto the space of SSP, we have:

$$\sum_{q=1}^{t}\sum_{p=t+1}^{T}\sum_{\ell=p}^{T} w^*_{iq\ell} \cdot ed_{ip} \geq \sum_{j\in\eta_i} r_{ij} \cdot \sum_{q=1}^{t}\sum_{p=t+1}^{T}\sum_{\ell=p}^{T} w^*_{jq\ell} \cdot ed_{jp} \qquad \forall\, i \in [1,I] \setminus endp,\ t \in [1,T].$$

Consequently, we know $(y^*, w^*) \in c.(22)$. Using the relationships between $u^*$ and $w^*$, we can also easily show that $(y^*, w^*) \in c.(23) \cap c.(24) \cap c.(25) \cap c.(27)$.

To show that $(y^*, w^*)$ is valid for constraints c.(26) in SSP, we apply the following relationships:

$$\sum_{p=t}^{T} u^*_{itp} = \sum_{p=t}^{T} w^*_{itp} \cdot ed_{itp} \qquad \forall\, i \in [1,I],\ t \in [1,T].$$

$$\sum_{p=1}^{t-1} u^*_{itp} = \sum_{p=1}^{t-1} w^*_{itp} \cdot ed_{i,t-1,p} \qquad \forall\, i \in [1,I],\ t \in [1,T].$$

Using these relationships, we then project c.(18) onto the space of SSP, and we can easily see that $(y^*, w^*) \in$ c.(26). Thus far, we have shown $(y^*, w^*) \in X^{LP}_{SSP}$ given that $(y^*, u^*) \in X^{LP}_{SFL}$, and therefore $proj_{y,w}(X^{LP}_{SFL}) \subseteq X^{LP}_{SSP}$.

Next, we want to show $proj_{y,u}(X^{LP}_{SSP}) \subseteq X^{LP}_{SFL}$. It is easy to note that the above proof process is invertible. That is, if we let $(y^*, w^*) \in X^{LP}_{SSP}$, and then project all constraints within $X^{LP}_{SSP}$ onto the space of SFL, we can show that $(y^*, u^*) \in X^{LP}_{SFL}$ and $proj_{y,u}(X^{LP}_{SSP}) \subseteq X^{LP}_{SFL}$. This completes the proof. □

A related paper by the first author (Wu 2011) provided theoretical proof demonstrating that the LP relaxations of SFL and SSP provide at least the same or tighter bounds when compared with EILS; we also performed an extensive number of computational tests. Computational results showed that SFL can increase the LP lower bound by 1.0% on average when compared with EILS, and can improve upper bounds by almost 30% on average (the degree of improvement is generally greater for test problems with setup times and high capacity utilization) when the branch-and-bound method is applied to the two formulations. Though both SFL and SSP have advantages over EILS, based on our computational tests for a large number of test instances, on average, solving SFL requires about 85% of the time required for solving SSP.

Therefore, we use the SFL formulation in our implementation of the LugNP solution approach, which we next provide in detail.

## 3. Lower and Upper Bound Guided Nested Partitions

In this section we first present the details of the Lower and upper bound guided Nested Partitions (LugNP) method for mixed integer programming optimization, which includes the optimization of the capacitated lot-sizing problem. Then we will discuss the specific implementation of the framework for the ML-CLSB.

We consider problems of the following form:

$$\text{Minimize} \quad c_1 x + c_2 y$$

$$\begin{aligned} \text{Subject to:} \quad & A_1 x + A_2 y \geq b \\ & x \in \mathbb{R}^n, \;\; y \in \{0,1\}^m \end{aligned} \tag{$\mathcal{P}$}$$

In problem $\mathcal{P}$, $x$ is a vector of real variables, $y$ is a vector of binary variables ($y$ can also be represented as $y_q$ where $q = \{1, \ldots, m\}$), and $c_1$, $c_2$, $A_1$ and $A_2$ are given parameters (or parameter vectors/matrices). The above formulation can be slightly altered for presenting generic MIP problems, i.e., the objective may correspond to a maximization, $y$ may be a vector of general integer variables (instead of only binary variables), and the symbol "$\geq$" in the constraints may be either "$=$" or "$\leq$". Without loss of generality, the above formulation is used for the presentation in this paper. In practice, the presentation of problem $\mathcal{P}$ is simple, but the problem is often considerably difficult to solve due to a large number of variables and constraints. More than likely, difficulties in solving the problem arise as a result of the existence of binary variables $y$.

### 3.1. Introduction to the LugNP Method

The motivation behind the LugNP method is to find an efficient way to fix a subset of the binary variable vector $y$ that leads to a restricted version of problem $\mathcal{P}$ that is easier to solve and enables quickly finding high-quality feasible solutions. The starting point of this method is a candidate feasible solution $y$ that can be quickly obtained using some heuristic or exact method within a limited solution time. For simplicity of presentation, we write a candidate feasible solution as $\overline{y}$ or $\overline{y}_q$ (for $q = \{1, \ldots, m\}$), and let $Q$ denote some subset of $\{1, \ldots, m\}$. In the LugNP method, given a solution $\overline{y}$ and some subset $Q$, the variables $y_q$ for $q \in Q$ are then temporarily fixed to $\overline{y}_q$, while the remaining variables $y_q$ ($q \in \{1, \ldots, m\} \setminus Q$) are treated as binary variables. This approach creates a restricted subproblem that contains fewer binary variables and is therefore easier to solve.

To more clearly describe a subproblem in LugNP, we introduce four subsets of the binary variables $y$ ($qf$, $qp$, $qs$ and $qm$). The subset $qf$ defines an index set of binary variables that have been fixed based on the results of previous iteration(s), $qp$ defines an index set of binary variables that are used for partitioning the feasible region, and $qs$ defines an index set of sampled binary variables. The union of these three subsets defines the subset $Q$ described above, i.e., $qf \cup qp \cup qs = Q$. The remaining subset $qm$ defines the index set of all remaining binary variables. The union of $qf$, $qp$, $qs$ and $qm$ thus corresponds to the full set of binary variables in the original problem; that is, $qf \cup qp \cup qs \cup qm = \{1, \ldots, m\}$.

With these definitions, and given subsets $qf$, $qp$, $qs$, and $qm$, the corresponding subproblem, $\mathcal{P}^S$, can be defined as follows.

$$\begin{aligned} \text{Minimize} \quad & c_1 x + \sum_{p=1}^{m} c_{2p} y_p \\ \text{Subject to:} \quad & A_1 x + \sum_{p=1}^{m} A_{2p} y_p \geq b \\ & y_p = \overline{y}_p, \;\; p \in qf \cup qp \cup qs \\ & y_p \in \{0,1\}, \;\; p \in qm \\ & x \in \mathbb{R}^n, \end{aligned} \tag{$\mathcal{P}^S$}$$

In each iteration of the method we assume that we have a region, i.e., a subset of $\Theta$ (the entire feasible region) that is considered the most promising, which is defined by the fixed variables in the set $qf$. We partition this most promising region into $S$ subregions by randomly selecting binary variables in $\{1,\ldots,m\} \setminus qf$ as partitioning variables, i.e., as elements of $qp$ that will be fixed to the values they take in the candidate feasible solution $\overline{y}$. Each selection of $qf \cup qp$ thus determines a subregion of the promising region, and we create $S$ such subregions. All remaining surrounding regions are then aggregated into one additional "surrounding" subregion. At each iteration, we therefore look at $S+1$ disjoint subsets (or subregions) of the feasible region. Within each of the $S$ promising subregions, we create a number of subproblems of the form $\mathcal{P}^S$, where each of these is obtained by randomly selecting a set of sampling variables $qs$ from among the elements of $\{1,\ldots,m\} \setminus qf \cup qp$; these randomly selected variables are also fixed to the values they take in the candidate feasible solution $\overline{y}$. In the surrounding subregion, restricted subproblems are also sampled using a random sampling scheme (where the restricted subproblem is defined by the variables whose values are fixed). Unlike the strategy of fixing variables to $\overline{y}$ in the promising subregions, the values of the variables that are fixed are selected completely at randomly. The objective function values of these randomly selected subproblems are used to calculate a promising index for each of the $S+1$ subregions. This index determines which region is the most promising region in the next iteration. If one of the subregions is found to be the best, this region becomes the most promising region by fixing the binary variables in $qf \cup qp$. If the surrounding region is found to be the best, the algorithm backtracks to a larger region that contains the former most promising region. The new most promising region is then partitioned and sampled in a similar fashion.

Observe that the partitioning and sampling procedures in the promising region only select two subsets ($qp$ and $qs$) of binary variables whose values must be fixed to their corresponding values in the candidate solution $\overline{y}$. The difference between the variables in $qp$ and $qs$ lies in the fact that the former set is generally much smaller than the latter, and the elements of $qp$ serve as candidates to enter $qf$ if the corresponding subproblem leads to a current-best upper bound (or candidate feasible solution). The set $qf \cup qp \cup qs$ defines the variables whose values are fixed in a given subproblem, and the remaining variables comprise the set $qm$, which are free variables in the corresponding restricted *subproblem*.

In the LugNP method, the sets $qf$, $qp$, and $qs$ are initially empty, and $qm$ corresponds to the full set of setup variables in the initial iteration. Subproblem $\mathcal{P}^S$ is, therefore, initially equivalent to $\mathcal{P}$. As the LugNP procedure evolves, more variables are selected and fixed for partitioning in each iteration, and the four subsets of decision variables are updated iteratively until the problem is solved. Next, we will highlight some other important features of the method.

***a*) Domain knowledge guided partitioning and sampling:** To improve the efficiency of the LugNP method, a set of lower bound solutions is also used to guide partitioning and sampling. For simplicity of presentation, the set of lower-bound solutions is defined as $\underline{y}$, which, for example, can be obtained by solving the LP relaxation of subproblem $\mathcal{P}^S$ at the beginning of each iteration. Given $\overline{y}$ and $\underline{y}$, if only one partitioning variable is to be selected, selecting a variable using uniform (equal) probabilities is one possible choice. However, selecting a variable based on individual variable probabilities that depend on $\overline{y}$ and $\underline{y}$ is

likely to lead to a better choice, i.e., the smaller the gap between the variable's value in $\overline{y}$ and $\underline{y}$, the higher the probability of selecting this variable as the partitioning variable. To enforce this priority, we define the function $Pr_p$ corresponding to the likelihood of selecting a binary variable in $qm$ as a partitioning or sampling variable in Equation (28). In this function, $\rho$ is a parameter used to adjust the weights used in sampling and partitioning. We assume $\rho \geq 1$ so that a larger value of $\rho$ leads to a higher probability of selecting a variable with a smaller gap between its corresponding $\overline{y}$ and $\underline{y}$. The value of $\rho$ used for partitioning variables should be larger than the value of $\rho$ used for sampling variables. This ensures the variables with tiny gaps are more likely to be chosen as partitioning variables. We define this probability function as follows.

$$Pr_p = \frac{\rho^{\left[1-\left|\overline{y}_p - \underline{y}_p\right|\right]}}{\rho} / \sum_{p' \in qm} \frac{\rho^{\left[1-\left|\overline{y}_{p'} - \underline{y}_{p'}\right|\right]}}{\rho}, \qquad \forall\, p \in qm. \tag{28}$$

***b*) Flexibility in applying heuristics and exact solution approaches:** The LugNP method offers flexibility in selecting approaches for obtaining candidate solutions $\overline{y}$ and $\underline{y}$. For obtaining $\underline{y}$, standard methods such as LP relaxation, Lagrangian relaxation (LR), column generation (CG), and cutting planes (CP) can be adopted. For obtaining $\overline{y}$, many heuristic algorithms, such as tabu search, relax-and-fix, simulated annealing, and/or genetic algorithms may serve as candidates. For simplicity, we let $H$ denote the heuristic used in the LugNP method, define $H_u$ as the resulting solution upper bound value, and define $H_y$ as the values of the $y$ variables obtained by the heuristic. For example, for a specific subproblem $\mathcal{P}^S$, $H_u(\mathcal{P}^S)$ is the upper bound solution value for this subproblem and $H_y(\mathcal{P}^S)$ is the associated set of variables $y$. Similarly, we let $L$ denote a lower bounding (e.g., LP-based) solution technique, defining $L_o$ as the associated objective function lower bound, and $L_y$ as the corresponding set of $y$ variables.

***c*) Double-checking of promise index:** If we are considering $N_j$ subproblems within the $j^{th}$ $(j \in S+1)$ subregion of an iteration, then solving each of these subproblems may be computationally burdensome if we require obtaining a feasible solution using a method $H$. Therefore, the LugNP method applies a lower bounding method $L$ to first solve these subproblems quickly, leading to a lower bound promise index for each subproblem. The promise index for a subproblem is inversely proportional to the LP relaxation lower bound. In other words, a smaller lower bound results in a better promise index. Within each subregion, the sample problem with the best promise index is then identified for further evaluation by computing an upper bound promise index. Consequently, $S+1$ sample problems are quickly identified for further exploration.

The upper bounding method $H$ is then applied to determine $H_u$ for each of the $S+1$ selected subproblems. Again, the upper bound promise indices are inversely proportional to the corresponding upper bounds, i.e., a smaller $H_u$ results in a greater promise index. Finally, the subproblem with the best upper bound promise index is selected as the most promising sample. If the best sample is from within the current promising region, the values of the partitioning variables in the subproblem are fixed, and these indices are inserted into $qf$. Otherwise, if the best sample is from the surrounding region, a backtracking procedure (described later) is then performed.

***d*) Gradually-updated feasible solution candidates:** In each iteration of the LugNP method, $S+1$ subproblems are solved by the heuristic, possibly obtaining a better solution than the current feasible solution

candidate $\overline{y}$. To provide better guidance for partitioning and sampling at subsequent iterations, $\overline{y}$ is therefore continuously updated when a better candidate solution is found.

***e*) Backtracking rules:** Several alternative backtracking rules may be used in the LugNP method. The following options for backtracking are considered because of their ease of implementation. Backtracking Rule I: Backtrack to the super-region of the currently most promising region. Backtracking Rule II: Backtrack to the entire feasible region. The difference between these two rules can be thought of in terms of memory requirements. If Rule II is used, then the procedure immediately moves out of a region in a single transition. In case of Rule I, on the other hand, moving completely out of a region of depth greater than one requires more than one transition backwards. Therefore, Rule I has greater memory requirements than Rule II, as the former keeps track of more detailed information at each step. In our implementation of the LugNP method, Rule II is used. Note that the feasible solution candidate $\overline{y}$ must be updated to the solution corresponding to the best subproblem in the surrounding region after backtracking.

***f*) Stopping criteria:** There are two options for stopping the algorithm. The first option is to set a time limit for the LugNP method, and the second option is to stop the algorithm when $qf = m$. For both options, the feasible solution candidate $\overline{y}$ at the stopping time is taken as the final solution.

### 3.2. An Illustrative Example

The following example illustrates how the LugNP method works for a generic MIP problem.

**EXAMPLE**. Consider an MIP problem that has five binary variables and one nonnegative real variable.

$$\begin{aligned} \text{Minimize} \quad & 2x + 5y_1 - 6y_2 - 4y_3 + 2y_4 - y_5 \\ \text{Subject to:} \quad & x + 2y_1 - 4y_2 + 3y_3 - 3y_4 + 2y_5 \geq 8.2 \\ & x \geq 0, \quad y_1, y_2, y_3, y_4, y_5 \in \{0, 1\} \end{aligned} \tag{Ex}$$

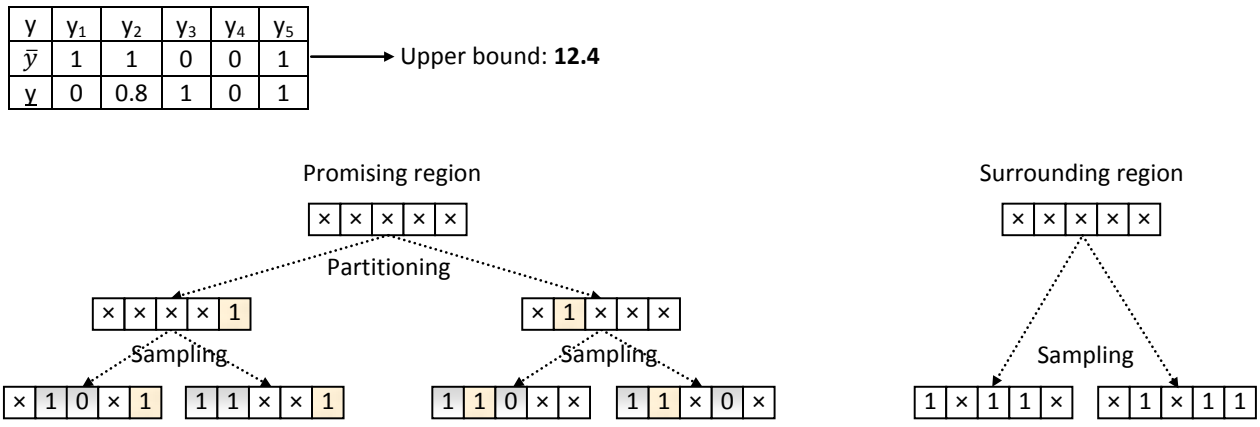


**Figure 1 Partitioning and sampling in the first iteration**

In the first iteration, suppose the feasible solution candidate $\overline{y}$ and lower bound solution $\underline{y}$ shown in Figure 1 have been found in advance. LugNP then relies on these two solutions to perform partitioning and

sampling. As we discussed, the binary variables $y_2$, $y_4$ and $y_5$ are more likely to be chosen as partitioning and sampling variables because they have smaller gaps between $\overline{y}$ and $\underline{y}$. In this example, assume there is only one partitioning variable, and the current most promising region is partitioned into two subregions, $\eta_1 = \{\times, \times, \times, \times, 1\}$ and $\eta_2 = \{\times, 1, \times, \times, \times\}$ ($\times$ indicates the variable is free). In each of these two subregions, suppose there are two sampled subproblems, in which two binary variables are chosen as sampling variables; this leads to four subproblems in the promising region. Let us take the subproblem on the left in Figure 1, for example, which can be described as follows (other subproblems can be constructed similarly).

$$
\begin{aligned}
\text{Minimize} \quad & 2x + 5y_1 - 6*1 - 4*0 + 2y_4 - 1 \\
\text{Subject to:} \quad & x + 2y_1 - 4*1 + 3*0 - 3y_4 + 2*1 \geq 8.2 \\
& y_5 = \overline{y}_5 = 1, \quad \{5\} \in qp \\
& y_2 = \overline{y}_2 = 1, \quad y_3 = \overline{y}_3 = 0, \quad \{2,3\} \in qs \\
& x \geq 0, \quad y_1, y_4 \in \{0,1\}
\end{aligned}
\qquad (Ex^{S1})
$$

To ensure that all points in the feasible region have a possibility of being evaluated, a random sampling procedure is performed in the surrounding region. However, less effort is put on this region, as random sampling is performed directly without first performing partitioning in this region. Let us assume that two sample problems are identified from the surrounding region (see Figure 1). Three binary variables are first chosen randomly as sampling variables and their values are also randomly selected. This results in a total of six subproblems (defined as $\mathcal{P}^{S1}$ to $\mathcal{P}^{S6}$ in Figure 1 from left to the right) in the first iteration.

Let us assume that all of these subproblems and their LP relaxations can be solved to optimality using methods $H$ and $L$, respectively; the lower bounds of these subproblems are 11.4, 8, 12.4, 7, 10.4 and 9.4, respectively. Within each partitioned subregion and in the surrounding region, the subproblem with the best promise index is identified for further evaluation using its upper bound promise index. Consequently, subproblems $\mathcal{P}^{S2}$, $\mathcal{P}^{S4}$ and $\mathcal{P}^{S6}$ are identified for further exploration. After solving $\mathcal{P}^{S2}$, $\mathcal{P}^{S4}$ and $\mathcal{P}^{S6}$ using $H$, their upper bounds are known to be 8, 7 and 9.4, respectively. This indicates that subproblem $\mathcal{P}^{S4}$ has the best upper bound promise index, and this is therefore selected as the most promising subproblem. Because this subproblem comes from within the promising region, the value of partitioning variable $y_2$ in the subproblem is fixed, and its index $\{2\}$ is inserted into $qf$. Subproblem $\mathcal{P}^{S4}$ has a smaller upper bound than the original upper bound. Therefore, the candidate feasible solution $\overline{y}$ must be updated to the upper bound solution of $\mathcal{P}^{S4}$, which is $\{1, 1, 1, 0, 0\}$. At the same time, the lower bound solution for the promising region must also be updated as $y_2$ has been fixed to 1.

In the second iteration, the procedure is similar to the first one. Suppose we now have the six new subproblems shown in Figure 2, and that the fourth subproblem from the left has the lowest upper bound, equal to 3. Therefore, this subproblem is identified as the most promising subproblem. A similar procedure as before would then be performed until meeting one of the stopping criteria. If at any point a subproblem in the surrounding region provides the best upper bound promising index, backtracking must be performed, and the process is repeated with a new candidate feasible solution $\overline{y}$.

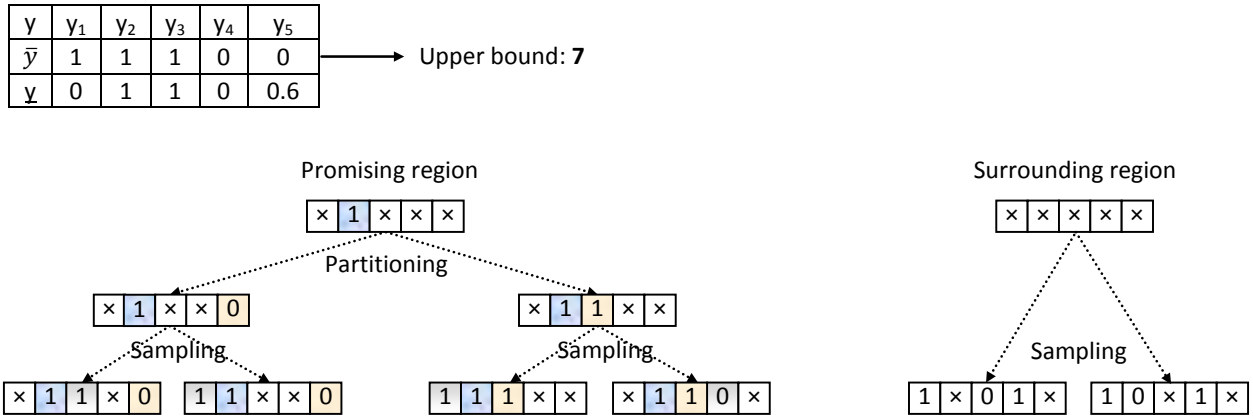


**Figure 2 Partitioning and sampling in the second iteration**

### 3.3. The Lower and Upper Bound Guided Nested Partitions Algorithm

To present a step-by-step description of the LugNP method we require the following additional notation:

$\Theta$ = The feasible region.
$\mathcal{T}_{lim}$ = The algorithm time limit for the framework.
$\overline{y}$ = The best candidate feasible solution.
$\underline{y}$ = The best lower bound solution.
$\sigma(k)$ = The most promising region in the $k^{th}$ iteration.
$\mathcal{P}^S{}_{kji}$ = The $i^{th}$ subproblem in the $j^{th}$ partition of the $k^{th}$ iteration.
$qf$ = The index set containing fixed binary variables.
$qp_{kj}$ = The index set containing partitioning variables in the $j^{th}$ partition of the $k^{th}$ iteration.
$qs_{kji}$ = The index set containing sampling variables in the $i^{th}$ subproblem of the $j^{th}$ partition of the $k^{th}$ iteration.

With all the necessary notation in hand, we can now provide a precise description of the LugNP algorithm.

**Algorithm LugNP:**

**Step-I: Initialization.** Solve $\mathcal{P}$ using $H$ and $L$ to obtain an initial feasible solution candidate $\overline{y}$ and a lower bound solution $\underline{y}$. Set $k=1$, $qf=\emptyset$ and the current most promising region, $\sigma(k)=\Theta$. Go to Step-II.

**Step-II: Partitioning.** Let $S_{\sigma(k)}$ denote the number of subregions of $\sigma(k)$. Partition $\sigma(k)$ into $S_{\sigma(k)}$ subregions $\sigma_1(k)$, ..., $\sigma_{S_{\sigma(k)}}(k)$ by selecting $qp_{kj}$ for each subregion $\sigma_j(k)$, $j=1,...,S_{\sigma(k)}$. Aggregate the surrounding region $\Theta \setminus \sum_{j=1}^{S_{\sigma(k)}} \sigma_j(k)$ into one region $\sigma_{S_{\sigma(k)}+1}(k)$. Go to Step-III.

**Step-III: Sampling.** Let $N_j$ denote the number of subproblems from region $\sigma_j(k)$. Use a sampling procedure to select $qs_{kji}$, $i=1,...,N_j$, in order to construct subproblems from each of the regions $\sigma_j(k)$, $j=1,2,...,S_{\sigma(k)}+1$,; these subproblems are listed as follows:

$$\mathcal{P}^S{}_{kj1},\ \ \mathcal{P}^S{}_{kj2},\ \ ...,\ \ \mathcal{P}^S{}_{kjN_j},\ \ j=1,2,...,S_{\sigma(k)}+1. \tag{29}$$

Go to Step-IV.

**Step-IV: Estimating promising index.** Solve all subproblems using $L$ to obtain lower bounds,

$$L_o(\mathcal{P}^S{}_{kj1}),\ \ L_o(\mathcal{P}^S{}_{kj2}),\ \ ...,\ \ L_o(\mathcal{P}^S{}_{kjN_j}),\ \ j=1,2,...,S_{\sigma(k)}+1. \tag{30}$$

Determine the most promising subproblem within each subregion, where

$$\hat{i}_{kj} = \arg \min_{i \in \{1,\ldots,N_j\}} L_o(\mathcal{P}^S{}_{kji}), \quad j = 1, 2, ..., S_{\sigma(k)} + 1. \tag{31}$$

Note that $\hat{i}_{kj}$ is the index of the most promising subproblem within subregion $\sigma_j(k)$. This results in $S_{\sigma(k)} + 1$ subproblems, which are solved by $H$ in order to obtain upper bounds.

$$H_u(\mathcal{P}^S{}_{kj(\hat{i}_{kj})}), \quad j = 1, 2, ..., S_{\sigma(k)} + 1. \tag{32}$$

Determine the most promising subproblem, where

$$\hat{j}_k = \arg \min_{j \in \{1,\ldots,S_{\sigma(k)}+1\}} H_u(\mathcal{P}^S{}_{kj(\hat{i}_{jk})}). \tag{33}$$

Note that $\hat{j}_k$ is the index of the most promising subproblem in the $k^{th}$ iteration, and $\hat{j}_k$ is also the index of the most promising region to which the most promising subproblem belongs. If two or more regions are equally promising, ties can be broken arbitrarily. If this index corresponds to a region that is a subregion of $\sigma(k)$, then let this serve as the most promising region in the next iteration. Note that in this promising region, $qp_{k(\hat{j}_k)}$ has been inserted into $qf$. Go to Step-V. Otherwise, if the index corresponds to the surrounding region, go to Step-VII.

**Step-V: Updating the feasible solution candidate.** Update $\overline{y}$ to $H_y(\mathcal{P}^S{}_{k(\hat{j}_k)(\hat{i}_{k\hat{j}})})$ if $H_u(\mathcal{P}^S{}_{k(\hat{j}_k)(\hat{i}_{k\hat{j}})})$ is the smallest upper bound so far. Set $k = k + 1$. Go to Step-VI.

**Step-VI: Stopping criteria check.** If $qf = m$ or the solution time exceeds the algorithm time limit $\mathcal{T}_{lim}$, then the algorithm stops. Let $\overline{y}$ be the final solution, otherwise, go to Step-II.

**Step-VII: Backtracking.** Set $qf = \emptyset$, $k = 1$. Update the most promising region to $\Theta$, and $\overline{y}$ to $H_y(\mathcal{P}^S{}_{k(\hat{j}_k)(\hat{i}_{k\hat{j}})})$ if $H_u(\mathcal{P}^S{}_{k(\hat{j}_k)(\hat{i}_{k\hat{j}})})$ is the smallest upper bound so far. Go to Step-II.

### 3.4. Design of the Optimization Framework for the ML-CLSB

This subsection describes the application of the LugNP method to the ML-CLSB problem class. As we noted before, the LugNP framework offers flexibility in selecting approaches for achieving lower and upper bound solutions. For obtaining lower bound solutions, standard methods such as LP relaxation, LR, CG, and CP can be adopted. In the case of ML-CLSB, we directly use the LP relaxation technique ($L$) to obtain the lower bound solutions $\underline{\mathbf{y}}$. For obtaining upper bound solutions, many heuristic algorithms, such as tabu search, relax-and-fix, simulated annealing, and genetic algorithms, can be selected. Because of our focus on lot-sizing problems, we use a relax-and-fix heuristic ($H$) technique to achieve upper bound solutions $\overline{\mathbf{y}}$.

The basic idea of the relax-and-fix algorithm implemented here is to partition all periods in the entire time horizon into three parts. The first part corresponds to a time window that contains several periods, the second includes the periods preceding the time window, and the third covers the periods following the time window. In the early stages of this algorithmic approach, only the lot-sizing problem in the first time window is solved (here, $\alpha$ is defined as the size of the time window). Within this time window, the binary variables in the first few periods are required to be binary, and the binary variables in the remaining periods are relaxed so as to be continuous (here, the number of such remaining periods is denoted as $\gamma$). An MIP

solver is then used for solving these smaller problems (here, the solving time limit is defined as $\mathcal{T}_{rf}$), with fewer complicating binary variables, and the resulting solution is used to fix the binary variables in the first few periods in the time window (here, the size of such periods is defined as $\beta$, with $\beta \leq \alpha$ - $\gamma$). The following periods (from period $\beta+1$ to $\beta + \alpha$) are then processed in the same manner until all binary variables are fixed. In the example shown in Figure 3, $\alpha$ equals 4, $\beta$ equals 2, and $\gamma$ equals 1.

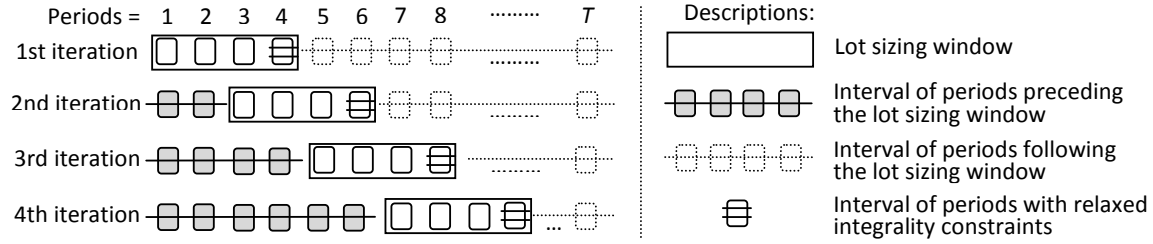


**Figure 3 The relax-and-fix algorithm**

In the case of the ML-CLSB problem class, subproblems in the LugNP framework are expressed as follows.

$$
\begin{aligned}
\text{Minimize} \quad & (13) \\
\text{Subject to:} \quad & (14), (15), (16), (17), (18) \\
& u_{itp} \geq 0 \quad \forall\, i \in [1, I],\ t \in [1, T],\ p \in [t, T]. \qquad (Z^S_{SFL}) \\
& y_{it} = \overline{y}_{it}, \quad it \in qf \cup qp \cup qs \\
& y_{it} \in \{0, 1\}, \quad it \in qm
\end{aligned}
$$

If we replace problem $\mathcal{P}$ with $Z_{SFL}$, replace problem $\mathcal{P}^S$ with $Z^S_{SFL}$, use the LP relaxation technique as the lower bound technique $L$, and use the relax-and-fix algorithm as the upper bound technique $H$, then the LugNP framework expressed above can be implemented to solve the ML-CLSB problem class.

## 4. Computational Results

Our computational experiments were conducted on numerous test instances of different sizes, in order to characterize the performance of LugNP across a wide range of problem instances. Moreover, as noted by Stadtler (2003), "once this problem is solvable for some 100 items over a planning interval of 6-12 months it may well substitute for the current MRP II logic". Some of the test problems we used for computational results are based on problems with sizes in this range or even bigger.

### 4.1. Description of Test Instances

The first group of test data sets we used was generated by Tempelmeier and Derstroff (1996) and Stadtler (2003). These data sets include sets A+, B+, C, and D, where A+ contains 120 test instances, B+ contains 312 test instances, C contains 144 test instances, and D contains 79 test instances. Sets A+ and B+ include problems with 10 items, 24 periods, and 3 machines, while sets C and D include problems with 40 items, 16

periods, and 6 machines. There are no setup times for sets A+ and C, but sets B+ and D include positive setup times. Moreover, these data sets were constructed using a full factorial design with seven factors as follows:

1. *Operations structure*: there are two such settings: general and assembly. Assembly structures have the limitation that each item has only one 'child' item in the BOM, i.e., an item can only be used as a component of one other item. In a general structure, no such limitation exists.

2. *Resource assignment*: there are two such settings, acyclic and cyclic. No item is allowed to use the same machine as one or more of its predecessors for acyclic problems, though such situations are permitted for cyclic problems. Acyclic problems are generally more difficult to solve than the corresponding cyclic problems; we only consider acyclic problems.

3. *Setup times*: there are five such settings denoted by 0, 1, 2, 3, and 4, where 0 indicates that there are no setup times, and 1, 2, 3, and 4 indicate that setup times are required before producing an item.

4. *Coefficient of demand variation*: we consider two settings denoted by 1 and 2, where 1 indicates slight demand variation and 2 indicates sizable variation.

5. *Resource utilization*: there are five such settings denoted by 1, 2, 3, 4, and 5, where 1 represents high utilization, 2, 4, and 5 represent medium utilization, and 3 corresponds to low utilization.

6. *TBO*: TBO denotes the time between orders, and we consider three such settings denoted as 2, 3 and 4, where 2 indicates a high TBO, 3 indicates a medium TBO, and 4 corresponds to a low TBO.

7. *Amplitude of seasonal pattern*: we consider three settings denoted by 0, 1, and 2, where 0 indicates no seasonality for item demands, 1 indicates slight seasonality, and 2 indicates strong seasonality.

For more details about these instances, see Tempelmeier and Derstroff (1996) and Stadtler (2003). Originally, these data sets had no allowance for backlogging; we thus slightly alter the problem instances in order to permit backlogging. We use a ratio of backlogging costs to inventory costs such that $bc_i = 10{*}hc_i$ for $i \in endp$. The resulting data sets are referred to as $\overline{A}+$, $\overline{B}+$, $\overline{C}$, and $\overline{D}$, after this modification.

We generated an additional group of data sets based on the above sets, in which, for each test instance, the demand for all items increases by twenty percent for the first half of the time horizon, while the resource capacities increase by ten percent over the time horizon. This new group of data sets was generated so that backlogging plays a more significant role. The resulting data sets are referred to as $\overline{\overline{A}}+$, $\overline{\overline{B}}+$, $\overline{\overline{C}}$, and $\overline{\overline{D}}$.

The third group of test data sets we used was generated by Simpson and Erenguc (2005) and Akartunalı and Miller (2009). These data sets include sets SET1, SET2, SET3, and SET4, each set having 30 instances with low, medium and high variability of demand. This group of data sets was originally generated without backlogging; Akartunalı and Miller (2009) later modified the data sets by adding backlogging costs to them. The backlogging costs are set to twice the inventory holding costs for the first two sets, and 10 times the inventory holding costs for the last two sets. Except for the problems in SET2, which have a horizon of 24 periods, all instances have 16 periods. All instances have 78 items and have an assembly structure, and backlogging is allowed in the last period. For more details about the instances, see Simpson and Erenguc (2005) and Akartunalı and Miller (2009).

### 4.2. Settings for Computational Tests

We compare the LugNP procedure with the heuristic method proposed by Akartunalı and Miller (2009) (denoted by Aheur) and the commercial MIP solver CPLEX 11.2 (B&C) in order to establish its efficiency relative to state-of-the-art methods. The very efficient algorithm proposed by Akartunalı and Miller (2009) is able to obtain excellent results for lot-sizing problems, and is the only algorithm in the extant literature that has been implemented to solve ML-CLSB. The commercial MIP solver, CPLEX 11.2, is one of the most powerful solvers with the branch-and-cut algorithm embedded. This solver is very efficient at solving lot-sizing problems. In order to ensure fair comparisons, all three approaches were implemented on the same SFL model, and the same computing capacity (Intel Pentium 4, 3.16 GHz processor) was used. Each of these approaches was programmed using GAMS, a high-level algebraic modeling language, in which CPLEX 11.2 is called as the solver.

A total computing time of 100 seconds was allocated for instances in sets $\overline{A}+$, $\overline{B}+$, $\overline{\overline{A}}+$, $\overline{\overline{B}}+$, and SET1, and 300 seconds for instances in sets $\overline{C}$, $\overline{D}$, $\overline{\overline{C}}$, $\overline{\overline{D}}$, SET3, and SET4 (because of the complexity). A computing time of 150 seconds was allocated for instances in set SET2, due to the particularly bad results achieved by Aheur when the time was set to 100 seconds. The total time assigned to the three approaches is the same; so this comparison is reasonable and valid. There are many possible parameter settings, even when the total solution time is limited for each of the above approaches. One possible setting that might maximize the performance of the corresponding approach is based on empirical data. First, for LugNP, values of $\alpha$, $\beta$, and $\gamma$ are set to $\lceil 5 + \mathcal{T}_{lim}/300 \rceil$, 2, and 2, respectively, for obtaining the initial upper bound solutions, while the value of $\alpha$ is altered to the total length of the entire horizon when solving the subproblems, $Z^{NP}_{SFL}$. The number of partitioning variables at any iteration is set to 2, and the number of sampling variables at any iteration is set to $\max\left(\frac{I\times T}{2}, |qm| * 0.6\right)$ in the $d$th iteration. $\mathcal{T}_{rf}$ is set to $1 + \mathcal{T}_{lim}/100$ seconds for data sets $\overline{A}+$, $\overline{B}+$, $\overline{\overline{A}}+$, $\overline{\overline{B}}+$, and set to $5 + \mathcal{T}_{lim}/100$ seconds otherwise. The number of partitioned subregions within the promising region at any iteration is set to 10, the number of sample problems in each partitioned subregion at any iteration is set to 3, and the number of sample problems in the surrounding region at any iteration is set to 1. Second, in Aheur, the strategy recommended by Akartunalı and Miller (2009) is applied to set parameter values; details are omitted here. Finally, for the CPLEX solver, the "flow cover" and "Mixed Integer Rounding (MIR)" cuts are activated to improve solution quality.

The computational results in terms of duality gaps are given in the Tables 1-3. The duality gap is calculated as the difference between upper and lower bound values, divided by the lower bound value. For a fair comparison, the lower bound yielded by the LP relaxations of SFL was used for calculating the duality gaps for all three approaches. As the same lower bound is used for all three approaches, the comparison of duality gaps also can be considered as a comparison of cost savings. In these tables, Imp-1 indicates the improvement brought about by LugNP, compared with Aheur, while Imp-2 denotes the improvement compared with B&C. Imp-1 is calculated as (Aheur's duality gap − LugNP's duality gap)/(Aheur's duality gap); Imp-2 is calculated in the same manner. The first two tables provide details of the computational results for different problem characteristics, whereas Table 3 provides a summary. As these results indicate, LugNP is consistently better than the other two benchmarks considered.

**Table 1 Gap comparison of Aheur, B&C, and LugNP for the full factorial experiment of dataset $\overline{A}+$.**

| Factors | Coefficient | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|---|
| Utilization | U1 (High) | 54.91% | 50.09% | 41.71% | 24.05% | 16.74% |
| | U2 (Medium) | 27.44% | 32.61% | 21.79% | 20.57% | 33.16% |
| | U3 (Low) | 12.27% | 14.63% | 10.92% | 11.04% | 25.37% |
| | U4(Medium) | 25.04% | 29.64% | 22.56% | 9.91% | 23.91% |
| | U5(Medium) | 31.38% | 34.24% | 26.83% | 14.48% | 21.63% |
| TBO | TBO3 (Medium) | 32.89% | 33.79% | 27.01% | 17.86% | 20.06% |
| | TBO4 (Low) | 27.53% | 30.69% | 22.51% | 18.23% | 26.66% |
| Demand variance | D1 (Low) | 30.02% | 32.62% | 24.63% | 17.96% | 24.48% |
| | D2 (High) | 30.39% | 31.86% | 24.89% | 18.10% | 21.88% |
| Seasonality | S0 (Low) | 25.91% | 28.64% | 21.48% | 17.10% | 25.01% |
| | S1 (Medium) | 30.64% | 33.40% | 24.25% | 20.86% | 27.40% |
| | S2 (High) | 34.08% | 34.68% | 28.56% | 16.19% | 17.66% |
| | Average | 30.21% | 32.24% | 24.76% | 18.03% | 23.20% |

**Table 2 Gap comparison of Aheur, B&C, and LugNP for the full factorial experiment of dataset $\overline{B}+$.**

| Factors | Coefficient | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|---|
| Setup Time | Setup1 (Low) | 29.07% | 31.84% | 23.52% | 19.12% | 26.14% |
| | Setup2 (Medium) | 28.09% | 36.97% | 23.04% | 17.97% | 37.67% |
| | Setup3 (Medium) | 33.83% | 34.64% | 26.43% | 21.86% | 23.70% |
| | Setup4 (High) | 30.61% | 32.94% | 25.16% | 17.81% | 23.63% |
| Utilization | U1 (High) | 50.78% | 56.33% | 37.46% | 26.23% | 33.50% |
| | U2 (Medium) | 23.67% | 29.03% | 19.59% | 17.24% | 32.51% |
| | U3 (Low) | 10.99% | 12.58% | 9.66% | 12.17% | 23.23% |
| | U4(Medium) | 22.64% | 25.35% | 19.43% | 14.20% | 23.35% |
| | U5(Medium) | 30.29% | 34.73% | 26.06% | 13.98% | 24.97% |
| TBO | TBO3 (Medium) | 32.54% | 35.70% | 25.28% | 22.31% | 29.17% |
| | TBO4 (Low) | 26.00% | 32.75% | 22.35% | 14.01% | 31.73% |
| Demand variance | D1 (Low) | 28.47% | 31.48% | 23.28% | 18.22% | 26.05% |
| | D2 (High) | 30.37% | 37.04% | 24.44% | 19.54% | 34.03% |
| Seasonality | S0 (Low) | 28.48% | 30.37% | 22.21% | 22.04% | 26.87% |
| | S1 (Medium) | 27.55% | 32.20% | 22.16% | 19.56% | 31.20% |
| | S2 (High) | 32.23% | 40.22% | 27.21% | 15.57% | 32.34% |
| | Average | 29.42% | 34.26% | 23.86% | 18.90% | 30.36% |

**Table 3 Gap comparison of Aheur, B&C, and LugNP for datasets $\overline{\overline{A}}+$ and $\overline{\overline{B}}+$.**

| Data Sets | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|
| Set $\overline{\overline{A}}+$ | 28.98% | 30.82% | 24.92% | 13.99% | 19.12% |
| Set $\overline{\overline{B}}+$ | 34.30% | 34.69% | 28.57% | 16.69% | 17.63% |

### 4.3. Computational Results for Test Instances of Medium Size

According to the results shown in Tables 1 and 2, the level of capacity usage has a significant influence on the duality gap for all three approaches. For example, the duality gaps for highly capacitated problems are about four times the gaps for less capacitated problems. In addition, TBO and seasonality also have a significant effect. For example, the duality gaps for problems with medium TBO levels are about 4% higher

than the gaps for problems with low TBO level. Demand variance, on the other hand, has only a slight effect, whereas the differences in duality gaps for problems with high (and low) demand variations are as much as about 2%. Based on our experimental results, these factors, including capacity usage, TBO, seasonality, and demand variance, have a similar degree of influence for data sets $\overline{\overline{A}}+$ and $\overline{\overline{B}}+$ as they do for data sets $\overline{A}+$ and $\overline{B}+$.

From Tables 1, 2, and 3, we can see that LugNP obviously offers solutions superior to the other two methods; LugNP can obtain smaller duality gaps across various parameter settings when compared with Aheur and B&C. The average improvement in duality gaps is about 23% for data sets $\overline{A}+$ and $\overline{B}+$, while the average improvement is about 16% for data sets $\overline{\overline{A}}+$ and $\overline{\overline{B}}+$. More specifically, in the case of highly capacitated lot-sizing problems in data set $\overline{A}+$, the duality gaps obtained by Aheur and B&C are 54.91% and 50.09%, respectively, while the duality gap obtained by LugNP is only about 41.71%. Similar trends can be observed for other parameters also, such as seasonality, where LugNP can improve the duality gap significantly.

### 4.4. Computational Results for Test Instances of Large Size

A comparison of the computational results for data sets $\overline{C}$, $\overline{D}$, $\overline{\overline{C}}$, and $\overline{\overline{D}}$ is given in Tables 4, 5, and 6. It is clear from the results that LugNP can achieve much smaller duality gaps compared with Aheur and B&C, and the improvement is especially obvious for large size instances which are more difficult to solve. Furthermore, in terms of duality gaps, LugNP has a bigger advantage over the other approaches for test instances with setup times than for instances without setup times.

**Table 4 Gap comparison of Aheur, B&C, and LugNP for the full factorial experiment of dataset $\overline{C}$.**

| Factors | Coefficient | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|---|
| Utilization | U1 (High) | 76.19% | 70.23% | 59.29% | 22.18% | 15.58% |
| | U2 (Medium) | 58.74% | 40.69% | 25.97% | 55.79% | 36.18% |
| | U3 (Low) | 22.54% | 19.45% | 11.08% | 50.86% | 43.05% |
| | U4(Medium) | 33.88% | 28.64% | 20.08% | 40.74% | 29.88% |
| TBO | TBO2 (Low) | 80.26% | 66.94% | 44.05% | 45.12% | 34.20% |
| | TBO3 (Medium) | 31.56% | 25.63% | 21.65% | 31.42% | 15.53% |
| | TBO4 (Low) | 31.68% | 26.69% | 21.61% | 31.78% | 19.02% |
| Demand variance | D1 (Low) | 50.71% | 42.26% | 30.03% | 40.78% | 28.94% |
| | D2 (High) | 44.96% | 37.25% | 28.18% | 37.33% | 24.35% |
| Seasonality | S0 (Low) | 41.25% | 39.48% | 24.33% | 41.02% | 38.37% |
| | S1 (Medium) | 47.97% | 38.67% | 29.21% | 39.11% | 24.45% |
| | S2 (High) | 54.28% | 41.12% | 33.77% | 37.79% | 17.87% |
| | Average | 47.84% | 39.75% | 29.10% | 39.16% | 26.79% |

The computational time was limited to 300 seconds for the above tests. In order to determine how the solution qualities of three methods might change as the maximum computational time increases, we further tested all instances with high utilization and low seasonality from within set $\overline{C}$. The allowed time limit, $\mathcal{T}_{lim}$, was then set to 300, 500, 750, 1000, 1500, and 3000 seconds, respectively.

**Table 5 Gap comparison of Aheur, B&C, and LugNP for the full factorial experiment of dataset $\overline{D}$.**

| Factors | Coefficient | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|---|
| Setup time | Setup1 (Low) | 33.34% | 29.16% | 18.64% | 44.08% | 36.08% |
| | Setup4 (High) | 23.55% | 30.29% | 11.03% | 53.14% | 63.58% |
| Utilization | U1 (High) | 67.12% | 50.46% | 32.31% | 51.86% | 35.97% |
| | U2 (Medium) | 19.80% | 29.25% | 10.82% | 45.33% | 63.00% |
| | U3 (Low) | 7.22% | 6.55% | 4.64% | 35.76% | 29.26% |
| | U4(Medium) | 15.83% | 16.26% | 9.47% | 40.19% | 41.77% |
| | U5(Medium) | 32.01% | 46.05% | 16.94% | 47.09% | 63.22% |
| TBO | TBO3 (Medium) | 30.04% | 34.83% | 15.04% | 49.94% | 56.83% |
| | TBO4 (High) | 27.01% | 24.74% | 14.74% | 45.43% | 40.42% |
| Seasonality | S1 (Medium) | 25.28% | 23.37% | 14.06% | 44.38% | 39.84% |
| | S2 (High) | 31.65% | 35.91% | 15.69% | 50.43% | 56.30% |
| | Average | 28.51% | 29.72% | 14.89% | 47.78% | 49.91% |

**Table 6 Gap comparison of Aheur, B&C, and LugNP for datasets $\overline{\overline{C}}$ and $\overline{\overline{D}}$.**

| Data Sets | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|
| Set $\overline{\overline{C}}$ | 46.79% | 40.28% | 29.92% | 36.05% | 25.73% |
| Set $\overline{\overline{D}}$ | 43.31% | 79.95% | 32.73% | 24.43% | 59.06% |

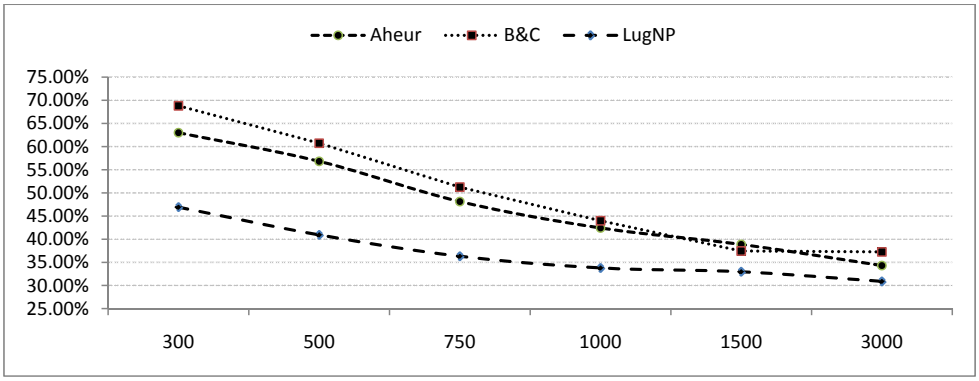


**Figure 4 Gap Comparison of Aheur, B&C, and LugNP**

Figure 4 shows the gaps, on average, of the tested instances. In the figure, the horizonal axis denotes the computational time, and the vertical axis indicates the average gaps achieved by these three methods. As we can see from the figure, all gaps have a stable reduction as the time increases. For example, the solution gap obtained by B&C has been reduced from 69% to 37% as the allowed computational time increases from 300 to 3000 seconds. The same tendency can be observed for the gap obtained by LugNP, but the rate of decrease is comparatively smaller. In addition, the figure shows that the solution quality of LugNP dominates the solution qualities of B&C and Aheur. Figure 4 also shows advantages of LugNP over other approaches in terms of time savings. As we can see, LugNP achieves a 46% duality gap in 300 seconds, while Aheur and B&C take more than 800 seconds to achieve the same level of solution quality. Similarly, LugNP achieves a 36% duality gap in 750 seconds, while Aheur and B&C do not achieve the same level of solution quality even with a computational time of more than 3600 seconds.

### 4.5. Computational Results for Other Test Instances

A comparison of the computational results for data sets SET1, SET2, SET3, and SET4 is given in Table 7. Note that the inventory and lot sizing formulation with backlogging presented by Akartunalı and Miller (2009) was used to test this group of test instances (in which a setup can be used for a family of products instead of only one product, making this formulation different from the ILS formulation presented in this paper).

Compared with the solution results for the previous groups of test instances, the improvement obtained by LugNP is less noticeable for this group of data sets, which we attribute to one significant reason. That is, there are no setup costs for all instances in this group, and setup period decisions are therefore unimportant as compared with the prior test instances we discussed.

**Table 7 Gap comparison of Aheur, B&C, and LugNP for datasets SET1, SET2, SET3, and SET4.**

| Data Sets | Aheur | B&C | LugNP | Imp-1 | Imp-2 |
|---|---|---|---|---|---|
| SET1 | 14.36% | 19.59% | 14.28% | 0.54% | 27.07% |
| SET2 | 8.84% | 12.43% | 8.82% | 0.22% | 28.98% |
| SET3 | 206.04% | 248.13% | 172.69% | 16.18% | 30.40% |
| SET4 | 108.68% | 123.60% | 105.59% | 2.84% | 14.57% |

## 5. Conclusions and Future Research

We proposed two new MIP reformulations for the capacitated multi-level lot-sizing problem with backlogging. Theoretical results show their relationships in terms of achievable lower bounds, and results on their computational efficiency can be used to guide formulation choices for these problems. In addition, we investigated a new optimization framework, LugNP. This method effectively integrates lower and upper bounding techniques within the NP method, incorporating an efficient partitioning and sampling strategy, retaining the search within the most promising region, and leading to comparatively excellent solutions. To demonstrate the solution quality, computational tests based on benchmark problems were performed, which showed that LugNP is superior to two other state-of-the-art approaches.

Future work along this line of research should focus on implementing the LugNP framework for general MIP optimization problems. In particular, due to their similar structures, scheduling problems can benefit from this framework substantially and therefore lie within our current interests. We are also interested in exploring theoretical results on how lower and upper bounds can be used to define intelligent partitioning approaches that improve the efficiency of the LugNP method. The fundamental unsolved problem addressed by this research is to determine how the bounds obtained through MIP and heuristic methods for each region are best utilized by the NP framework. To understand the issues involved, it is helpful to consider the parallel to how such bounds are utilized in branch-and-bound, branch-and-cut, and branch-cut-and-price.

In branch-and-bound and its variants, the goal is to obtain a sufficiently tight bound so that a branch that does not contain the optimal solution can be fathomed (eliminated). Such a strong statement is not needed for the LugNP method. What is needed is for the correct region to be selected with a sufficiently high probability. If this probability does not substantially increase when a bound improves, there is little or no benefit to this improvement. Thus, a less tight bound, which can be obtained faster and which (together with an upper-bound) can identify a correct LugNP move may be the best. Therefore, the basic research questions addressed by this topic are three-fold.

First, how tight does a bound need to be to impose a good structure on the search space through intelligent partitioning? Here we will address the fundamental trade-off: is it better to spend more time tightening a bound, or to use the looser bound and reserve time for more main iterations of the algorithm? Second, how are such bounds best obtained? Here we will investigate embedding standard methods such as LP relaxations, Lagrangian relaxations, column generation, and cutting planes. While the performance of such methods is known within other contexts, it is unknown which approach is best for incorporation within the LugNP framework proposed here. Finally, how sensitive are the preference orders of the partitioning variables to the LugNP framework? In the ML-CLSB, our experience suggests that when there are many variables to be fixed, the selection of variables might impact the convergence rate of the LugNP algorithm. We will investigate this issue and develop theoretical results which can guide the LugNP search effectively.

## References

Agra, A., M. Constantino. 1999. Lotsizing with backlogging and start-ups: the case of wagner-whitin costs. *Operations Research Letters* **25**(2) 81–88.

Akartunalı, K., A. J. Miller. 2009. A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research* **193**(2) 396–411.

Almada-Lobo, B., R. J. W. James. 2010. Neighbourhood search meta-heuristics for capacitated lot-sizing with sequence-dependent setups. *International Journal of Production Research* **48**(3) 861–878.

Barbarosoglu, G., L. Özdamar. 2000. Analysis of solution space-dependent performance of simulated annealing: the case of the multi-level capacitated lot sizing problem. *Computers & Operations Research* **27**(9) 895–903.

Billington, P., J. McClain, L. Thomas. 1983. Mathematical programming approaches to capacity-constrained MRP systems: Review, formulation and problem reduction. *Management Science* **29**(10) 1126–1141.

Cheng, C. H., M. S. Madan, Y. Gupta, S. So. 2001. Solving the capacitated lot-sizing problem with backorder consideration. *Journal of the Operational Research Society* **52**(8) 952–959.

Eppen, G. D., R. K. Martin. 1987. Solving multi-item lot-sizing problems using variable redefinition. *Operations Research* **35**(6) 832–848.

Federgruen, A., M. Tzur. 1993. The dynamic lot-sizing model with backlogging: A simple *O(n log n)* algorithm and minimal forecast horizon procedure. *Naval Research Logistics* **40**(4) 459–478.

Ganas, I., S. Papachristos. 2005. The single-product lot-sizing problem with constant parameters and backlogging: Exact results, a new solution, and all parameter stability regions. *Operations Research* **53**(1) 170–176.

Hindi, K. S. 1995. Solving the single-item, capacitated dynamic lot sizing problem with startup and reservation costs by tabu search. *Computers & Industrial Engineering* **28**(4) 701–707.

Hindi, K. S. 1996. Solving the clsp by a tabu search heuristic. *Journal of the Operational Research Society* **47**(1) 151–161.

Hung, Y. F., C. P. Chen, C. C. Shih, M. H. Hung. 2003. Using tabu search with ranking candidate list to solve production planning problems with setups. *Computers & Industrial Engineering* **45**(4) 615–634.

Karimi, B., S. M. T. F. Ghomiand, J. M. Wilson. 2006. A tabu search heuristic for solving the CLSP with backlogging and set-up carry-over. *Journal of the Operational Research Society* **57**(2) 140–147.

Krarup, J., O. Bilde. 1977. Plant location, set covering and economic lot sizes: An *O(mn)* algorithm for structured problems. *Optimierung bei Graphentheoretischen und Ganzzahligen Probleme* **Birkhauser Verlag, Basel, Switzerland** 155–180.

Küçükyavuz, S., Y. Pochet. 2009. Uncapacitated lot sizing with backlogging: the convex hull. *Mathematical Programming* **118**(1) 151–175.

Kuik, R., M. Salomon. 1990. Multi-level lot-sizing problem: Evaluation of a simulated annealing heuristic. *European Journal of Operational Research* **45** 2537.

Kuik, R., M. Salomon, L. N. van Wassenhove, J. Maes. 1993. Linear-programming, simulated annealing and tabu search heuristics for lotsizing in bottleneck assembly systems. *IIE Transactions* **25**(1) 62–72.

Mathieu, V. V. 2006. Linear-programming extended formulations for the single-item lot-sizing problem with backlogging and constant capacity. *Mathematical Programming* **108**(1) 53–77.

Millar, H. H., M. Z. Yang. 1994. Lagrangian heuristics for the capacitated multiitem lot-sizing problem with backordering. *International Journal of Production Economics* **34**(1) 1–15.

Miller, A. J., G. L. Nemhauser, M. W. P. Savelsbergh. 2003a. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming* **94**(2-3) 375–405.

Miller, A. J., G. L. Nemhauser, M. W. P. Savelsbergh. 2003b. A multi-item production planning model with setup times: algorithms, reformulations, and polyhedral characterizations for a special case. *Mathematical Programming* **95**(2-3) 7190.

Özdamar, L., M. A. Bozyel. 2000. The capacitated lot sizing problem with overtime decisions and setup times. *IIE Transactions* **32**(11) 1043–1057.

Oztürk, C., A.M. Ornek. 2010. Capacitated lot sizing with linked lots for general product structures in job shops. *Computers & Industrial Engineering* **58**(1) 151–164.

Pi, L., Y. Pan, L. Shi. 2008. Hybrid nested partitions and mathematical programming approach and its applications. *IEEE Transactions on Automation Science and Engineering* **5**(4) 573–586.

Pinedo, M. 2008. *Scheduling: theory, algorithms, and systems*. Springer.

Pochet, Y., L. A. Wolsey. 1988. Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming* **40**(3) 317–335.

Pochet, Y., L. A. Wolsey. 1991. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science* **37**(1) 53–67.

Pochet, Y., L.A. Wolsey. 2006. *Production planning by mixed integer programming*. Springer.

Salomon, M. 1991. *Deterministic lot sizing models for production planning*. Spinger, Inc.

Shi, L., S. Ólafsson. 2000. Nested partitions method for global optimization. *Operations Research* **48**(3) 390–407.

Shi, L., S. Ólafsson. 2007. *Nested Partitions Optimization: Methodology And Applications*, *International Series in Operations Research & Management Science*, vol. 109. Springer.

Shi, L., S. Ólafsson, Q. Chen. 2001. An optimization framework for product design. *Management Science* **47**(12) 1681–1692.

Shi, L., S. Ólafsson, N. Sun. 1999. New parallel randomized algorithms for the traveling salesman problem. *Computers & Operations Research* **26**(4) 371–394.

Simpson, N.C., S.S. Erenguc. 2005. Modeling multiple stage manufacturing systems with generalized costs and capacity issues. *Naval Research Logistics* **52**(6) 560–570.

Song, Y.Y., G.H. Chan. 2005. Single item lot-sizing problems with backlogging on a single machine at a finite production rate. *European Journal of Operational Research* **161**(1) 191–202.

Stadtler, H. 2003. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research* **51**(3) 487–502.

Tang, O. 2004. Simulated annealing in lot sizing problems. *International Journal of Production Economics* **88**(2) 173–181.

Tempelmeier, H., M. Derstroff. 1996. A lagrangian-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science* **42**(5) 738–757.

Thizy, J. M., L. N. van Wassenhove. 1985. Lagrangean relaxation for the multi-item capacitated lot-sizing problem: a heuristic implementation. *IIE Transactions* **17**(4) 308313.

Trigeiro, W., L. J. Thomas, O. M. John. 1989. Capacitated lot sizing with setup times. *Management Science* **35**(3) 353–366.

Trigeiro, W. W. 1987. A dual-cost heuristic for the capacitated lot sizing problem. *IIE Transactions* **19**(3) 6772.

Wu, T. 2011. Lower and upper bounds evaluation of mathematical formulations for production planning with backorders. *Under revision in Operations Research Letters* .

Wu, T., L. Shi, N. A. Duffie. 2010. An HNP-MP approach for the capacitated multi-item lot sizing problem with setup times. *IEEE Transactions on Automation Science and Engineering* **7**(3) 500–511.

Zangwill, W. I. 1969. A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science* **15**(9) 506–527.