Multi-Module Capacitated Lot-Sizing Problem, and its Generalizations with Two-Echelons and Piecewise Concave Production Costs

Author information is purposely removed for double-blind review

Abstract

We study new generalizations of the classical capacitated lot-sizing problem with concave production (or transportation), holding, and subcontracting cost functions in which the total production (or transportation) capacity in each time period is the summation of capacities of a subset of n available modules (machines or vehicles) of different capacities. We refer to this problem as Multi-module Capacitated Lot-Sizing Problem without or with Subcontracting, and denote it by MCLS or MCLS-S, respectively. These are NP-hard problems if n is a part of the input and polynomially solvable for $n = 1$. In this paper, we address an open question: *Does there exist a polynomial time exact algorithm for solving the MCLS or MCLS-S with fixed $n \geq 2$?*. We present exact fixed-parameter tractable (polynomial) algorithms that solve MCLS and MCLS-S in $O(T^{2n+3})$ time for a given $n \geq 2$. We also present exact algorithms for two-generalizations of the MCLS and MCLS-S: (a) a lot-sizing problem with piecewise concave production cost functions (denoted by LS-PC-S) that takes $O(T^{2m+3})$ time, where m is the number of breakpoints in these functions, and (b) two-echelon MCLS that takes $O(T^{4n+4})$ time. We perform computational experiments to evaluate the efficiency of our algorithms for MCLS and LS-PC-S and their parallel computing implementation, in comparison to Gurobi 9.1. The results of these experiments show that our algorithms are computationally efficient and stable. Furthermore, our algorithm for MCLS-S also addresses another open question related to the existence of a polynomial time algorithm for optimizing a linear function over n-mixing set (a generalization of the well-known 1-mixing set).

Keywords: multi-module capacitated lot-sizing; piecewise concave production cost; two-echelon capacitated lot-sizing; subcontracting costs; n-mixing set; fixed-parameter tractable algorithms.

1 Introduction

With the rise in incorporation of data analytics in business decision making, there has been an increasing interest to efficiently coordinate transportation, production, subcontracting, and inventory planning. Capacitated lot-sizing problem (CLSP) is one of the most widely studied problems in operations management as its objective is to provide a transportation/production/procurement, inventory, and subcontracting plan for a given planning horizon such that demand for each time period is satisfied, and the overall cost, which includes setup, transportation (or production), holding, and subcontracting costs, is minimum. This problem has applications in numerous areas such as manufacturing, retailing, transportation, power systems, telecommunication, biomass logistics, and many more (Pochet and Wolsey 2006, Voss and Woodruff 2006). In most of the

CLSP literature, it is assumed that there is only one machine (or vehicle) with fixed capacity in each period. As a result, the production (or transportation) capacity in each time period is a binary multiple of the fixed available capacity. The capacity of the machine can remain constant (Florian and Klein 1971) or vary with time (Florian et al. 1980), thereby leading to lot-sizing problem with constant or varying capacity, respectively. We refer to these families of CLSP as single-module capacitated lot-sizing problems (SCLS). Though SCLS with varying capacity provides the flexibility of having a machine with different (fixed) capacity in each time period, it requires that the capacities of the machine are known a priori for the entire planning horizon. Moreover, even for known capacities, this problem is NP-hard. On the other hand, SCLS with constant capacity is polynomially solvable, but it restricts the production planning using only one machine of same capacity over the entire planning horizon.

In today's competitive world, transportation and manufacturing companies often have multiple (n) vehicles and machines, respectively, of different capacities that are available in each time period. Therefore, in the more realistic setting, production (or transportation) planning requires deciding which subset of these machines (or vehicles) are to be utilized in each time period. This implies that the total production capacity in each time period is the summation of binary multiples of n different capacities. We refer to this generalization of SCLS as multi-module capacitated lot-sizing (MCLS) problem (Sanjeevi and Kianfar 2012, Bansal and Kianfar 2015) and denote MCLS with an uncapacitated subcontracting (or outsourcing) option by MCLS-S. The modules can represent machines, vehicles, or suppliers of different capacities in the context of production, transportation, or procurement, respectively.

1.1 MCLS-S: Problem Definition, Motivation, and Applications

Given a planning horizon $\mathcal{T} := \{1, \ldots, T\}$, a set of n machines (denoted by $\mathcal{N} := \{1, \ldots, n\}$) with time invariant capacities $C_1, \ldots, C_n$, such that $C_1 \leq C_2 \leq \ldots \leq C_n$ (w.l.o.g.), and demand d_t of a retailer for period $t \in \mathcal{T}$, the MCLS-S is defined as follows:

$$\text{Minimize} \quad \sum_{t=1}^{T} p_t^i(x_t^i) + h_t(s_t) + g_t(z_t) + \sum_{i=1}^{n} q_t^i y_t^i \tag{1a}$$

$$\text{s.t.} \quad s_{t-1} + \sum_{i=1}^{n} x_t^i + z_t = d_t + s_t, \ t \in \mathcal{T}, \tag{1b}$$

$$x_t^i \leq C_i y_t^i, \ t \in \mathcal{T}, \ i \in \mathcal{N} \tag{1c}$$

$$x_t = \sum_{i \in N} x_t^i \leq \sum_{i \in N} C_i y_t^i, \ t \in \mathcal{T}, \tag{1d}$$

$$y_t \in \{0,1\}^n, x_t, s_t, z_t \geq 0, \ t \in \mathcal{T}, \tag{1e}$$

where x_t^i is the amount produced on machine $i \in N$ in period $t \in T$, s_t is the inventory at the end of period $t \in T$, z_t is the subcontracting quantity in period t, and y_t^i is a binary variable which determines whether the machine (or module) i of capacity C_i is utilized in period t or not. For each module $i \in N$, a setup cost of q_t^i is incurred if $y_t^i = 1$ for $t \in T$. We assume that the production cost functions $p_t^i(\cdot)$ associated with machine $i \in N$, the holding cost function $h_t(\cdot)$, and the subcontracting cost function $g_t(\cdot)$ for period $t \in T$ are concave. Also, initial inventory s_0 is assumed to be zero without loss of generality (Florian and Klein 1971). Constraints (1b) are the classical inventory (or flow) balance constraints. Constraints (1c)-(1d) ensure that the production capacity in each time period is the summation of binary multiple of n different capacities, i.e., summation of capacities of a subset of $\{C_1, \ldots, C_n\}$. The MCLS (or MCLS-S without subcontracting) is equivalent to (1) with $z_t = 0$ and $g_t(z_t) = 0$ for all $t \in T$. It is important to note that the SCLS with constant capacity in each period, i.e., MCLS-S with $n = 1$, and SCLS with variable capacity in each period (Florian et al. 1980), i.e., MCLS-S with $n = T$ and $y_t^i = 0$ for all $i \neq t$, are special cases of the MCLS-S. Note that the MCLS-S with n capacities is equivalent to MCLS with $n + 1$ capacities where $C_{n+1} = \sum_{j=1}^{T} d_j$ and setup costs $q_t^{n+1} = 0$ for all $t \in T$. However, solving MCLS-S using algorithm for MCLS is not efficient (discussed later in the paper).

Interestingly, the MCLS and MCLS-S are NP-hard even for $T = 1$, if n is a part of the input, because they reduce to a knapsack problem. However, for $n = 1$, the MCLS and MCLS-S are solved in $O(T^4)$ time by Florian and Klein (1971) and in $O(T^5)$ time by Atamtürk and Hochbaum (2001), respectively. This leads to an open question: Does there exist a polynomial time exact algorithm for solving MCLS or MCLS-S with fixed $n \geq 2$? Observe that the formulation (1) is a mixed-binary program with nT binary variables and therefore, even for fixed $n \geq 2$, it is not trivial that the MCLS-S is polynomially solvable because the number binary variables is not fixed (Lenstra 1983). In this paper, we address this open question, which also helps in showcasing the existence of a polynomial time algorithm for optimizing a linear cost functions defined over a multi-constrained continuous mixed integer knapsack set: the n-mixing set studied by (Sanjeevi and Kianfar 2012, Bansal and Kianfar 2017) which generalizes the 1-mixing set studied by Günlük and Pochet (2001). Moreover, it creates pathways for polyhedral analysis of other mixed integer sets, such as continuous multi-mixing set (Bansal and Kianfar 2014, 2015, 2017) and its extensions. (Refer to Section 3.7 for more details.)

Practical Applications of MCLS-S. In addition to the aforementioned applications in production

planning and inventory management, MCLS-S is applicable in various di erent settings of the supply chain industry. One such setting involves a retailer andn di erent suppliers each with a capacity of C_i, $i \in \{1, \ldots, n\}$, in each time period. In this setting, the setup costs are analogous to the xed ordering costs associated to each supplier, and the retailer has to decide the amount to be ordered in each time period from each supplier such that the demands are met and the overall ordering (xed and transportation) and holding costs are minimized. As discussed before, the MCLS-S is also applicable for a supply chain involving a supplier with multiple vehicles of di erent capacities and a retailer (or a warehouse of the supplier). The MCLS-S also has applications in power generation and storage management using renewable resources. In particular, renewable energy is often stored in batteries and these charged batteries of di erent storage capacities are in turn used to satisfy consumers' demand in each time period of a planning horizon. In this setting, the decisions are: (a) which batteries to activate and (b) how much energy to supply from each of these batteries, and production cost functions are analogous to linear recharging cost functions (Pandzic et al. 2015, Xu et al. 2017). Other applications involving the presence of multiple capacities include o shore natural gas pipeline systems (Brimberg et al. 2003), regional wastewater systems (Jarvis et al. 1978), cloud computing and data storage centers (Joch 2013).

1.2 Two Generalizations of MCLS-(S)

In this paper, we also present exact algorithms for two generalizations of the MCLS-(S), which are de ned as follows.

(A) Lot-sizing problem with piecewise concave production costs and subcontracting (LS-PC-S). The MCLS-S can be reformulated as a lot-sizing problem with piecewise concave production cost functions having nite (m) number of time invariant breakpoints (see Section 3.6 for details). We denote this generalization of MCLS-S by LS-PC-S, which is formulated as:

$$\text{Minimize} \sum_{t=1}^{T} p_t(x_t) + h_t(s_t) \quad s_{t-1} + x_t = d_t + s_t;\ t \in T;\ s_0 = 0;\ x_t, s_t \geq 0, t \in T \quad ; \qquad (2)$$

where the production cost function $p_t(\cdot)$ is a piecewise concave function with$m$ time invariant breakpoints $b_{i-1} < b_i$ for $i \in \{1, \ldots, m+1\}$ where $b_0 = 0$ and $b_{m+1} = 1$. Notice that there are $m+1$ cost segments in the production cost function with m breakpoints where the rightmost cost segment going fromb_m to $b_{m+1} = 1$ corresponds to subcontracting costs in this paper. However, if the option of subcontracting does not exist, LS-PC-S reduces to LS-PC, where the production cost function still has m breakpoints with $b_{m+1} = b_m$ but only m cost segments.

The LS-PC-S is equivalent to MCLS without constraints (1c), and with $p_t(\cdot)$ as production cost function. Refer to Section 2 for other special cases of LS-PC-S studied in the literature.

(B) Two-echelon MCLS (2E-MCLS). We generalize MCLS by considering an integrated supply chain model where in addition to planning the production and inventory at the manufacturing shop (i.e., first echelon), we also decide the amount to be transported to a retailer (i.e., second echelon) with known demand d_t, $t \in \mathcal{T}$, and the inventory at the retailer's end during each period of the planning horizon. The network flow representation of 2E-MCLS is given in Figure 1, where node $(e;t)$ represents echelon $e \in \{1,2\}$ and time period $t \in \mathcal{T}$ of the planning horizon. Decision variable $x_t^{1,i}$ for $i \in \mathcal{N} = \{1,\dots,n\}$ denotes the amount produced in period $t \in \mathcal{T}$ using machine i of capacity C_i, and x_t^2 denotes the amount transported from the first echelon to the second echelon during period $t \in \mathcal{T}$. The amount of inventory held at the end of period t in first and second echelons are denoted by s_t^1 and s_t^2, respectively.

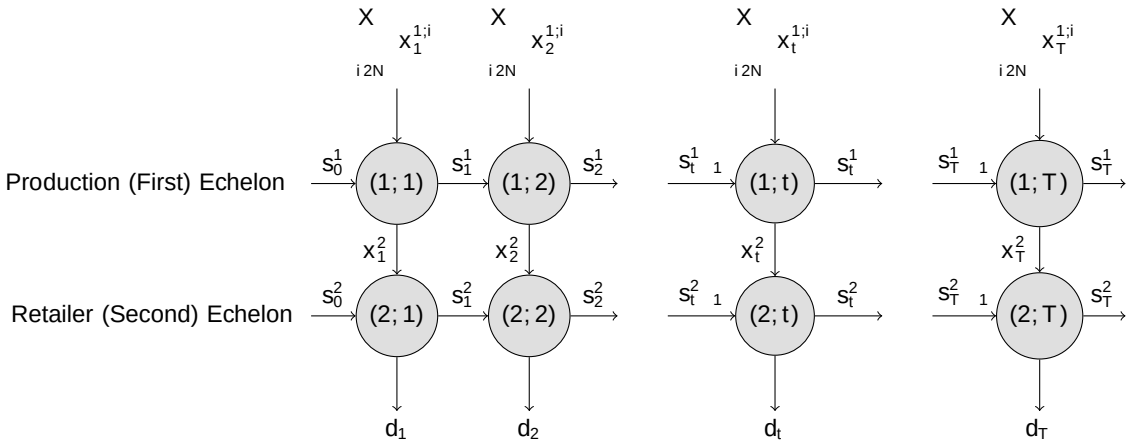

Figure 1: Network Flow Representation of the 2E-MCLS

Mathematically, the 2E-MCLS is formulated as follows:

$$\text{Minimize} \quad \sum_{t=1}^{T} \sum_{i=1}^{n} p_t^{1,i}(x_t^{1,i}) + q_t^{1,i} y_t^{1,i} + p_t^2(x_t^2) + q_t^2 y_t^2 + \sum_{e=1}^{2} h_t^e(s_t^e) \tag{3a}$$

$$\text{s.t.} \quad s_{t-1}^1 + \sum_{i=1}^{n} x_t^{1,i} = s_t^1 + x_t^2, \; t \in \mathcal{T}; \tag{3b}$$

$$s_{t-1}^2 + x_t^2 = d_t + s_t^2, \; t \in \mathcal{T}; \tag{3c}$$

$$x_t^{1,i} \le C_i y_t^{1,i}, \; t \in \mathcal{T}, \; i \in \mathcal{N}; \tag{3d}$$

$$x_t^2 \le M y_t^2, \; t \in \mathcal{T}; \tag{3e}$$

$$y_t^{1,i}, y_t^2 \in \{0,1\}, \; x_t^{1,i}, x_t^2, s_t^e \ge 0, \; t \in \mathcal{T}, \; e \in \{1,2\}; \tag{3f}$$

where $p_t^{1,i}(\,)$, $p_t^2(\,)$ and $h_t^e(\,)$ for $e \in \{1,2\}$ denote the production, transportation, and holding cost functions, respectively, which are assumed to be concave. Parameter $q_t^{1,i}$ is the cost to setup machine i at the first echelon in period t, and q_t^2 is the setup cost for transportation from the first echelon to the second echelon. Constraints (3b) and (3c) are the flow-balance constraints corresponding to the first and the second echelons. According to Constraints (3d), the total production in each period is the sum of binary multiples of n different capacities as binary variable $y_t^{1,i}$ is equal to 1 if $x_t^{1,i} > 0$, and zero otherwise. Likewise, binary variable y_t^2 in constraint (3e) is equal to 1 if $x_t^2 > 0$ and thereby leading to uncapacitated transportation as M is a large number. The 2E-MCLS with $n = 1$ reduces to the two echelon single-module capacitated lot-sizing problem (or 2E-SCLS), studied by van Hoesel et al. (2005). Note that when $p_t^2 = q_t^2 = s_t^2 = 0$ for all $t \in T$, 2E-MCLS reduces to MCLS with machine dependent production cost functions.

1.3 Contributions of This Paper

The contributions of this paper are as follows.

(i) For a given $n \in Z_+$, we develop dynamic programming (DP) algorithms to solve MCLS and MCLS-S to optimality in $O(T^{2n+3})$ time. These algorithms belong to the class of fixed-parameter tractable algorithms (Downey and Fellows 2012, Flum and Grohe 2006) because for fixed n, these are polynomial algorithms for MCLS and MCLS-S. Note that solving MCLS-S using the DP algorithm for MCLS yields a worst-case time complexity of $O(T^{2n+5})$. To the best of our knowledge, these are the first dynamic programming algorithms that directly solve MCLS and MCLS-S with a fixed $n \geq 2$. The novelty of the proposed DP algorithm for MCLS-(S) is in designing state and stage variables, and recursive equations along with the idea of fractional production levels. Moreover, Gurobi 9.1 could not solve 21 out of the 50 MCLS instances within the time limit of 2000 seconds whereas the DP algorithm solved all these unsolved instances in less than 161 seconds and 47 seconds on average. Additionally, for the remaining 29 unsolved instances, our algorithm took 43 seconds and Gurobi took 484 seconds, on average.

(ii) Using the algorithm for MCLS-S, we address the existence of a polynomial time algorithm for optimizing a linear function over n-mixing set, i.e.,

$$P^{n,T} := \{(\,;\,) \in R_+ \times Z_+^{nT} : \; + \sum_{i \in N} C_i \; {}_t^i \;\; d_{1t} \text{ for } t \in T \;\; ; \qquad (4)$$

for fixed $n \in \mathbb{Z}_+$. Günlük and Pochet (2001) provided the convex hull description of $P^{1,T}$ (i.e., 1-mixing set). Miller and Wolsey (2003) presented a compact tight extended formulation for $P^{1,T}$, thereby showcasing the existence of a polynomial time algorithm for optimizing a linear function over the 1-mixing set. However, for $n \geq 2$, only a few families of facet-defining inequalities for $P^{n,T}$ are known in the literature (refer to Section 3.7 for details). Even for any fixed $n \geq 2$, the convex hull description, a compact tight extended formulation, or a polynomial time optimization algorithm for $P^{n,T}$ are not known in the literature.

(iii) We also present a computationally efficient algorithm for LS-PC-S, which is a generalization of MCLS-S and an important problem in its own right as it subsumes various other classes of lot-sizing problems. In comparison to the first and the only known DP algorithm for this problem by Koca et al. (2014), the new DP algorithm explores a significantly reduced search space using a reduced number of state variables and thereby leading to different recursive equations. As a result, the latter is 16 and 10 times (on average) faster than the former and Gurobi 9.1 for solving the LS-PC-S instances. To provide insights, we solve a numerical example in Appendix E, and observe that the former computes 4352 function values, in comparison to 100 function values computed by the latter. For detailed comparison, please refer to the recursive equations in Appendix E and Section 4.

(iv) For fixed $n \in \mathbb{Z}_+$, we propose the first DP algorithm for 2E-MCLS that takes $O(T^{4n+4})$ time, thereby generalizing DP algorithms of van Hoesel et al. (2005) for 2E-MCLS with $n = 1$ and of Zangwill (1969) for 2E-MCLS with $n = 1$ and $C_1 = 1$ (or a large number).

In Table 1, we summarize the worst-case time complexities of exact algorithms for MCLS, MCLS-S, LS-PC-S, and 2E-MCLS. We use MCLS-(S) to denote MCLS and MCLS-S together.

Problem	Worst-Case Complexity	Contributors
MCLS with $n = 1$	$O(T^4)$	Florian and Klein (1971)
MCLS-S with $n = 1$	$O(T^5)$	Atamtürk and Hochbaum (2001)
MCLS-(S) with fixed $n \geq 2$	$O(T^{2n+3})$	This Paper
LS-PC-S, i.e., MCLS-S with $n = 1$, $C_1 = 1$, and piecewise concave production costs	$O(T^{2n+3})$	Koca et al. (2014); This Paper
2E-MCLS with $n = 1$	$O(T^7)$	van Hoesel et al. (2005)
2E-MCLS with fixed $n \geq 2$	$O(T^{4n+4})$	This Paper

Table 1: Worst-case complexity of algorithms for MCLS, MCLS-S, and their special cases

Remark 1. The MCLS-S problem with n capacities can be reformulated as the LS-PC-S problem

where the breakpoints of the production functions are determined by taking all possible binary combinations of the n available capacities. As a result, aforementioned solution approach for LS-PC-S solves the MCLS-S in $O(T^{2^{n+1}+1})$ time (see Section 3.6 for more details). This implies that in order to solve the MCLS-S with 5, 10, or 20 vehicles of different capacities, this approach takes $O(T^{65})$, $O(T^{2049})$, or $O(T^{2097153})$ time, respectively, in comparison to $O(T^{13})$, $O(T^{23})$, or $O(T^{43})$ time taken by solving it using our algorithm for MCLS-S. Likewise, a special case of the MCLS-S problem with linear production cost functions can be reformulated as LS-PC-S problem with piecewise linear production cost functions having $2^n - 1$ breakpoints. Using the DP algorithm of Ou (2017), this reformulation can be solved in $O(T^{2^{n+1}} \log T)$ time, i.e., for $n = 20$, it takes $O(T^{1048577} \log T)$ time.

Below we provide a list of acronyms of variants of MCLS and MCLS-S discussed in this paper:
ULSP : Uncapacitated lot-sizing problem (Wagner and Whitin 1958, Aggarwal and Park 1993).
SCLS : Single-module capacitated lot-sizing problem.
SCLS-CC : SCLS with constant capacities (Florian and Klein 1971).
SCLS-CC-S : SCLS-CC with an option of subcontracting (Atamtürk and Hochbaum 2001).
LS-PC-S : Lot-sizing problem with piecewise concave production costs, (Koca et al. 2014).

Organization of the rest of the paper. In Section 2, we discuss special cases and variants of MCLS-S,LS-PC-S, and 2E-MCLS studied in the literature. We present DP algorithms to solve MCLS-(S) LS-PC-S, and 2E-MCLS in Sections 3, 4, and 5 respectively, and derive their worst-case time complexities. In Section 3, we also showcase how the DP algorithm for MCLS-S is applicable for optimizing a linear cost function over n-mixing set, and present LS-PC-S reformulation of MCLS-S along with the computational complexity of solving this reformulation using an algorithm for LS-PC-S. In Section 6, we report the results of our computational experiments for MCLS-S and LS-PC-S instances. In Section 7, we provide concluding remarks.

2 Literature Review

We review the lot-sizing problems which are closely related to MCLS-(S), LS-PC-S, and 2E-MCLS, in particular, uncapacitated, constant capacitated, constant batch, multi-mode, and two-echelon capacitated lot-sizing problems. We also discuss about lot-sizing problems with subcontracting and/or piecewise concave production cost functions. For extensive literature review on single-item lot-sizing problems, reader can refer to the survey papers by Brahimi et al. (2006, 2017).

Uncapacitated Lot-Sizing Problem (ULSP). Wagner and Whitin (1958) studied the uncapacitated version of the MCLS, i.e. MCLS with $n = 1$ and $C_1 = 1$, with linear production and holding cost functions, and provided an $O(T^2)$ dynamic programming exact algorithm for it. Wagelmans et al. (1992) studied the ULSP with and without non-speculative costs, and presented an $O(T)$ and $O(T \log T)$ time algorithms for them, respectively. Van Hoesel et al. (1994) studied the discrete lot-sizing problem where only a fixed setup cost and holding cost is incurred every time period, and they presented a dynamic programming algorithm to solve it. Later, Aggarwal and Park (1993) studied the same problems with and without non-speculative costs and presented algorithms to solve it in $O(T)$ and $O(T \log T)$ time, respectively. Veinott (1963) showed that the ULSP with concave production and holding cost functions can be solved in $O(T^2)$ time.

ULSP with Piecewise Concave Production Cost Functions (denoted by LS-PC-S). Federgruen and Lee (1990) introduced an $O(T^4)$ time DP algorithm for the ULSP with piecewise linear (with one time invariant breakpoint) production cost functions and linear holding cost functions. In contrast, Chan et al. (2002) proved that the ULSP with piecewise linear production cost functions, where the number of breakpoints is arbitrary over time, is NP-hard. Several authors have considered lot-sizing problems with step-wise or staircase cost structures. For example, Li et al. (2012) studied ULSP with all-units discount, i.e. slopes of the cost segments (which is essentially the per unit production cost) are in a decreasing order, and they presented an $O(T^{m+3})$ time DP algorithm to solve it, where m is the number of time invariant breakpoints in the piecewise linear production cost functions. Archetti et al. (2014) developed an $O(m^2T^3)$ time algorithm for ULSP with a special case of all-unit discount costs in which the production cost function has alternate flat and increasing cost sections such that all breakpoints are equispaced and the slopes of increasing cost sections are in a decreasing order.

Atamtürk and Hochbaum (2001) studied the ULSP with piecewise production cost function with one time invariant breakpoint and two concave cost segments. In other words, they studied LS-PC-S with $m = 1$, and introduced a DP algorithm that runs in $O(T^5)$ time to solve it. It should be noted that all aforementioned problems are special cases of LS-PC-S. For a given number of time invariant breakpoints, Koca et al. (2014) developed the first polynomial $O(T^{2m+3})$ time DP algorithm that solves LS-PC-S to optimality. Recently, Ou (2017) used a specific range minimum query data structure within the algorithm of Koca et al. (2014) for solving LS-PC-S with linear holding cost function and piecewise linear production cost function in $O(T^{m+2} \log T)$

time. There is no computational result known for this algorithm. Several other special cases of LS-PC-S have been studied in the literature; reader can refer to Koca et al. (2014) for details.

Single-Module Capacitated Lot-Sizing Problem with Constant Capacity (SCLS-CC). The SCLS where the production capacity is constant over time, i.e., MCLS with $n = 1$, is a generalization of ULSP and a special case of LS-PC-S with $m = 1$ and no subcontracting. Florian and Klein (1971) presented a DP algorithm, which runs in $O(T^4)$ time, for the SCLS-CC with concave production and holding cost functions. Van Hoesel and Wagelmans (1996) and Ou (2012) studied the foregoing problem with linear holding cost functions and gave algorithms that run in $O(T^3)$ time. As mentioned before, Atamtürk and Hochbaum (2001) developed an $O(T^5)$ time DP algorithm to solve LS-PC-S with $m = 1$, which is equivalent to SCLS-CC with an additional option of subcontracting (SCLS-CC-S) i.e. MCLS-S with $n = 1$. Specifically, in LS-PC-S with $m = 1$, the first cost segment corresponds to regular production cost function, the second segment corresponds to subcontracting cost, and breakpoint represents capacity of the machine.

Multi-Module Capacitated Lot-Sizing Problem (MCLS). Florian et al. (1980) and Bitran and Yanasse (1982) independently proved that MCLS with $n = T$ and $y_t^i = 0$ for all $i \neq t$, i.e., if the capacities in each time period over the planning horizon are arbitrary, is NP-hard in general. Florian et al. (1980) presented a pseudo-polynomial time algorithm to solve this problem in $O(D_T C_T)$ time, where $D_T = \sum_{t=1}^{T} d_t$ is the total demand and $C_T = \sum_{t=1}^{T} C_t$ is the total capacity of the planning horizon. Pochet (1988) analyzed the polyhedral structure of the feasible region of this problem and developed valid inequalities for it. Shaw and Wagelmans (1998) studied a generalization of this problem where the production cost function is piecewise linear with m_t breakpoints in each time period. They provided a pseudo-polynomial algorithm that runs in $O(D_T M_T)$ where $M_T = \sum_{t=1}^{T} m_t$. Akbalik and Rapine (2013) studied a multi-mode replenishment problem where multiple batches of same capacity can be ordered in any given time period, but the capacities are time-variant. For linear cost functions, the authors proved that the foregoing problem and its several special cases are NP-hard. Several heuristics based on Lagrangian relaxation or column generation approach have been developed to solve the multi-item versions of MCLS with $n = T$ and $y_t^i = 0$ for all $i \neq t$, and we refer the reader to Karimi et al. (2003) for more details.

Additionally, polyhedral structure of variants of MCLS with linear cost functions have been studied where the production capacity in each time period is the sum of some integer multiples

of capacities with di erent sizes, i.e. MCLS with $y_t^j \in Z_+$ instead of $y_t^j \in \{0,1\}$, for $t \in T$ and $j \in N$. This class of lot-sizing problems has been referred to multi-mode/batch/module capacitated lot-sizing problems (we denote it by iMCLS). Pochet and Wolsey (1993) presented an $O(T^3)$ DP algorithm and so-called (k,l,S,I) valid inequalities for the lot-sizing problem with multiple batches of a single capacity, each incurring a setup cost, i.e., iMCLS with $n = 1$. Sanjeevi and Kianfar (2012) and Bansal and Kianfar (2015) presented generalizations of (k,l,S,I) inequalities for iMCLS without backlogging and iMCLS with backlogging, respectively. Bansal (2019) presented the conditions under which the aforementioned inequalities are facet-de ning. Conforti and Wolsey (2008) presented an extended formulation for a special case of iMCLS with $n = 2$ where the capacities are divisible, i.e., capacity C_1 is an integer multiple of capacity C_2. Recently, Kulkarni and Bansal (2020) studied MCLS with backlogging and multiple items, but assumed that the production capacity $x_t = \sum_{i \in N} C_i y_t^i$ for $t \in T$. They presented an $O(T^{n+1})$ time algorithm for single-item version, thereby generalizing an algorithm of Van Vyve (2007) for $n = 1$. They also utilized this algorithm within a Lagrangian decomposition framework to solve the multi-item version of this problem.

Akbalik et al. (2017) also studied a variant of SCLS-CC-S (referred to as lot-sizing problem with batch ordering and capacity reservation contracts or LS-BCR) where R batches of a single capacity can be produced in-house and the rest of the batches can be subcontracted. This problem is also equivalent to iMCLS with $n = 2$ where $C_1 = C_2 = C$, $y_t^1 \quad R$, y_t^2 is unrestricted, and production cost functions are linear. Observe that LS-BCR is a special case of iMCLS with $n = 2$ and divisible capacities studied by Conforti and Wolsey (2008) where $y_t^1 \quad R$ for all $t \in T$. For $R = 1$ and $y_t^2 \quad 1$, LS-BCR reduces to MCLS with $n = 2$, $C_1 = C_2 = C$, and linear cost functions. Akbalik et al. (2017) studied special cases of LS-BCR with time varying capacities and proved that they are generalizations of SCLS with time varying capacities and therefore, they are NP-hard. They also presented a pseudo-polynomial time algorithm that solves LS-BCR with time varying capacities.

Eksioglu (2009) studied a variant of iMCLS with linear holding costs and additional constraints $x_t^i \quad (\sum_{j=t}^{T} d_j) y_t^i$ for $i \in N$, $t \in T$ where $y_t^i \in \{0,1\}$ incurs a machine speci c time invariant cost of K_i when $y_t^i = 1$. In addition, it is assumed that the production and replenishment costs are stationary, i.e. $p_t^i(x_t^i) = p^i x_t^i$ and $q_t^i = q^i$ for all $i \in N$ and $t \in T$. They refer to this problem as an uncapacitated economic lot-sizing with multi-mode replenishment and provide a reformulation to derive a stronger lower bound in comparison to linear programming relaxation.

For a special case with zero replenishment cost, i.e., $q^i = 0$ for all $i \in N$, they utilize `Zero Inventory Ordering Property' by Wagner and Whitin (1958) to solve this special case in $O(nT^2)$ time. Ou and Feng (2019) studied a variant of the iMCLS with $n = 1$ and $y_t^1 \leq M$, where M is an input parameter. In addition to the conventional production, setup, and holding costs, a capacity adjustment cost $V(y_{t-1}^1, y_t^1)$ is considered and it is assumed that the cost function values $V(y_{t-1}^1, y_t^1)$ are given for all feasible values of y_{t-1}^1 and y_t^1 for $t \in T$. They provided an $O(M^2T^4)$ time algorithm that solves the foregoing problem.

Two-Echelon Lot-Sizing Problems. Zangwill (1969) developed $O(T^4)$ time algorithm for solving 2E-MCLS with $n = 1$ and $C_1 = 1$, i.e., uncapacitated production. van Hoesel et al. (2005) presented an $O(T^7)$ time DP algorithm that solves 2E-MCLS with $n = 1$. Kaminsky and Simchi-Levi (2003) studied a special case of three echelon serial supply chain where two of the echelons represent production and the middle echelon represents transportation. They showed that when the costs are assumed to be non-speculative and linear, the problem can be solved in $O(T^8)$ time. Hwang et al. (2013) introduced a new approach of using basis paths to solve 2E-MCLS with $n = 1$ in $O(T^8)$ time. They also proved that this approach solves a generalization of L-echelon MCLS with $n = 1$ in $O(LT^8)$ time. Later, Hwang et al. (2016) studied 2E-MCLS with $n = 1$ with general concave production and holding costs and provided an $O(T^6)$ time algorithm. Zhang et al. (2012) presented an $O(T^4)$ time algorithm to solve 2E-MCLS with $n = 1$, $C_1 = 1$, and intermediate demand, i.e., demand in each period at both first and second echelons. Recently, Zhao and Zhang (2020) studied 2E-MCLS with $n = 1$ and intermediate demands. We refer the reader to a survey paper by Brahimi et al. (2017) that covers several other special cases of the problem 2E-MCLS with production capacities and concave production and holding costs. In this paper, we assume the demand in the first echelon to be zero. To the best of our knowledge, 2E-MCLS has not been studied explicitly in the literature.

3 Exact Algorithms for MCLS and MCLS-S

In this section, we analyze the structure of optimal solution of MCLS-(S) problem and present a DP algorithm to exactly solve MCLS, which we then extend to solve MCLS-S. We also prove that for a given n, these algorithms take polynomial $O(T^{2n+3})$ time. Below we first define the concepts of regeneration interval, semi-regeneration interval, and fractional period for MCLS-(S).

Definition 1 (Regeneration Interval). For a given k and l, $1 \leq k \leq l \leq T$, an interval

$[k;l] := \{k, k+1, \ldots, l-1, l\}$ is called a regeneration interval if and only if $s_{k-1} = s_l = 0$ and $s_t > 0$ for all $t = k, \ldots, l-1$.

Definition 2 (Semi-Regeneration Interval). For a given $k \in \{1, \ldots, T\}$, an interval $[k;T]$ is called a semi-regeneration interval if and only if $s_{k-1} = 0$ and $s_t > 0$ for $t = k, \ldots, T$.

Definition 3 (Fractional Period for MCLS). For MCLS, a period is called a fractional period if exactly one machine is not producing at full capacity. In other words, period t is a fractional period if $0 < x_t^j < C_j$ for some $j \in N$, and $x_t^i \in \{0, C_i\}$ for $i \in N \setminus \{j\}$

Definition 4 (Fractional Period for MCLS-S). For MCLS-S, a period t is a fractional period if one of the following conditions is satisfied:

(a) $0 < x_t^j < C_j$ for some $j \in N$, $x_t^i \in \{0, C_i\}$ for $i \in N \setminus \{j\}$, and $z_t = 0$;

(b) $x_t^i \in \{0, C_i\}$ for all $i \in N$, and $z_t > 0$.

Theorem 1. There exists an optimal solution to MCLS-(S) which comprises of: (a) a series of regeneration intervals, each having at most one fractional period, that span the interval $[1;T']$ for some $0 \leq T' \leq T$, and (b) a semi-regeneration interval $[T'+1;T]$ with $s_t > 0$ for all $T'+1 \leq t \leq T$ which has no fractional period.

Proof. Refer to Appendix A. □

Our DP algorithm for MCLS with fixed $n \in Z_+$ comprises of the following two phases:

(I) Consider all regeneration and semi-regeneration intervals $[k;l]$ where $1 \leq k \leq l \leq T$ and compute the optimal costs $_{kl}$ for these intervals.

(II) Construct a directed acyclic graph $G = (V;E)$ where $V = (1, \ldots, T+1)$ and $E = \{(k;l+1) : 1 \leq k \leq l \leq T\}$, assign the cost $_{kl}$ to arc $(k;l+1)$, and find the least cost route from node 1 to $T+1$.

Since phase II can be performed using any shortest path algorithm (Cormen et al. 2009), for the remaining of this section, we mainly focus on phase I, that is, computing $_{kl}$, the minimum cost of planning the production for an interval $[k;l]$. To do so, we leverage the structure of the optimal solution to the MCLS provided in Theorem 1 and compute the following:

1. Minimum cost for satisfying the demands in each period of the regeneration or semi-regeneration interval $[k;l]$ with no fractional period (Section 3.1);
2. All possible fractional production levels in the regeneration interval $[k;l]$ (Section 3.2);

3. Minimum cost for satisfying the demands in each period of regeneration interval $[k, l]$ with at most one fractional period (Section 3.3).

We define e_i as a unit vector of size n in which only the ith element is one and the rest are zero. Let $\alpha \in Z_+^n$ be a vector of α_i's where α_i is the number of times machine i of capacity C_i has been set up. We define d_{kt} as the cumulative demand from period k up to period t, i.e., $d_{kt} = \sum_{j=k}^{t} d_j$ for $t \geq k$.

3.1 Minimum cost with no fractional period

In this section, we compute the minimum cost of satisfying the demands in the regeneration interval $[k, l]$ where $1 \leq k \leq l \leq T$ and semi-regeneration interval $[k, T]$ where $1 \leq k \leq T$. For $t \in \{k, \ldots, l\}$, let $G_k(t; \alpha; 0)$ be a function that denotes the value of minimum cost solution for periods k up to t during which machine $i \in \{1, \ldots, n\}$ runs α_i times at full capacity and no fractional production takes place. The zero in $G_k(t; \alpha; 0)$ implies that there is no fractional production during interval $[k, t]$. Thus, the total amount produced between period k and t is $\sum_{i=1}^{n} \alpha_i C_i$. It is important to note that if $\sum_{i=1}^{n} \alpha_i C_i \geq d_{kt}$ for $t \geq l - 1$, then s_t cannot be strictly positive and in such case, we set the value of function $G_k(t; \alpha; 0)$ to infinity. Likewise, in case $\sum_{i=1}^{n} \alpha_i C_i \leq d_{kl}$ for $l \leq T - 1$, we set $G_k(l; \alpha; 0) = \infty$ because in the regeneration interval $[k, l]$, either demand is not satisfied or the inventory at the end of period l is positive. However, for $l = T$, it is not necessary for the ending inventory s_T to be zero (Theorem 1). Thus, we set $G_k(l; \alpha; 0) = \infty$ if $\sum_{i=1}^{n} \alpha_i C_i < d_{kl}$ for $l = T$. Also, since there are only $t - k + 1$ periods available between k and t, machine i can run in at most $t - k + 1$ periods. Thus, if $\alpha_i > t - k + 1$ for any $i \in N$, we set $G_k(t; \alpha; 0) = \infty$ as the corresponding production schedule is infeasible. We also define $G_k(k - 1; \alpha; 0) = 0$ if $\alpha_i = 0$ for all $i \in N$ and $G_k(k - 1; \alpha; 0) = \infty$ otherwise.

In order to ensure that there is no fractional period in the (semi)-regeneration interval $[k, l]$, our goal is to find the minimum among two choices at every time period $t \in \{k, \ldots, l\}$: either not to produce at all, or to produce by running any nonempty subset of the n available machines at full capacity. If we choose to set up some subset $S \subseteq N$ of the n available machines during time period t, then the value of function $G_k(t; \alpha; 0)$ is obtained by taking the sum of the function value at $(t - 1; \alpha - \sum_{i \in S} e_i; 0)$, the cost of setting up the set S of machines and producing at their full capacity, and the holding costs at the end of period t, i.e., $G_k(t; \alpha; 0) = G_k(t - 1; \alpha - \sum_{i \in S} e_i; 0) + \sum_{i \in S} p_t^i(C_i) + q_t^i + h_t \left(\sum_{i=1}^{n} \alpha_i C_i - d_{kt} \right)$, and then minimizing this summation over all possible subsets S of N. If we choose not to produce any-

thing, then $S = \emptyset$. By incorporating these observations, we derive the forward recursion (5) to compute $G_k(t; \ ; 0)$ for all possible values of $\ $ and $t \in \{k, \ldots, l\}$. For a given t and $\ $, $G_k(t; \ ; 0)$ can be computed in constant time because the number of all possible subsets of $\{1, \ldots, n\}$ is constant for a fixed n. Note that $0 \leq \ _i \leq T$ for all $i \in N$ and thus the total number of possible $\ $ vectors are $O(T^n)$. Since there are $O(T)$ time periods within an interval $[k; l]$, all values of $G_k(t; \ ; 0)$ can be computed in $O(T^{n+1})$ time.

$$G_k(t; \ ; 0) = \begin{cases} \infty, & \text{if } \ _i > t - k + 1 \text{ for any } i \in N \\ & \text{or } \sum_{i=1}^{n} \ _i C_i < d_{kt} \text{ for } t = l - 1, \text{ or } \sum_{i=1}^{n} \ _i C_i \leq d_{kl} \text{ for } t = l = T - 1, \\ & \text{or } \sum_{i=1}^{n} \ _i C_i < d_{kt} \text{ for } t = l = T \\ \min_{S \subseteq N} \left\{ G_k(t-1; \ - e_i; 0) + \sum_{i \in S} p_t^i(C_i) + q_t^i + h_t \left(\sum_{i=1}^{n} \ _i C_i - d_{kt} \right) \right\}, & \text{otherwise.} \end{cases} \tag{5}$$

3.2 Calculating possible fractional production levels

We define $^{max} \in Z_+^n$ as a vector of $_i^{max}$'s where $_i^{max}$ is the maximum number of times machine i can be set up in the interval $[k; l]$. Clearly, $_i^{max} = \min\{T; \lfloor d_{kl}/C_i \rfloor\}$ for $i = 1, \ldots, n$. We also define $\ := \{ \ \in Z_+^n : 0 < d_{kl} - \sum_{i=1}^{n} \ _i C_i < C_n; \text{ and } d_{kl} - \sum_{i=1}^{n} \ _i C_i \geq \{C_1, \ldots, C_{n-1}\} \}$. Then, the set of all possible fractional production levels in a regeneration interval $[k; l]$ is given by $F := \{ f_u : f_u = d_{kl} - \sum_{i=1}^{n} \ _i^u C_i \text{ for all } \ ^u \in \ \}$. For a given $\ ^u \in \ $, we can compute the fractional production level in constant time and since $\ _i^u \leq \ _i^{max} \leq T$, there are $O(T^n)$ possible $\ ^u$ vectors in $\ $. Hence, we find all possible fractional production levels in $O(T^n)$ time.

3.3 Minimum cost with at most one fractional period

Next, we compute the value of minimum cost for satisfying demands in a regeneration interval $[k; l]$ with at most one fractional period. For each fractional production level $f_u \in F$, we define $v_{f_u} \in N$ such that $f_u < C_{v_{f_u}}$ to denote the machine on which the fractional production can occur. For each fractional production level $f_u \in F$ (computed in the previous section) and $t \in \{k, \ldots, l\}$, let function $G_k^u(t; \ ; 1)$ denote the value of minimum cost solution for periods k up to t during which machine $i \in N$ runs $_i$ times at full capacity, and the fractional production takes place at most once. We set $G_k^u(t; \ ; 1) = \infty$ for the following infeasible production schedule cases: (a) $\ _i > t - k + 1$ as the maximum number of times machine i can run from period k through t is $t - k + 1$; (b) $\sum_{i=1}^{n} \ _i C_i + f_u < d_{kt}$, that is, demand up to period $t - l - 1$ is not satisfied (since backlogging is not allowed) and $s_t > 0$ for all $t \in \{k, \ldots, l - 1\}$; and

(c) $\sum_{i=1}^{n} \alpha_i C_i + f_u \neq d_{kl}$ for $t = l$, since $s_l \neq 0$ which contradicts the definition of regeneration interval. In case the fractional production level is zero, $G_k^u(t; \alpha; 1) = G_k(t; \alpha; 0)$.

Now, in order to ensure that the fractional production f_u occurs at most once throughout the regeneration interval $[k; l]$, our goal is to find the minimum among three possible choices at every time period $t \in \{k, \ldots, l\}$: (i) choose not to produce at all, (ii) setup some nonempty subset of n machines and run them to produce only at full capacity, and (iii) run some machines at full capacity and one machine at less than its capacity (fractional production). If we choose to run a subset $S \subseteq N$ among n available machines at full capacity, then the value of the function $G_k^u(t; \alpha; 1)$ is calculated by aggregating the function value at $t - 1; \sum_{i \in S} e_i$, the cost of running the set S of machines at full capacity, and the holding costs at the end of period t, i.e., $G_k^u(t - 1; \sum_{i \in S} e_i; 1) + \sum_{i \in S} p_t^i(C_i) + q_t^i + h_t \left(\sum_{i=1}^{n} \alpha_i C_i + f_u - d_{kt} \right)$, and then taking minimum of the foregoing expression among all possible subsets S of N. Notice that when $S = \emptyset$, we choose not to produce anything in period t.

However, as per case (iii), if we choose to setup a nonempty subset $S \subseteq N \setminus \{v_{f_u}\}$ of the $n - 1$ machines such that we produce the fractional production f_u in period t on machine with capacity $C_{v_{f_u}}$ and run the remaining machines at full capacity, then the value of the function $G_k^u(t; \alpha; 1)$ will be equal to the minimum, among all possible subsets S, of the sum of the value of minimum cost for satisfying demands in the interval $[k; t - 1]$ with no fractional period during which machine i runs $\alpha_i - 1$ times for $i \in S$ and α_i times, otherwise, i.e. $G_k(t - 1; \sum_{i \in S} e_i; 0)$, the cost of setting up the set S of machines among $n - 1$ available machines, the cost of producing on set S of machines at full capacity and producing f_u on machine v_{f_u}, and the cost of holding inventory at the end of period t, i.e.

$$G_k\left(t - 1; \sum_{i \in S} e_i; 0\right) + \sum_{i \in S} q_t^i + q_t^{v_{f_u}} + \sum_{i \in S} p_t^i(C_i) + p_t^{v_{f_u}}(f_u) + h_t \left(\sum_{i=1}^{n} \alpha_i C_i + f_u - d_{kt} \right).$$

Using these observations, we again derive a forward recursion (6) to compute the values of $G_k^u(t; \alpha; 1)$ for all $t \in \{k, \ldots, l\}$, $\alpha \in Z_+^n$, and $f_u \in F$. Recall from Section 3.2 that the total number of fractional production levels are $O(T^n)$. Also, for a given regeneration interval $[k; l]$, since $0 \leq \alpha_i \leq T$ for all $i \in N$ and $k \leq t \leq l$, the total number of possible α vectors and time periods are $O(T^n)$ and $O(T)$, respectively. Moreover, for a given $t \in \{k, \ldots, l\}$, $\alpha \in Z_+^n$, and $f_u \in F$, $G_k^u(t; \alpha; 1)$ can be computed in $O(2^n)$ time which is essentially constant time since n is assumed to be fixed. Therefore, it takes $O(T^{2n+1})$ time to compute $G_k^u(t; \alpha; 1)$ for all possible values of t, α, and f_u.

$$
G_k^u(t;\ ;1) = \begin{cases}
1, & \text{if } {}_i > t \quad k+1 \text{ for any } i \in N \text{ or } \sum_{i=1}^{n} {}_i + 1 \quad n(t \quad k+1) \qquad (6a)\\
& \text{or } \sum_{i=1}^{n} {}_i C_i + f_u \quad d_{kt} \text{ for } t \quad l \quad 1 \qquad (6b)\\
& \text{or } \sum_{i=1}^{n} {}_i C_i + f_u \ \ d_{kl} \text{ for } t = l \qquad (6c)\\
G_k(t;\ ;0); & \text{if } d_{kl} \sum_{i=1}^{n} {}_i^u C_i = 0 \qquad (6d)\\
\min\Big\{ \min_{S \subseteq N} \Big[G_k^u(t \quad 1;\ \sum_{i \in S} e_i;1) + \sum_{i \in S} p_t^i(C_i) + q_t^i \Big] & \\
\quad + h_t \Big(\sum_{i=1}^{n} {}_i C_i + f_u \quad d_{kt} \Big); & \qquad (6e)\\
\min_{\substack{S \subseteq N \cap \{v_{f_u}\} \\ v_{f_u} \in N}} \Big[G_k(t \quad 1;\ \sum_{i \in S} e_i;0) + \sum_{i \in S} p_t^i(C_i) + q_t^i & \\
\quad + p_t^{v_{f_u}}(f_u) + q_t^{v_{f_u}} + h_t \Big(\sum_{i=1}^{n} {}_i C_i + f_u \quad d_{kt} \Big) \Big] \Big\} & \text{; otherwise.} \qquad (6f)
\end{cases}
$$

3.4 Extension to MCLS-S

In this section, we modify the aforementioned recursive equations for MCLS to compute the minimum cost of a regeneration interval $[k;l]$ for MCLS-S. Note that computing the minimum cost without fractional period for MCLS-S is same as for MCLS (discussed in Section 3.1). In MCLS-S, we treat the amount subcontracted as a fractional production and since there is no capacity constraint on the amount that can be subcontracted, the fractional production levels can be greater than C_n. Therefore, to compute the possible fractional production levels, we rede ne the set as $^0 := \quad 2 Z_+^n : d_{kl} \sum_{i=1}^{n} {}_i C_i > 0;$ and $d_{kl} \sum_{i=1}^{n} {}_i C_i \ 2 f C_1; : : : ; C_n g$.

Also, the set F (set of all possible fractional production levels) is rede ned as $F^0 := \quad f_u :$ $f_u = d_{kl} \sum_{i=1}^{n} {}_i^u C_i$ for all $^u \ 2 \quad ^0$. In order to compute the minimum cost with fractional period for MCLS-S, recursive equation (6) has to be modi ed to incorporate the subcontracting option. Speci cally, in each fractional period t, we can either produce the fractional quantity (f_u) `in-house' or subcontract it. Let $Q_t(f_u)$ be the cost incurred to produce/subcontract the fractional quantity f_u in period t which is equal to

$$
Q_t(f_u) = \begin{cases}
\min \quad p_t^{v_{f_u}}(f_u) + q_t^{v_{f_u}}; g_t(f_u) \quad ; & \text{if } 0 < f_u < C_n \\
g_t(f_u); & \text{if } f_u > C_n \text{ or } v_{f_u} \ 2 \ S:
\end{cases}
$$

If $v_{f_u} \ 2 \ S$ and $p_t^{v_{f_u}}(f_u) + q_t^{v_{f_u}} < g_t(f_u)$, we also set $Q_t(f_u)$ to $g_t(f_u)$, since full production

and fractional production cannot be performed on the same machine in the same time period. This leads to the following modified forward recursion to compute the values of $G_k^u(t; \ell; 1)$, $t \in \{k, \ldots, l\}$, $\ell \in \mathbb{Z}_+^n$, and $f_u \in F^0$, for MCLS-S:

$$G_k^u(t; \ell; 1) = \begin{cases} \infty & \text{(6a); (6b); (6c); (6d)} \\ \left(\min_{\substack{S \subseteq N: \\ v_{f_u} \in N_u}} G_k(t-1; \ell - \sum_{i \in S} e_i; 0) + \sum_{i \in S} p_t^i(C_i) + q_t^i \right) + Q_t(f_u) + h_t \sum_{i=1}^n \ell_i C_i + f_u d_{kt} & \text{min (6e); otherwise,} \end{cases} \tag{7}$$

where $N_u := \{v_{f_u} \in N : f_u < C_{v_{f_u}}\}$ is a subset of machines whose capacity is greater than or equal to the fractional production f_u.

3.5 Optimal solution and complexity analysis

After computing the minimum costs with at most one fractional period for all possible fractional production levels, we find the overall minimum cost α_{kl} for each interval $[k; l]$ as follows:

$$\alpha_{kl} = \begin{cases} \min_{u \in U} \{G_k^u(l; \ell^u; 1)\} & \text{for } 1 \le k \le l < T; \\ \min \left\{ \min_{u \in U} \{G_k^u(l; \ell^u; 1)\}, \min_{\ell \in \{0, \ldots, T\}^n} \{G_k(l; \ell; 0)\} \right\} & \text{for } 1 \le k \le l = T. \end{cases}$$

Note that the above equation computes α_{kl} for MCLS. In order to compute the minimum cost α_{kl} for MCLS-S, we just replace ℓ with 0, and incorporate the modifications discussed in Section 3.4. Once we compute these minimum costs for all possible intervals $[k; l]$ with $1 \le k \le l \le T$, the overall problem can be solved using phase (II) as described at the beginning of this section. We denote our DP algorithm for MCLS and MCLS-S by DP-MCLS and DP-MCLS-S, respectively, and for convenience, we denote both of these algorithms collectively by DP-MCLS-(S).

Theorem 2. Algorithm DP-MCLS-(S) solves the multi-capacitated lot-sizing problem with concave production, holding, and subcontracting cost functions, i.e., MCLS-(S) in $O(T^{2n+3})$ time, where T is the number of time periods in the planning horizon and n is the number of machines.

Proof. Refer to Appendix B.

Remark 2. Note that MCLS with $n = 1$ reduces to CLSP. For this special case, our algorithm runs in $O(T^4)$ time because of the following reasons. For each regeneration interval $[k; l]$, Steps 1 and 3 take $O(T^2)$ time. However, in the case of CLSP, there is only one fractional production

level possible, i.e. $F = d_{kl} - b\frac{d_{kl}}{C_1}c$. Hence, Step 2 runs in constant time. As a result $_{kl}$ is computed in $O(T^2)$ time for each regeneration interval $[k;l]$. Since there are $O(T^2)$ possible regeneration intervals, DP-MCLS-S solves CLSP in $O(T^4)$ time which is same as the running time of the algorithm developed by Florian and Klein (1971).

3.6 LS-PC-S Reformulation of MCLS-S

The MCLS-S problem with n capacities can be reformulated as the LS-PC-S problem where the breakpoints of the production cost functions are determined by taking all possible binary combinations of the n available capacities. Depending on the value of capacities, the number of breakpoints (m) can vary from n to $2^n - 1$. This implies that solving the MCLS-S problem using aforementioned DP algorithms for LS-PC-S will take between $O(T^{2n+3})$ to $O(T^{2^{n+1}+1})$ time. On the other hand, as discussed above, our DP algorithm for the MCLS-S problem (presented in Section 3) always takes $O(T^{2n+3})$ time. Notice that the number of breakpoints (m) is same as the number of capacities (n), i.e. $m = n$, only when the size of all the capacities is same, i.e., $C_i = C$ for $i = 1;\ldots;n$. This means that for solving MCLS-S where all the capacities are equal, the algorithms for the LS-PC-S reformulation of MCLS-S take $O(T^{2n+3})$ time. Apart from this special case, our algorithm for MCLS-S always outperforms the algorithms for its LS-PC-S reformulation. Similarly, a special case of MCLS-S with linear cost functions can be reformulated as LS-PC-S with piecewise linear transportation cost function with $2^n - 1$ breakpoints. As a result, the algorithm of Ou (2017) takes $O(T^{2^n+1}\log T)$ time to solve this special case of MCLS-S, in comparison to $O(T^{2n+3})$ time algorithm.

3.7 Optimization Over Structured Mixed Integer Sets using MCLS-S

We showcase how the algorithms presented for MCLS-S address the existence of polynomial time algorithms for optimization over a multi-constrained continuous mixed integer knapsack set: n-mixing set introduced by Sanjeevi and Kianfar (2012), i.e.,

$$P^{n;T} := \{(\ ;\ ) \in R_+ \quad Z_+^{nT} : \ + \sum_{i \in N} C_i \ _t^i \quad d_{1t} \text{ for } t \in T \ : \tag{8}$$

Günlük and Pochet (2001) provided the convex hull description of $P^{1;T}$ (i.e., mixing set). Miller and Wolsey (2003) also provided a tight extended formulation for $P^{1;T}$ and proved that

$$\text{conv}\left(P^{1;T} \setminus \{0 \quad _t^1 \quad _{t\ 1}^1 \quad 1 \text{ for all } t\}\right) = \text{conv}(P^{1;T}) \setminus \{0 \quad _t^1 \quad _{t\ 1}^1 \quad 1 \text{ for all } t\} :$$

Sanjeevi and Kianfar (2012) and Bansal and Kianfar (2015) provided facet-defining inequalities for $P^{n;T}$ when the capacities $\{C_i\}_{i \in N}$ satisfy so-called "n-step MIR" conditions. For $P^{n;T}$ and

general capacities $\{C_i\}_{i \in N}$, Bansal and Kianfar (2017) also derived another class of facet-defining inequalities. However, even for fixed $n \geq 2$, the convex hull description of $P^{n,T}$ is not known so-far. Also, it is unknown in the literature whether the following problem P with fixed $n \geq 2$ can be solved in polynomial time or not:

$$\min \quad c_1 + \sum_{t \in T} \sum_{i \in N} c^i_{2,t} \, \alpha^i_t : (z, \alpha) \in P^{n,T} \setminus \{0\} \qquad \alpha^i_t \geq \alpha^i_{t-1} \quad \text{for } i \in N, t \in T \qquad (P)$$

where $c_1, c^i_{2,t} \in R$. Let $z_1 = \alpha^i_1$, $z_t = 0$ for $t = 2, \ldots, T$, and $y^i_t = \alpha^i_t - \alpha^i_{t-1}$, for $t \in T$ and $i \in N$. Notice that $z_t \in R_+$, and $y^i_t \in \{0, 1\}$. By substituting,

$$\alpha^i_t = \sum_{j=1}^{t} z_j, \quad \alpha^i_t = \sum_{j=1}^{t} y^i_t, \qquad (9)$$

in problem (P), we obtain the following reformulation:

$$\text{Minimize} \quad c_1 z_1 + \sum_{t \in T} \sum_{i \in N} c^i_{2,t} \sum_{j=1}^{t} y^i_j \qquad (10a)$$

$$\text{s.t.} \quad \sum_{j=1}^{t} z_j + \sum_{i \in N} C_i \sum_{j=1}^{t} y^i_j \geq \sum_{j=1}^{t} d_j, \; t \in T; \qquad (10b)$$

$$y_t \in \{0, 1\}^n; z_t \geq 0, \; t \in T. \qquad (10c)$$

Upon adding non-negative surplus variables $\{s_t\}_{t \in T}$ to constraints (10b) we get

$$\sum_{j=1}^{t} z_j + \sum_{i \in N} C_i \sum_{j=1}^{t} y^i_j = \sum_{j=1}^{t} d_j + s_t.$$

We can now subtract the constraint (10b) for period $t-1$ from the constraint (10b) for period t to obtain $z_t + \sum_{i \in N} C_i y^i_t = d_t + s_t - s_{t-1}$. Also, we let $x^i_t = C_i y^i_t$ for $i \in N$ and $t \in T$ and rearrange cost coefficients to obtain the following reformulation of problem (P):

$$\text{Minimize} \quad c_1 z_1 + \sum_{t \in T} \sum_{i \in N} c^i_{2,t} y^i_t \qquad (11a)$$

$$\text{s.t.} \quad z_t + \sum_{i \in N} x^i_t + s_{t-1} = d_t + s_t, \; t \in T; \qquad (11b)$$

$$\sum_{i \in N} x^i_t = \sum_{i \in N} C_i y^i_t, \; t \in T \qquad (11c)$$

$$y_t \in \{0, 1\}^n; x^i_t; z_t \geq 0, \; t \in T; \qquad (11d)$$

where $z_t = 0$ for $t = 2, \ldots, T$.

Notice that the above reformulation is a special case of the problem MCLS-S where cost functions are linear, only full capacitated production is allowed in-house, and subcontracting is allowed

only in the first time period. We can solve this problem by slightly modifying the algorithm for solving MCLS-S. More specifically, for each regeneration interval $[1,l]$ for $l \in T$, we compute the minimum costs without a fractional period using the recursive equation (5). We then compute the set F^0 of all possible fractional production levels within the regeneration interval $[1,l]$ as discussed in Section 3.4. Since fractional quantity can neither be produced in-house nor be subcontracted during any period $t \geq 2$, we set the value of cost function $Q_t(f_u)$ to infinity for $t \geq 2$. Moreover, for $t = 1$, since the fractional quantity must be subcontracted, we set $Q_1(f_u) = c_1 f_u$ for all $f_u \in F^0$ and $N_u = \emptyset$. Finally, we compute the minimum cost with a fractional period for the regeneration interval $[1,l]$ using the recursive equation (7). Here, we denote the overall optimal cost of a regeneration interval $[1,l]$ by $^1_{1,l}$ which can be computed using the expression for $_{kl}$ in Section 3.5.

Note that within the optimal solution, in addition to the regeneration interval $[1,l]$ for some $l \in T$, there will also be a subsequent semi-regeneration interval $[l+1,T]$ where no fractional production takes place. Therefore, we also need to compute the minimum cost of satisfying the demands without a fractional period for the interval $[l+1,T]$ for every $l \in \{0,\ldots,T-1\}$. For each semi-regeneration interval $[l+1,T]$, we compute $G_{l+1}(t,\alpha,0)$ for $t \in \{l+1,\ldots,T\}$ and $\alpha \in \{0,\ldots,T\}$ using the recursive equation (5). The overall optimal cost of the semi-regeneration interval can be computed by finding the minimum of $G_{l+1}(T,\alpha,0)$ over all possible α vectors. We denote the optimal cost of semi-regeneration interval $[l+1,T]$ by $^0_{l+1,T}$.

Finally, the overall optimal cost of the problem (P) is equal to $\min\{^1_{1,l} + {}^0_{l+1,T} : l \in \{0,\ldots,T\}\}$ where $^1_{1,0} = {}^0_{T+1,T} = 0$. Interestingly, since we need to consider only $O(T)$ possible intervals across the planning horizon, the running time of this modified algorithm is $O(T^{2n+2})$. This implies that an optimal solution $\{x_t, y^i_t, s_t, z_t\}_{t \in T}$ obtained upon solving the aforementioned modified algorithm for MCLS-S can be utilized along with (9) to compute an optimal solution $(\alpha, \{\beta^i_t\}_{i \in N, t \in T})$ for problem (P) with fixed n, in polynomial time.

4 New Exact Algorithm for LS-PC-S Problem

In this section, we present a new DP algorithm for the LS-PC-S that explores a reduced search space using recursive equations with reduced number of state variables, thereby leading to 93.75% reduction in the run time of the first and the only known DP algorithm (Koca et al. 2014) for this problem. Moreover, this approach takes only 10% of the time taken by Gurobi to solve binary formulation of LS-PC-S by Croxton et al. (2003).

De nition 5 (Fractional Period for LS-PC-S). For LS-PC-S, a period is called a fractional period if the amount produced in that period is not equal to any of the breakpoints of the production cost function. In other words, period $t \in T$ is a fractional period if $x_t \neq \{b_0, b_1, \ldots, b_m\}$.

Theorem 3 (Koca et al. (2014)). There exists an optimal solution to the LS-PC-(S) which comprises of: (a) a series of regeneration intervals, each having at most one fractional period, that span the interval $[1, T']$ for some $0 \leq T' \leq T$, and (b) a semi-regeneration interval $[T'+1, T]$ with no fractional period.

For a given $m \in Z_+$, we present a DP algorithm for LS-PC-S which also comprises of two phases, i.e., first computing optimal costs $_{kl}$ for all (semi)-regeneration intervals $[k, l]$ where $1 \leq k \leq l \leq T$, and then finding the least cost route from node 1 to $T+1$ in a directed acyclic graph $G = (V, E)$ where $V = (1, \ldots, T+1)$ and $E = \{(k, l+1) : 1 \leq k \leq l \leq T\}$, with the cost $_{kl}$ assigned to arc $(k, l+1)$. Again, we take advantage of the structure of the optimal solution of the LS-PC-S provided in Theorem 2 to compute the following:

1. Minimum costs for satisfying the demands in each period of (semi)-regeneration interval $[k, l]$ with no fractional period;
2. All possible fractional production levels in the regeneration interval $[k, l]$;
3. Minimum costs for satisfying the demands in each period of regeneration interval $[k, l]$ with at most one fractional period.

Observe that though the aforementioned steps look similar to our DP algorithm for MCLS-S, the computation of $_{kl}$ for LS-PC-S and associated recursive functions are completely different (recall that LS-PC-S subsumes MCLS-S). We denote $_i$ as a unit vector of size m in which only the ith element is one and the rest are zero. Let $\in Z_+^m$ be a vector of $_i$'s where $_i$ is the number of times production takes place at a level b_i for $i \in \{1, \ldots, m\}$. For $t \in \{k, \ldots, l\}$, let $G_k(t, , 0)$ be a function that denotes the value of minimum cost solution for periods k up to t during which production of b_i, $i \in \{1, \ldots, m\}$, units occurs $_i$ times, and no fractional production takes place. The computation of $G_k(t, , 0)$ using our algorithm is similar to the DP algorithm of Koca et al. (2014), except that $_i \in \{0, \ldots, \min\{T, \lfloor d_{kl}/b_i \rfloor\}\}$ for $i \in \{1, \ldots, m\}$ in the former case, whereas $_i \in \{0, \ldots, T\}$ for $i \in \{1, \ldots, m\}$ in the latter.

In the ensuing sections, we present Steps 2 and 3 of our DP algorithm using the new concept of fractional production levels. More specifically, in Koca et al. (2014), recursive function implicitly compute the fractional production levels using additional state variables $\in Z^m$ which denotes

the amount to be produced in future (see function $G_k(t; ; ; 1)$ in Appendix E for details). In contrast, our algorithm takes advantage of the observation that the fractional production levels are solely dependent on the values of breakpoints and the total demand in the regeneration interval and not on the future production. Therefore, we do not keep track of possible future production amounts as it is not necessary in our algorithm, but is a key requirement in the algorithm by Koca et al. (2014) for LS-PC-S. This results in a reduction of search space (see an example in Appendix E) and state variables in the recursive equations (see Section 4.2).

4.1 Calculating possible fractional production levels

We de ne $^{max} 2 Z_+^m$ as a vector of $_i^{max}$'s where $_i^{max}$ is the maximum number of times production at a level b_i can be done in the interval $[k; l]$. Clearly, $_i^{max} = \min f T; bd_{kl}=b_i cg$ for $i = 1; : : : ; m$, where $d_{kl} = \sum_{j=k}^{l} d_j$. We also de ne $:= \quad 2 Z_+^m : d_{kl} \sum_{i=1}^{m} {}_i b_i > 0;$ and $d_{kl} \sum_{i=1}^{m} {}_i b_i 2 f b_1; : : : ; b_m g$. The set of all possible fractional production levels in a regeneration interval $[k; l]$ is given by $F := \quad f_r : f_r = d_{kl} \sum_{i=1}^{m} {}_i^r b_i$ for all $^r 2$. For a given $^r 2$, we can compute the fractional production level in constant time and since $_i^r \quad _i^{max} \quad T$, there are $O(T^m)$ possible vectors of ^{r}. This implies we can nd all possible fractional production levels in $O(T^m)$ time.

4.2 Minimum cost with at most one fractional period

Next, we compute the value of minimum cost for satisfying demands in a regeneration interval $[k; l]$ with at most one fractional period. Again, we consider only regeneration intervals because in Step 1, the optimal costs of all semi-regeneration intervals with no fractional period have already been computed. For each fractional production level $f_r 2 F$ and $t 2 f k; : : : ; l g$, let function $G_k^r(t; ; 1)$ denote the value of minimum cost solution for periods k up to t during which b_i, for $i = f 1; : : : ; m g$, units are produced in $_i$ time periods, and the fractional production takes place at most once. We set $G_k^r(t; ; 1) = 1$ whenever $\sum_{i=1}^{m} {}_i > t \quad k$ or $\sum_{i=1}^{m} {}_i b_i + f_r \quad d_{kt}$ for $t \quad l \quad 1$, or $\sum_{i=1}^{m} {}_i b_i + f_r 6 d_{kl}$ for $t = l$. It should be noted that if fractional production level is zero, the function $G_k^r(t; ; 1)$ is equal to $G_k(t; ; 0)$, as there is no fractional production. Now, in order to ensure that the fractional production f_r occurs at most once throughout the regeneration interval $[k; l]$, our goal is to nd the minimum among three possible choices at every time period $t 2 f k; : : : ; l g$: (i) choose not to produce at all, (ii) produce up to b_i units for any $i 2 f 1; : : : ; m g$, and (iii) produce at a fractional production level.

If we choose to produce nothing in period t, the optimal value of the function $G_k^r(t; \ ; 1)$ will be equal to the optimal value of the function at time period $t-1$ plus the holding costs at the end of time period t, i.e., $G_k^r(t; \ ; 1) = G_k^r(t-1; \ ; 1) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right)$. On the other hand, if we choose to produce up to b_i units, $i = \{1, \ldots, m\}$, during period t, then the value of the function $G_k^r(t; \ ; 1)$ will be equal to the sum of function value at $t-1$ and α_i, the cost of producing b_i units in period t which would be $p_t(b_i)$, and the holding costs at the end of period t, i.e., $G_k^r(t; \ ; 1) = G_k^r(t-1; \ \alpha_i; 1) + p_t(b_i) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right)$. Lastly, as per choice (iii), if we choose to produce at fractional production level $f_r \in F$ in period t, the value of the function $G_k^r(t; \ ; 1)$ will be equal to the sum of the value of minimum cost for satisfying demands in the interval $[k; t-1]$ with no fractional period at $t-1$ and , i.e. $G_k(t-1; \ ; 0)$, the cost of producing f_r units which would be $p_t(f_r)$, and the holding cost at the end of period t, i.e. $G_k^r(t; \ ; 1) = G_k(t-1; \ ; 0) + p_t(f_r) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right)$. In a nutshell, we compute the values of $G_k^r(t; \ ; 1)$ for all $t \in \{k, \ldots, l\}$, $\alpha \in Z_+^m$, and $f_r \in F$ using the following forward recursion:

$$G_k^r(t; \ ; 1) = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} \alpha_i + 1 \leq (t-k+1) \\ & \text{or } \sum_{i=1}^{m} \alpha_i b_i + f_r \leq d_{kt} \text{ for } t = l-1; \text{ or } \sum_{i=1}^{m} \alpha_i b_i + f_r < d_{kl} \text{ for } t = l \\ G_k(t; \ ; 0), & \text{if } d_{kl} \leq \sum_{i=1}^{m} \alpha_i b_i \\ \min \begin{cases} G_k^r(t-1; \ ; 1) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right) \\ G_k(t-1; \ ; 0) + p_t(f_r) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right) \\ \min_{i \in \{1, \ldots, m\}} G_k^r(t-1; \ \alpha_i; 1) + p_t(b_i) + h_t \left(\sum_{i=1}^{m} \alpha_i b_i + f_r - d_{kt} \right) \end{cases} & \text{otherwise.} \end{cases}$$

4.3 Optimal solution and complexity analysis

After computing the minimum costs with at most one fractional period for all possible fractional production levels, we find the overall minimum cost γ_{kl} for the interval $[k; l]$ using

$$\gamma_{kl} = \begin{cases} \min_{r \in 2} \{G_k^r(l; \ ^r; 1)\}; & \text{for } 1 \leq k \leq l < T; \\ \min \left\{ \min_{r \in 2} \{G_k^r(l; \ ^r; 1)\}; \ \min_{\in \{0, \ldots, T\}^m} \{G_k(l; \ ; 0)\} \right\}; & \text{for } 1 \leq k \leq l = T. \end{cases}$$

Once we compute these minimum costs for all possible intervals $k[l]$ with $1 \leq k \leq l \leq T$, the overall problem is solved by constructing a directed acyclic graph as described at the beginning

of this section and nding a least cost route from node 1 to $T + 1$. We denote this new DP algorithm for LS-PC-S by New-DP-LSPCS

Theorem 4. Algorithm New-DP-LSPCS solves the lot-sizing problem with piecewise concave production cost functions and concave holding and subcontracting cost functions in $O(T^{2m+3})$ time, where T is the number of time periods in the planning horizon and m is the number of breakpoints in the production cost function.

Proof. Refer to Appendix C. □

Remark 3. (Parallel Computing Implementation) Observe that a salient feature of the design of our algorithms for LS-PC-S and MCLS-(S) is that it allows parallel computing implementation with respect to the fractional production levels. More speci cally, the recursive equations to compute the minimum costs for a given fractional production level is independent of the recursive equations for a di erent fractional production level. Therefore, these computations can be performed in parallel. This results in improving the performance of the proposed algorithm by 5.6 times on average (refer to Section 6 for more details). Note that the design of algorithm by Koca et al. (2014) for LS-PC-S does not allow such parallel implementation because of the implicit computation of the fractional production levels.

5 New Exact Algorithm for 2E-MCLS Problem

In this section, we present a dynamic programming algorithm for the two-echelon lot-sizing problem with multiple production capacities in each time period, de ned by (3). The initial and nal inventories in each of the echelons are assumed to be zero i.e., $s^e_0 = 0$ and $s^e_T = 0$ for $e \in \{1; 2\}$ as considered widely in the literature (van Hoesel et al. 2005, Florian and Klein 1971, Zangwill 1969). We rst present some properties of an optimal solution of 2E-MCLS using the following de nitions. Recall that we refer to time period $t \in T$ in echelon $e \in \{1; 2\}$ as \node $(e; t)$" (as shown in Figure 1).

De nition 6. Subplan$[a_1; a_2; b_1; b_2]$ is a collection of nodes $(1; a_1); \ldots; (1; b_1)$, $(2; a_2); \ldots; (2; b_2)$, where $1 \le a_1 \le a_2 \le b_1 \le b_2 \le T$ and $a_e; b_e \in T$ for $e \in \{1; 2\}$, such that $s^1_{a_1\ 1} = 0$, $s^2_{a_2\ 1} = 0$, $s^1_{b_1} = 0$, $s^2_{b_2} = 0$, and at most one among $s^1_{a_1}; \ldots; s^1_{b_1\ 1}$, $s^2_{a_2}; \ldots; s^2_{b_2\ 1}$ is equal to zero. Furthermore, Subplan$[a_1; a_2; b_1; b_2]$ and Subplan$[b_1 + 1; b_2 + 1; c_1; c_2]$ are referred to as consecutive subplans.

De nition 7 (Fractional Period for 2E-MCLS) . A period t is a fractional period if $0 < x_t^{1;j} < C_j$ for some $j \in N$, and $x_t^{1;i} \in \{0; C_i\}$ for $i \in N \setminus \{j\}$.

Proposition 1. There exists an optimal solution to the problem 2E-MCLS that comprises of a series of consecutive subplans where within each Subplan $[a_1; a_2; b_1; b_2]$: (A) there is at most one fractional period, and (B) the cumulative transportation quantity from period a_2 up to period $t \in \{a_2; \ldots; b_2\}$ is either equal to total production in periods $\hat{t}_1; \ldots; t$, i.e., $\sum_{j=\hat{t}_1}^{t} \sum_{i=1}^{n} x_j^{1;i}$, for some $\hat{t}_1 \in \{a_1; \ldots; t\}$ or $d_{t;\hat{t}_2}$ for some $\hat{t}_2 \in \{t; b_2\}$.

Proof. Akin to the network ow representation of MCLS in Figure 2, we can represent 2E-MCLS with n capacitated ow arc going from the source node to node $(1; t)$ for each $t \in T$. Then, (A) follows from the free arc property by Ahuja et al. (1988), and the proof for (B) is same as the proof of Proposition 2.3 of van Hoesel et al. (2005) for 2E-MCLS with $n = 1$. □

5.1 Computing optimal cost of a given subplan

We harness the above proposition by rst computing the minimum cost of planning the production and transportation schedule for each Subplan $[a_1; a_2; b_1; b_2]$ within the planning horizon. For notational convenience, we denote $[a_1; a_2; b_1; b_2]$ by .

De nition 8. Similar to MCLS, we de ne a set of fractional production levels $F := \{f^u = d_{a_2;b_2} - \sum_{i=1}^{n} {}_i^u C_i$ for all $u \in \}$, where $= \{u : 0 < d_{a_2;b_2} - \sum_{i=1}^{n} {}_i^u C_i < C_n$ and $d_{a_2;b_2} - \sum_{i=1}^{n} {}_i^u C_i \in \{C_1; \ldots; C_{n-1}\}\}$.

De nition 9. Let Y_t be a set of possible cumulative transportation quantity from period a_2 to period t in the subplan. Since transportation from the rst echelon to the second echelon can only occur between periods a_2 and b_1, $Y_t = \{0\}$ for $a_1 \le t < a_2$, and $Y_t = \{d_{a_2;b_2}\}$ for $b_1 \le t \le b_2$. In contrast, for $t \in \{a_2; \ldots; b_1 - 1\}$, $Y_t = Y_t^1 \cup Y_t^2 \cup Y_t^3$ where

$$Y_t^1 = \{d_{t;t}; \ldots; d_{t;b_2}\}; \quad Y_t^2 = \Big\{ \sum_{i=1}^{n} {}_i C_i : {}_i \in \{0; \ldots; {}_i^u\} \text{ for all } i \in N; {}^u \in \ \}; \text{ and}$$

$$Y_t^3 = \Big\{ \sum_{i=1}^{n} {}_i C_i + f^u : f^u = d_{a_2;b_2} - \sum_{i=1}^{n} {}_i^u C_i \text{ and } {}_i \in \{0; \ldots; {}_i^u\} \text{ for all } i \in N; {}^u \in \ \}.$$

Minimum cost with no fractional period . We now de ne a function $F(t; ; 0; {}_t)$ which computes the minimum cost of producing $\sum_{i=1}^{n} {}_i C_i$ units from period a_1 to period $t \in \{a_1; \ldots; b_2\}$ and transporting ${}_t$ units from periods a_2 upto t. Note that ${}_t \in Y_t$ for $t \in \{a_1; \ldots; b_2\}$ such

that $_t \quad d_{a_2;t}$ (demand must be met) and $_t \quad \sum_{i=1}^{n} {}_i C_i$ (cannot transport more than what has been produced). Therefore, if the foregoing conditions do not hold, we set the value of the cost function to in nity. In addition to this, we also set the value of the function to in- nity for the following infeasible cases: (i) $\sum_{i=1}^{n} {}_i C_i \quad d_{a_2;t}$ since the demand is not satis ed, (ii) $_i > t \quad a_1 + 1$ for $i \in N$, and (iii) $_t \in Y_t^3$ as we do not consider fractional production.

In period t, we have two choices for production: (a) choose not to produce at all, or (b) setup some nonempty subset of n machines and run them to produce only at full capacity. In addition, we have two choices for transportation in the period t: (a) choose not to transport at all, i.e., $_t = _{t\ 1}$, or (b) transport $_t \quad _{t\ 1} > 0$ units where $_{t\ 1} \in Y_{t\ 1}$, which costs $p_t^2(\ _t \quad _{t\ 1}) + q_t^2$. Also, inventory at the end of period t in the rst echelon and second echelon is equal to $\sum_{i=1}^{n} {}_i C_i \quad _t$ and $_t \quad d_{a_2;t}$, respectively. If we choose to produce on some subset $S \subseteq N$ of machines at full capacity, and transport strictly positive $_t \quad _{t\ 1}$ units for some $_{t\ 1} \in Y_{t\ 1}$ during period t, the value of the cost function $F(t; \ ; 0; \ _t)$ in this case is equal to the sum of the following: (a) value of the cost function $F(t \quad 1; \ \sum_{i \in S} e_i; 0; \ _{t\ 1})$, (b) Cost of setting up and producing on the set S of machines at full capacity, i.e. $\sum_{i \in S} p_t^{1;i}(C_i) + q_t^i$, (c) Cost of transporting $_{t\ 1} \in Y_{t\ 1}$ units, i.e. $p_t^2(\ _t \quad _{t\ 1}) + q_t^2$, and (d) the holding costs incurred in both the echelons at then end of period t, i.e. $h_t^1(\sum_{i=1}^{n} {}_i C_i \quad _t) + h_t^2(\ _t \quad d_{a_2;t})$. Based on the aforementioned observations, we provide the following recursive equation for a given subplan that computes $F(t; \ ; 0; \ _t)$ for all $t \in \{a_1; \ldots; b_2\}$, $_i \in \{0; \ldots; \ _i^{max}\}$ for $i \in N$, and $_t \in Y_t$:

$$F(t; \ ; 0; \ _t) = \begin{cases} 1 \ ; & \text{if } _i > t \quad a_1 + 1 \text{ for any } i \in \{1; \ldots; n\} \\ & \text{or } \sum_{i=1}^{n} {}_i C_i \quad d_{a_2;t} \text{ for } t \quad b_1 \quad 1; \text{ or } \sum_{i=1}^{n} {}_i C_i \quad d_{a_2;b_2} \text{ for } t = l \quad T \quad 1; \\ & \text{or } \sum_{i=1}^{n} {}_i C_i < d_{a_2;t} \text{ for } t = l \text{ and } l = T \\ & \text{or } _t \in Y_t^3 \text{ for } t \in \{a_1; \ldots; b_2\} \text{ or } _t > \sum_{i=1}^{n} {}_i C_i \text{ for } t \in \{a_1; \ldots; b_2\} \\ \min\limits_{S \subseteq N} \Big\{ & F(t \quad 1; \ \sum_{i \in S} e_i; 0; \ _t) + \sum_{i \in S} p_t^{1;i}(C_i) + q_t^i \\ & + h_t^1(\sum_{i=1}^{n} {}_i C_i \quad _t) + h_t^2(\ _t \quad d_{a_2;t}) \\ \min\limits_{_{t\ 1} \in Y_{t\ 1}: \ _{t\ 1} < \ _t} \Big\{ & F(t \quad 1; \ \sum_{i \in S} e_i; 0; \ _{t\ 1}) + \sum_{i \in S} p_t^{1;i}(C_i) + q_t^i \\ & + p_t^2(\ _t \quad _{t\ 1}) + q_t^2 + h_t^1(\sum_{i=1}^{n} {}_i C_i \quad _t) + h_t^2(\ _t \quad d_{a_2;t}) \Big\} \Big\} \ ; & \text{otherwise.} \end{cases} \tag{12}$$

Minimum cost with a fractional period. Given a fractional production level $f^u \in F$, we now define $F_u(t; \cdot; 1; \cdot_t)$ as a cost function that computes the minimum cost of producing $\sum_{i=1}^{n} \cdot_i C_i + f^u$ units from periods a_1 through t, and transporting $\cdot_t$ units from periods a_2 through t. The argument `1' in the cost function denotes that the fractional production has occurred between periods a_1 up to t. Let N_u be the set of indices of machines on which fractional quantity f^u can be produced, i.e. $N_u := \{i \in N : f^u < C_i\}$. Also, let $v_{f^u} \in N_u$ denote the index of the machine on which f^u units can be produced. Similar to MCLS, in each time period t within the subplan, we can either (a) choose not to produce at all, or (b) produce on some subset S of the n available machines at full capacity, or (c) produce f^u units in period t on machine v_{f^u} and utilize a subset $S \subseteq N \setminus \{v_{f^u}\}$ of the $n-1$ machines at full capacity. In addition to these first echelon choices, in any given period t, we can also choose to either not transport at all, or transport $\cdot_t - \cdot_{t-1} > 0$ units for some $\cdot_{t-1} \in Y_{t-1}$. The overall inventory costs incurred during period t is equal to $\cdot_t = h_t^1(\sum_{i=1}^{n} \cdot_i C_i + f^u - \cdot_t) + h_t^2(\cdot_t - d_{a_2;t})$ where $\sum_{i=1}^{n} \cdot_i C_i + f^u - \cdot_t$ and $\cdot_t - d_{a_2;t}$ is the inventory at the end of period t in first and second echelon, respectively. Based on the discussion above, we present a forward recursive equation to compute $F_u(t; \cdot; 1; \cdot_t)$ for all $f^u \in F$, $t \in \{a_1; \ldots; b_2\}$, $\cdot_i \in \{0; \ldots; \cdot_i^{\max}\}$ for $i \in N$, and $\cdot_t \in Y_t$:

$$
F_u(t; \cdot; 1; \cdot_t) = \begin{cases}
1; & \text{if } \cdot_i > t - a_1 + 1 \text{ for any } i \in \{1; \ldots; n\} \quad (13a)\\
& \text{or } \sum_{i=1}^{n} \cdot_i C_i + f^u > d_{a_2;t} \text{ for } t < b_1; \quad (13b)\\
& \text{or } \sum_{i=1}^{n} \cdot_i C_i + f^u \leq d_{a_2;b_2} \text{ for } t \geq b_1; \quad (13c)\\
& \text{or } \cdot_t > \sum_{i=1}^{n} \cdot_i C_i + f^u \text{ for } t \in \{a_1; \ldots; b_2\} \quad (13d)\\
\min\limits_{S \subseteq N} \Big(F_u(t-1; \cdot; 1; \cdot_t + d_{a_2;t}) + \sum_{i \in S} e_i(\cdot_i; 1; \cdot_t + d_{a_2;t}) + \sum_{i \in S} p_t^{1;i}(\cdot_i) + q_t^j + \cdot_t \Big); & (13e)\\
\min\limits_{S \subseteq N \setminus \{v_{f^u}\}} \Big(F_u(t-1; \cdot; 1; \cdot_t + d_{a_2;t}) + \sum_{i \in S} e_i(\cdot_i; 0; \cdot_t + d_{a_2;t}) + \sum_{i \in S \cup \{v_{f^u}\}} p_t^{1;i}(\cdot_i) + q_t^j + \cdot_t \Big); & (13f)\\
\min\limits_{\substack{S \subseteq N \\ \cdot_{t-1} \in Y_{t-1}: \\ \cdot_{t-1} < \cdot_t}} \Big(F_u(t-1; \cdot; 1; \cdot_{t-1} + d_{a_2;t}) + \sum_{i \in S} e_i(\cdot_i; 1; \cdot_{t-1} + d_{a_2;t}) + \sum_{i \in S} p_t^{1;i}(\cdot_i) + q_t^j & \\
\quad + p_t^2(\cdot_t - \cdot_{t-1}) + q_t^2 + \cdot_t \Big); & (13g)\\
\min\limits_{\substack{v_{f^u} \in N_u \\ S \subseteq N \setminus \{v_{f^u}\} \\ \cdot_{t-1} \in Y_{t-1}: \\ \cdot_{t-1} < \cdot_t}} \Big(F_u(t-1; \cdot; 1; \cdot_{t-1} + d_{a_2;t}) + \sum_{i \in S} e_i(\cdot_i; 0; \cdot_{t-1} + d_{a_2;t}) + \sum_{i \in S \cup \{v_{f^u}\}} p_t^{1;i}(\cdot_i) + q_t^j & \\
\quad + p_t^2(\cdot_t - \cdot_{t-1}) + q_t^2 + \cdot_t \Big); & (13h)\\
\infty, & \text{otherwise.}
\end{cases}
$$

In this recursive equation, (13e) denotes the cases where we choose to produce on a subset $S \subseteq N$ of machines at full capacity, and (13f) denotes the cases where we choose to produce f^u on machine v_{f^u} for some $v_{f^u} \in N_u$ and produce on some subset $S \subseteq N \setminus \{v_{f^u}\}$ of machines at full capacity. In (13e)-(13f), we assume that nothing is transported in period t. In contrast, (13g) and (13h) denote the foregoing cases where we also choose to transport $t_{t-1} > 0$ units in period t. The overall optimal cost $\alpha([a_1, a_2, b_1, b_2])$ of subplan $\gamma = [a_1, a_2, b_1, b_2]$ can be computed using the following expression:

$$\alpha([a_1, a_2, b_1, b_2]) = \min \left\{ \min_{u \in} F_u(b_2, \ ^u, 1, d_{a_2, b_2}), \ \min_{\in \{0, \ldots, T\}^n} F(b_2, \ , 0, d_{a_2, b_2}) \right\}.$$

5.2 Computing overall optimal cost for 2E-MCLS

Upon computing the optimal costs for all possible subplan within the planning horizon, we find an optimal series of consecutive subplans that spans the planning horizon in both echelon using a shortest path algorithm. Specifically, we construct a directed acyclic graph where nodes are labeled in pairs of (a_1, a_2) for $a_1, a_2 \in T$ and edges between the nodes are added with minimum costs of subplans as edge weights. We define a function $OPT(a_1, a_2)$ that computes the overall optimal cost of planning the production schedules from periods $a_1, \ldots, T$ and transportation schedules from periods $a_2, \ldots, T$ where $a_1 \geq a_2$, such that $s^1_{a_1 - 1} = 0$ and $s^2_{a_2 - 1} = 0$. We can then apply following backward recursion to compute $OPT(a_1, a_2)$ for all $a_1, a_2 \in T$:

$$OPT(a_1, a_2) = \min_{\substack{b_1, b_2 \in T \\ a_2 \leq b_1 \leq b_2}} \left\{ OPT(b_1 + 1, b_2 + 1) + \alpha([a_1, a_2, b_1, b_2]) \right\},$$

where $OPT(T + 1, T + 1) = 0$. The overall optimal solution of 2E-MCLS can be obtained by computing the value of $OPT(1, 1)$ which essentially provides the optimal cost of scheduling production and transportation from periods 1 to T. We denote the algorithm to solve 2E-MCLS problem by DP-2E-MCLS.

Remark 4. Observe that computing optimal costs for Subplan $[1, a_2, b_1, b_2]$ also yields optimal costs of Subplan $[a_1, a_2, b_1, b_2]$ for all $a_1 \in \{1, \ldots, a_2\}$, as a byproduct. Therefore, instead of computing optimal cost for all $O(T^4)$ subplans $\gamma = [a_1, a_2, b_1, b_2]$ where $1 \leq a_1 \leq a_2 \leq b_1 \leq b_2 \leq T$, we only need to compute the optimal cost for $O(T^3)$ subplans $\gamma = [1, a_2, b_1, b_2]$ where $1 \leq a_2 \leq b_1 \leq b_2 \leq T$.

Theorem 5. Algorithm DP-2E-MCLS provides an optimal solution for an instance of 2E-MCLS in $O(T^{4n+4})$ time, where T is the number of time periods in the planning horizon and n is the number of machines in each time period.

Proof. To compute optimal cost for a given subplan $= [a_1; a_2; b_1; b_2]$ where $1 \quad a_1 \quad a_2 \quad b_1 \quad b_2 \quad T$, DP-2E-MCLS performs the following steps:

(i) It computes minimum costs without a fractional period i.e. $F(t; ; 0; _t)$ for all $t \in \{a_1; : : : ; b_2\}$, $0 \quad T$ for $i \in N$, and $_t \in Y_t$. For given t, , and $_t$, $F(t; ; 0; _t)$ can be computed in $O(T^n)$ time. Observe that there are $O(T^{2n+1})$ possible permutations of t, , and $_t$ since there are $O(T)$ possible time periods within a subplan, $O(T^n)$ possible vectors, and $O(T^n)$ possible values of $_t$. As a result, all minimum costs without fractional period can be computed in $O(T^{3n+1})$ time.

(ii) There are $O(T^n)$ possible fractional production levels F for a given subplan, and computing each fractional production level f^u takes $O(1)$ time.

(iii) It also computes $F_u(t; ; 1; _t)$ for each $f^u \in F$, t, , and $_t$. Similar to minimum costs without fractional period, $F_u(t; ; 1; _t)$ for given $f^u \in F$, t, , and $_t$ can be computed in $O(T^n)$ time. Since there are $O(T)$ periods in a subplan, $O(T^n)$ possible vectors, $O(T^n)$ possible values of $_t$, and $O(T^n)$ possible fractional production levels, all the minimum cost function values can be computed in $O(T^{4n+1})$ time.

Since optimal costs for only $O(T^3)$ subplans in the planning horizon have to be computed (Remark 4), DP-2E-MCLS algorithm takes $O(T^{4n+4})$ time which is polynomial if n is xed. □

Remark 5. Note that 2E-MCLS with $n = 1$ reduces to 2E-SCLS which was studied by Van Hoesel and Wagelmans (1996). For this special case, DP-2E-MCLS takes $O(T^7)$ time and is same as the algorithm developed by Van Hoesel and Wagelmans (1996). When $n = 1$, there is exactly one fractional production level for a given subplan i.e. $|F| = 1$, and therefore $F(t; ; 0; _t)$ and $F_u(t; ; 1; _t)$ for all t, , and $_t$ can be computed in $O(T^4)$ time. Since we compute the optimal costs for $O(T^3)$ subplans, the overall running time of 2E-MCLS with $n = 1$ is $O(T^7)$.

Remark 6. In this paper, our aim is to provide a xed parameter tractable algorithm to solve the 2E-MCLS problem with xed $n \quad 2$. As per our knowledge, no computational results are known even for algorithm by van Hoesel et al. (2005) for 2E-MCLS with $n = 1$. Conducting extensive computational evaluation of various approaches known for two-echelon lot-sizing problems (van Hoesel et al. 2005, Hwang et al. 2013, 2016, Zhao and Zhang 2020) is a potential future research direction.

6 Computational Experiments

In this section, we evaluate the computational e ciency of our algorithms and their parallel computing implementations to solve MCLS and LS-PC instances, in comparison to the known algorithm(s) in the literature and state-of-the-art solver. The experiments were performed on a workstation with Intel Xeon E5-1660 processor and 32GB RAM. We implemented all DP algorithms using Python 2.7 and used Gurobi 9.1 with its default setting which incorporates parallel computing using all eight cores of the processor. In the rest of the section, we refer to our algorithm for MCLS and LS-PC as DP-MCLS and New-DP-LSPC, respectively, and refer to the algorithm developed by Koca et al. (2014) as Old-DP-LSPC. In Tables 2-7, the value of the objective function at optimality is reported in column labeled as OPT, and the columns labeled as Gurobi 9.1, Old-DP-LSPC, New-DP-LSPC, and DP-MCLS provide the time taken (in seconds) to solve MCLS instances using these algorithms or solver. Since we consider linear production and holding cost functions for our experiments, we also implement a MIP formulation for LS-PC (Croxton et al. 2003) using Gurobi 9.1.

The columns labeled as No-PI and With-PI report the solution time for our DP algorithms without and with parallel computing implementation, respectively. We use CNS to notify that the corresponding instance was not solved within the time limit of 2000 seconds. In Figures 3-6, we present charts to compare time taken (in seconds) by Gurobi and DP-MCLS/New-DP-LSPC with PI. For MCLS, we display solution times of all 5 instances of two selected instance categories denoted by $(T; C_1; \ldots; C_n)$. Similarly, for LS-PC, we display solution times of 5 instances over two instance categories denoted by $(T; b_1; \ldots; b_m)$. Each bar in the figures corresponds to an instance of a given instance category and they are ordered in the same way as done in the corresponding tables. The vertical arrow over certain bars denotes CNS. The newly generated instances will be available at https://github.com/Bansal-ORGroup/MCLS-LSPC-Data .

6.1 Experiments for MCLS with two or three machines in a period

For MCLS with $n = 2$, we consider six sets of $(T; C_1; C_2)$ such that the number of time periods in the planning horizon $T \in \{40; 50; 60\}$ has two sets of capacities associated to it; see column labeled as $(C_1; C_2)$ in Table 2. For MCLS with $n = 3$, we consider four sets of $(T; C_1; C_2; C_3)$ where the number of time periods in the planning horizon is equal to 40 and $C_1; C_2;$ and C_3 are the capacities in each period. For each set of $(T; C_1; \ldots; C_n)$, we generate five instances where for time period $t \in T$, demand d_t, setup cost q_t^1 of machine 1 with capacity C_1, setup cost

q_t^2 of machine 2 with capacity C_2, and setup cost q_t^3 for machine 3 of capacity C_3 (for MCLS with $n = 3$) are obtained as follows. Our instance generation process is motivated from Koca et al. (2014). For LS-PC instances in Koca et al. (2014), the demand in each time period is random integer drawn from Uniform [400, 500] for LS-PC with $m = 2$, and from Uniform [500, 600] for LS-PC with $m = 3$. For MCLS instances, the demand in each period is random integer drawn from Uniform [400, 600], thereby providing more variation in the demand values over the planning horizon. Likewise, Koca et al. (2014) considered time invariant set of fixed costs $(\ ^1; : : : ; \ ^m)$ is equal to (3000, 6000) for LS-PC with $m = 2$ and (3000, 6000, 9000) for LS-PC with $m = 3$. We generate MCLS instances with time varying setup costs where for each time period $t \ 2 \ T$, q_t^1, q_t^2, and q_t^3 are random integer drawn from Uniform [2850, 3150] (mean = 3000), Uniform [5850, 6150] (mean = 6000), Uniform [8850, 9150] (mean = 9000). In addition, the production cost p_t, $t \ 2 \ T$, is a random number drawn from Uniform [0.5, 1.0], and the holding cost $h_t = 0:05$ for all $t \ 2 \ T$.

In Tables 2 and 3, the values of capacities $(C_1; : : : ; C_n)$ were selected by using a trial and error approach during pre-testing such that the capacities are not divisible and the instances are not infeasible. Moreover, we consider capacities such that on average $q_t^1=C_1$, $q_t^2=C_2$, and $q_t^3=C_3$ (ratio of average setup cost and capacity of a machine) are comparable. Note that since q_t^1, q_t^2, and q_t^3 are time varying and randomly drawn, we get a rich variety of instances where in a given time period t, $q_t^i=C_i$ can be either less than, equal to, or greater than $q_t^j=C_j$ for $i \ 6= \ j$.

6.1.1 Computational results for MCLS with $n = 2$.

Based on the computational results reported in Table 2, we make the following observations:

(i) The DP-MCLS algorithm solved each instance in less than 531 seconds and 177 seconds on average, even without parallel implementation. On the other hand, Old-DP-LSPC could not solve any instance with 60 and 80 time periods within the time limit and likewise, New-DP-LSPC could not solve any instances with 80 time periods within the time limit. Moreover, for the remaining instances, DP-MCLS without parallel implementation is faster than Old-DP-LSPC, New-DP-LSPC without parallel implementation, and New-DP-LSPC with parallel implementation by 25, 9, and 1.5 times (on average). The parallel computing implementation of DP-MCLS further reduces the solution time by around 6 times on average.

(ii) The parallel implementation of DP-MCLS algorithm performs significantly better than Gurobi 9.1. The latter could not solve 11 out of the 30 instances within the time limit and solved

the remaining instances in 437 seconds on average, while the former solved each instance in less than 89 seconds and 30 seconds on average. For each set (C_1, C_2), the solution times of our algorithms are highly stable and consistent among all five instances, whereas the variation in Gurobi's solution times is significantly high (ranging from 28 seconds to more than 2000 seconds); see Figure 3.

Table 2: Computational Results for MCLS with n = 2

T	(C_1, C_2)	Instance	OPT	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC		DP-MCLS	
						No-PI	With-PI	No-PI	With-PI
40	(670, 1280)	1	105776.2	51.73	758.42	225.14	36.88	28.62	4.66
		2	102923.9	242.67	759.22	228.7	35.37	27.36	4.46
		3	103685.4	75.81	761.97	222.54	36.54	27.3	4.45
		4	109564.4	319.15	757.44	227.99	36.61	29.88	5.16
		5	102607.9	92.84	760.08	228.1	37.53	29.82	4.96
	(850, 1590)	1	84732.38	CNS	402.75	100.1	16.36	16.14	3.63
		2	84731.35	764.95	409.2	99.03	17.26	16.74	3.73
		3	88749.14	CNS	405.76	103.64	16.01	17.69	3.88
		4	84815.14	28.02	406.33	102.33	17.8	16.81	3.74
		5	84755.59	206	405.02	99.75	17.47	16.71	3.72
60	(960, 1970)	1	109912.4	176.79	CNS	784.65	131.38	96.05	15.64
		2	112890.9	898.16	CNS	781.85	128.84	95.83	15.6
		3	113122.8	177.82	CNS	779.47	127.99	96.15	15.65
		4	113003.9	322.16	CNS	782.13	128.42	97.66	15.9
		5	113014.8	CNS	CNS	780.62	130.18	95.84	15.6
	(870, 1590)	1	122124.1	596.98	CNS	1433.25	253.34	145.22	23.64
		2	124339.7	CNS	CNS	1421.58	255.06	144.31	23.5
		3	125196.7	256.95	CNS	1430.52	246.72	142.37	23.18
		4	125655.1	1037.23	CNS	1439.01	256.29	146.95	23.93
		5	119757.9	CNS	CNS	1425.38	254.05	147.87	24.08
80	(960, 1970)	1	153922.4	CNS	CNS	CNS	CNS	527.12	85.82
		2	151669.2	CNS	CNS	CNS	CNS	529.91	84.28
		3	153344.7	CNS	CNS	CNS	CNS	524.73	85.44
		4	157264.5	CNS	CNS	CNS	CNS	530.58	88.39
		5	152732.9	CNS	CNS	CNS	CNS	527.39	85.87
	(1310, 2570)	1	119716.1	770.55	CNS	CNS	CNS	250.32	41.77
		2	119763.4	CNS	CNS	CNS	CNS	251.88	41.01
		3	120956	351.14	CNS	CNS	CNS	252.11	41.05
		4	120552	1553.41	CNS	CNS	CNS	247.36	42.27
		5	120038.4	382.82	CNS	CNS	CNS	250.31	40.75

6.1.2 Computational results for MCLS with n = 3.

From the computational results presented in Table 3 for MCLS with n = 3, we observe that Old-DP-LSPC and New-DP-LSPC with and without parallel implementation could not solve LSPC reformulation of any MCLS instance within the time limit. Likewise, Gurobi with its default setting could not solve mixed integer programming formulation of 50% of the instances within 2000 seconds and solved the remaining 50% of the instances using its default parallel implementation in 570 seconds on average. The DP-MCLS algorithm solved all instances in 390 seconds (on average) without parallel implementation and 73 seconds (on average) with parallel implementation. For each set (C_1, C_2, C_3), Gurobi's solution times among five instances range from 13.7 seconds to more than 2000 seconds as opposed to stable solution times of DP-MCLS; refer to Figure 4.

Table 3: Computational Results for MCLS with n = 3 and 40 time periods

(C_1, C_2, C_3)	Instance	OPT	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC		DP-MCLS	
					No-PI	With-PI	No-PI	With-PI
(970, 1950,2810)	1	75340.11	CNS	CNS	CNS	CNS	173.81	35.21
	2	75504.95	CNS	CNS	CNS	CNS	177.58	34.92
	3	75460.09	1417.5	CNS	CNS	CNS	172.26	37.33
	4	76050.1	CNS	CNS	CNS	CNS	173.06	34.18
	5	76032.55	CNS	CNS	CNS	CNS	175.19	35.4
(790, 1650, 2410)	1	86187.6	CNS	CNS	CNS	CNS	336.54	51.29
	2	85017.72	13.7	CNS	CNS	CNS	331.94	49.68
	3	87299.93	1381.8	CNS	CNS	CNS	330.68	53.44
	4	87738.2	CNS	CNS	CNS	CNS	335.15	50.16
	5	87134.51	827.1	CNS	CNS	CNS	334.77	51.39
(670, 1280,1970)	1	94971.26	365.9	CNS	CNS	CNS	726.61	155.79
	2	92043.93	CNS	CNS	CNS	CNS	723.96	160.36
	3	92786.66	95.7	CNS	CNS	CNS	720.17	154.27
	4	89009.17	CNS	CNS	CNS	CNS	731.66	156.85
	5	91492.86	972.1	CNS	CNS	CNS	725.17	154.61
(860, 1650, 2590)	1	84201.9	CNS	CNS	CNS	CNS	324.86	47.62
	2	85163.32	14.1	CNS	CNS	CNS	325.73	47.38
	3	84273.62	532.0	CNS	CNS	CNS	328.64	49.95
	4	81205.9	89.0	CNS	CNS	CNS	325.19	51.37
	5	81779.04	CNS	CNS	CNS	CNS	331.28	50.92

6.2 Computational Experiments on LS-PC instances

We compare the time taken by New-DP-LSPC to solve LS-PC instances having $m = 2$ and $m = 3$ breakpoints with the solution times of Gurobi 9.1 and Old-DP-LSPC. In addition to performing our experiments on instances considered in Koca et al. (2014), we also generate new LS-PC instances with more number of time periods and conducted experiments on them as well.

6.2.1 Computational results for LS-PC instances from Koca et al. (2014).

For LS-PC with $m = 2$, Koca et al. (2014) consider 50 time periods in the planning horizon and three sets of breakpoints $(b_1; b_2)$, three sets of per unit production costs $({}^1; {}^2)$, and three sets of fixed costs $({}^1; {}^2)$; refer to the first three columns in Table 4 for details. For LS-PC with $m = 3$, they consider 20 time periods in the planning horizon, two sets of breakpoints $(b_1; b_2; b_3)$, two sets of fixed costs $({}^1; {}^2; {}^3)$, and seven sets of per unit production costs $({}^1; {}^2; {}^3)$; refer to the first three columns in Table 5 for details. In these instances, all input parameters, except demand, are time invariant. From Tables 4 and 5, we make the following observations:

(i) For LS-PC instances with $m = 2$ (Table 4), the average time taken by Old-DP-LSPC, New-DP-LSPC without parallel implementation, and New-DP-LSPC with parallel implementation are 116 seconds, 30 seconds, and 6.5 seconds, respectively. Similarly, for LS-PC instances with $m = 3$ (Table 5), the average time taken by Old-DP-LSPC, New-DP-LSPC without parallel implementation, and New-DP-LSPC with parallel implementation are 17 seconds, 6.7 seconds, and 3.1 seconds, respectively. In a nutshell, New-DP-LSPC with parallel implementation is on average 13 times faster than Old-DP-LSPC.

Table 4: Computational Results for LS-PC with m = 2 and 50 periods (Instances from Koca et al. (2014))

λ_1, λ_2	(α_1, α_2)	$(b_1; b_2)$	OPT	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC	
						No-PI	With-PI
(3000, 6000)	(0, 0)	(800, 1600)	87726.95	25.4	138.66	37.3	7.15
		(900, 1800)	76313.5	29.6	113.44	28.74	6.44
		(1000, 2000)	69943.85	CNS	100.31	23.01	5.01
	(1, 0.5)	(800, 1600)	99990.3	11.4	138.26	36.84	7.12
		(900, 1800)	89026	3.7	111.51	28.93	6.39
		(1000, 2000)	82695.05	864.8	92.66	23.58	5.83
	(1, 1)	(800, 1600)	110190.95	32.5	141.26	37.53	7.72
		(900, 1800)	98777.5	34.2	114.73	28.18	6.14
		(1000, 2000)	92407.85	CNS	94.08	23.7	5.31
(3000, 4000)	(0, 0)	(800, 1600)	60537.6	80.6	143.04	35.71	7.09
		(900, 1800)	53348.5	11.6	112.82	28.91	6.59
		(1000, 2000)	49035.55	12.7	93.73	23.81	5.18
	(1, 0.5)	(800, 1600)	71990.3	38.6	139.16	37.77	6.55
		(900, 1800)	64862.85	3.3	112.1	28.08	6.16
		(1000, 2000)	60695.05	5.5	92.33	23.77	5.48
	(1, 1)	(800, 1600)	83001.6	5.7	139.92	37.99	7.02
		(900, 1800)	75812.5	3.9	114.62	28.12	6.49
		(1000, 2000)	71499.55	11.7	93.45	22.98	6.07
(3000, 7500)	(0, 0)	(800, 1600)	87726.95	4.8	138.37	37.66	7.46
		(900, 1800)	76313.5	1.3	112.57	28.44	6.5
		(1000, 2000)	69943.85	387.7	93.43	23.37	5.99
	(1, 0.5)	(800, 1600)	110190.95	5.8	138.23	37.36	7.11
		(900, 1800)	98777.5	5.5	115.35	28.13	6.91
		(1000, 2000)	92407.85	67.3	92.68	23	5.1
	(1, 1)	(800, 1600)	110190.95	10.7	138.88	36.04	6.33
		(900, 1800)	98777.5	2.8	112.33	28.19	5.29
		(1000, 2000)	92407.85	1.4	93.06	23.88	6.18

(ii) For LS-PC with m = 2, it can be observed from Table 4 that Gurobi was unable to solve 2 out of 27 instances, and for solving the remaining 25 instances, Gurobi took 66.5 seconds on an average. Whereas, New-DP-LSPC without and with parallel implementation solved all instances in 30 seconds and 6.3 seconds (on average), respectively. For LS-PC with m = 3 (Table 5), the average solution times of Gurobi, New-DP-LSPC No-PI, and New-DP-LSPC With-PI are 118, 6.8, and 3.1 seconds, respectively. The standard deviations in the solution times of Gurobi, New-DP-LSPC without parallel implementation and New-DP-LSPC with parallel implementation are 248, 5.6 and 0.7 seconds, respectively, for LS-PC instances with m = 2, and 156, 1.8 and 0.5 seconds for LS-PC instances with m = 3.

(iii) For LS-PC with m = 2, out of the 27 instances, Gurobi outperforms New-DP-LSPC With-PI 10 times and New-DP-LSPC outperforms Gurobi 17 times. Similarly, for LS-PC with m = 3, out of the 28 instances, Gurobi outperforms New-DP-LSPC With-PI only 7 times and New-DP-LSPC outperforms Gurobi 21 times.

Table 5: Computational Results for LS-PC with m = 3 and 20 periods (Instances from Koca et al. (2014))

$(b_1; b_2; b_3)$	$_1;\ _2;\ _3$	$(\ ^1;\ ^2;\ ^3)$	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC	
					No-PI	With-PI
(500, 1000, 1500)	(3000, 6000, 9000)	(0, 0, 0)	41.9	19.66	8.91	3.27
		(1.3, 1.5, 1.8)	50.6	19.91	9.02	3.25
		(1.3, 1.8, 1.5)	115.1	19.3	8.97	3.79
		(1.5, 1.3, 1.8)	26.7	19.93	9.01	3.31
		(1.5, 1.8, 1.3)	32.4	19.75	8.87	3.29
		(1.8, 1.3, 1.5)	12.2	19.32	9.12	3.59
		(1.8, 1.5, 1.3)	727.6	19.37	8.99	3.07
	3000, 3500, 5000)	(0, 0, 0)	30.4	19.48	8.72	3.75
		(1.3, 1.5, 1.8)	136.3	19.49	8.26	3.64
		(1.3, 1.8, 1.5)	2.6	19.28	8.29	3.55
		(1.5, 1.3, 1.8)	7.9	20.16	8.24	3.7
		(1.5, 1.8, 1.3)	3.2	19.21	7.93	3.61
		(1.8, 1.3, 1.5)	292.0	19.19	8.27	3.88
		(1.8, 1.5, 1.3)	10.2	19.74	7.88	3.28
(600, 1200, 1800)	(3000, 6000, 9000)	(0, 0, 0)	1156.2	14.15	5.05	2.59
		(1.3, 1.5, 1.8)	43.4	14.08	5.08	2.74
		(1.3, 1.8, 1.5)	2.8	14.07	4.96	2.63
		(1.5, 1.3, 1.8)	16.9	14.02	4.96	2.86
		(1.5, 1.8, 1.3)	266.8	14.01	4.97	2.47
		(1.8, 1.3, 1.5)	122.0	14.06	5.09	2.56
		(1.8, 1.5, 1.3)	207.9	14.04	5.06	2.7
	3000, 3500, 5000)	(0, 0, 0)	3.4	14.12	5.01	2.69
		(1.3, 1.5, 1.8)	2.4	14.54	5.01	2.64
		(1.3, 1.8, 1.5)	1.5	13.99	5	2.55
		(1.5, 1.3, 1.8)	0.7	14.34	5.18	2.45
		(1.5, 1.8, 1.3)	0.9	13.8	5.22	2.58
		(1.8, 1.3, 1.5)	3.1	14.03	5.07	2.81
		(1.8, 1.5, 1.3)	1.8	14.59	5.1	2.84

6.2.2 Computational results for new LS-PC instances.

For LS-PC with two breakpoints (m = 2), we consider four sets of $(T; b_1; b_2)$ such that the number of periods in the planning horizon $T \ 2 \ f \ 50; 75g$ has two sets of breakpoints $(b_1; b_2)$ associated to it; see columns labeled as T and $(b_1; b_2)$ in Table 6. For each set of $(T; b_1; b_2)$, we solve ve instances where for each period $t \ 2 \ T$, demand d_t , xed costs $\ ^1_t$ and $\ ^2_t$ are random integers drawn from Uniform [400, 600],Uniform [2850, 3150] andUniform [5850, 6150], respectively. For each $t \ 2 \ T$, variable costs $\ ^1_t$ and $\ ^2_t$ are random numbers drawn fromUniform [0.5, 1.0], and holding cost h_t is 0.05.

For LS-PC with three breakpoints (m = 3), we consider 40 time periods in the planning horizon, and four di erent sets of breakpoints $(b_1; b_2; b_3)$: (970; 1950; 2810), (790; 1650; 2410), (670; 1280; 1970), and (860; 1650; 2590). Parameter $\ ^3_t$ for $t \ 2 \ T$ is a random integer drawn from Uniform [8850, 9150] and parameter $\ ^3_t$ is a random number drawn fromUniform [0.5, 1.0]. The remaining input parameters are generated in the same way as done in the case of LS-PC with m = 2. Similar to MCLS instances, the generation of new LS-PC instances is motivated from instances of Koca et al. (2014). We report the results of the experiments for LS-PC withm = 2

and LS-PC with $m = 3$ in Tables 6 and 7, respectively, and make the following observations.

Table 6: Computational Results for LS-PC with $m = 2$

T	(b_1, b_2)	Instance	OPT	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC	
						No-PI	With-PI
50	(800, 1600)	1	103801.9	42.3	168.14	39.71	7.21
		2	101954.3	126.9	158.38	37.68	7.83
		3	100903.2	91.5	159.83	37.65	7.1
		4	101574	11.2	162.56	39.02	8.08
		5	101762.9	57.5	161.5	37.13	7.44
	(1310, 2570)	1	76681.88	169.3	84.99	26.6	5.87
		2	76464.41	112.2	86.93	25.27	6.12
		3	76101.56	31.9	79.77	26.04	6.04
		4	77388.97	76.3	81.38	26.98	6.17
		5	77948.03	42.2	81.83	24.07	5.63
75	(1310, 2570)	1	101181.6	87.9	697.56	198.5	38.04
		2	103614.1	1627.2	690.51	196.11	40.88
		3	102503.5	CNS	701.19	201.7	38.61
		4	103992.1	466.8	711.15	199.35	39.4
		5	102477.8	563.5	707.63	200.82	38.29
	(1790, 3470)	1	94168.54	34.8	372.8	101.33	19.72
		2	93047.57	43.0	375.04	101.58	20.1
		3	94013.08	244.7	370.32	103.62	21.77
		4	93807.6	538.5	377.93	102.77	19.28
		5	93491.1	119.3	380.13	101.19	19.26

Table 7: Computational Results for LS-PC with $m = 3$ and 40 periods

(b_1, b_2, b_3)	Instance	OPT	Gurobi 9.1	Old-DP-LSPC	New-DP-LSPC	
					No-PI	With-PI
(970, 1950, 2810)	1	79067.31	27.29	376.22	144.54	28.33
	2	78821.4	CNS	378.58	143.1	29
	3	80363.27	CNS	376.04	141.09	27.16
	4	78050.1	712.63	377.5	145.48	29.46
	5	79032.55	620.53	378.35	145.03	29.81
(790, 1650, 2410)	1	86421.45	CNS	702.66	245.16	49.97
	2	86554.3	1021.9	701.16	248.55	47.39
	3	86366.94	77.47	699.02	244.94	48.46
	4	86587.29	1364.79	704.67	243.08	47.94
	5	86801.06	CNS	702.81	244.11	51.71
(670, 1280, 1970)	1	102733.3	314.63	1322.73	545.63	107.76
	2	103079.2	CNS	1317.66	541.87	105.04
	3	102767.9	104.43	1319.21	544.52	106.55
	4	102591.7	393.55	1331.59	542.59	108.18
	5	102740.9	1181.22	1328.25	540.56	105.78
(860, 1650, 2590)	1	87461.06	CNS	481.45	186.69	38.25
	2	87163.32	CNS	485.38	185.21	36.24
	3	87273.62	CNS	484.77	186.54	38.5
	4	87204	217.41	480.17	187.31	34.58
	5	87279.04	CNS	481.2	185	36.2

For solving LS-PC instances with $m = 2$, Gurobi was unable to solve one out of the 20 instances within the time limit of 2000 seconds. For the remaining 19 instances, Gurobi and Old-DP-LSPC took 236 and 330 seconds, respectively, on an average, whereas New-DP-LSPC took 91 seconds even without parallel implementation and 18 seconds with parallel implementation. For LS-PC with $m = 3$, Old-DP-LSPC took about 721 seconds on an average to solve all the instances

whereas, New-DP-LSPC without and with parallel implementation took around 279 seconds and 55 seconds, respectively. For LS-PC with $m = 3$, Gurobi was unable to solve 45% of the instances within the time limit of 2000 seconds and for the remaining 55% of the instances, average time taken by it is about 548 seconds. The standard deviations in the solution times of Gurobi and New-DP-LSPC with parallel implementation are 426 seconds and 0.95 seconds, respectively (see Figures 5 and 6).

7 Conclusion

We developed dynamic programming (DP) based exact algorithms to solve multi-module capacitated lot-sizing problem (MCLS) and MCLS with subcontracting (MCLS-S) where each time period has n modules (machines or vehicles) of different capacities, and cost functions are concave. These algorithms take $O(T^{2n+3})$ time, which is polynomial for fixed n 2. As a result, we addressed an open question on existence of a polynomial time algorithm for solving MCLS-(S) with fixed n 2. These DP algorithms generalize the results of Florian and Klein (1971) for MCLS with $n = 1$ and Atamtürk and Hochbaum (2001) for MCLS-S with $n = 1$. We also demonstrated how the algorithm for MCLS-S can be utilized to optimize a linear cost function over the n-mixing set (a generalization of well-known 1-mixing set), in polynomial time for a fixed n 2. Based on the results of our computational experiments for MCLS, it is evident that this DP algorithm is computationally efficient and stable, in comparison to Gurobi.

Additionally, we presented a new DP algorithm to solve a generalization of MCLS-S where the production cost functions are piecewise concave with m time invariant breakpoints (denoted by LS-PC-S). This approach explores a reduced solution space using a reduced number of state variables, and thereby took only 6.25% (on average) of the solution time taken by a DP algorithm of Koca et al. (2014). We also presented an exact algorithm to solve another generalization of MCLS, referred to as two-echelon multi-module capacitated lot-sizing problem (2E-MCLS), that takes $O(T^{4n+4})$ time for given n 1. This approach generalizes DP algorithms of van Hoesel et al. (2005) for 2E-MCLS with $n = 1$.

References

Aggarwal A, Park JK (1993) Improved Algorithms for Economic Lot Size Problems. Operations Research 41(3):549{571, URL http://dx.doi.org/10.1287/opre.41.3.549 .

Ahuja RK, Magnanti TL, Orlin JB (1988) Network ows. Sloan School of Management, Cambridge, Massachusetts.

Akbalik A, Hadj-Alouane AB, Sauer N, Ghribi H (2017) NP-hard and polynomial cases for the single-item lot sizing problem with batch ordering under capacity reservation contract. European Journal of Operational Research 257(2):483{493, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2016.07.028 .

Akbalik A, Rapine C (2013) The single item uncapacitated lot-sizing problem with time-dependent batch sizes: Np-hard and polynomial cases. European Journal of Operational Research 229(2):353{363, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2013.02.052 .

Archetti C, Bertazzi L, Grazia Speranza M (2014) Polynomial cases of the economic lot sizing problem with cost discounts. European Journal of Operational Research 237(2):519{527, URL http://dx.doi.org/10.1016/j.ejor.2014.02.044 .

Atamtürk A, Hochbaum DS (2001) Capacity Acquisition, Subcontracting, and Lot Sizing. Management Science 47(8):1081{1100, URL http://dx.doi.org/10.1287/mnsc.47.8.1081.10232 .

Bansal M (2019) Facets for single module and multi-module capacitated lot-sizing problems without backlogging. Discrete Applied Mathematics 255:117{141, URL http://dx.doi.org/10.1016/j.dam.2018.07.029.

Bansal M, Kianfar K (2014) n-step cycle inequalities: facets for continuous n-mixing set and strong cuts for multi-module capacitated lot-sizing problem. Lee J, Vygen J, eds., Integer Programming and Combinatorial Optimization, volume 8494 of Lecture Notes in Computer Science, 102{113.

Bansal M, Kianfar K (2015) n-step cycle inequalities: facets for continuous multi-mixing set and strong cuts for multi-module capacitated lot-sizing problem. Mathematical Programming 154(1):113{144, ISSN 1436-4646, URL http://dx.doi.org/10.1007/s10107-015-0906-1 .

Bansal M, Kianfar K (2017) Facets for continuous multi-mixing set with general coe cients and bounded integer variables. Discrete Optimization 26:1{25, ISSN 1572-5286, URL http://dx.doi.org/10.1016/j.disopt.2017.05.002 .

Bitran GR, Yanasse HH (1982) Computational Complexity of the Capacitated Lot Size Problem. Management Science 28(10):1174{1186, URL http://dx.doi.org/10.1287/mnsc.28.10.1174 .

Brahimi N, Absi N, Dauzère-Pérès S, Nordli A (2017) Single-item dynamic lot-sizing problems: An updated survey. European Journal of Operational Research 263(3):838{863, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2017.05.008 .

Brahimi N, Dauzere-Peres S, Najid NM, Nordli A (2006) Single item lot sizing problems. European Journal of Operational Research 168(1):1{16, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2004.01.054 .

Brimberg J, Hansen P, Lin KW, Mladenovic N, Breton M (2003) An oil pipeline design problem. Oper-

ations Research 51(2):228{239, ISSN 0030-364X, 1526-5463, URL http://dx.doi.org/10.1287/opre.51.2.228.12786 .

Chan LMA, Muriel A, Shen ZJ, Simchi-Levi D (2002) On the E ectiveness of Zero-Inventory-Ordering Policies for the Economic Lot-Sizing Model with a Class of Piecewise Linear Cost Structures. Operations Research 50(6):1058{1067, URL http://dx.doi.org/10.1287/opre.50.6.1058.350 .

Conforti M, Wolsey LA (2008) Compact formulations as a union of polyhedra. Mathematical Programming 114(2):277{289, ISSN 0025-5610, 1436-4646, URL http://dx.doi.org/10.1007/s10107-007-0101-0 .

Cormen TH, Stein C, Rivest RL, Leiserson CE (2009) Introduction to Algorithms (McGraw-Hill Higher Education), 3rd edition.

Croxton KL, Gendron B, Magnanti TL (2003) A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science 49(9):1268{1273.

Downey RG, Fellows MR (2012) Parameterized complexity (Springer Science & Business Media).

Eksioglu SD (2009) A primal{dual algorithm for the economic lot-sizing problem with multi-mode replenishment. European Journal of Operational Research 197(1):93{101, ISSN 03772217, URL http://dx.doi.org/10.1016/j.ejor.2008.04.043 .

Federgruen A, Lee CY (1990) The dynamic lot size model with quantity discount. Naval Research Logistics (NRL) 37(5):707{713, URL http://dx.doi.org/10.1002/1520-6750(199010)37:5<707::AID-NAV3220370509>3.0.CO;2-5

Florian M, Klein M (1971) Deterministic Production Planning with Concave Costs and Capacity Constraints. Management Science 18(1):12{20, URL http://dx.doi.org/10.1287/mnsc.18.1.12 .

Florian M, Lenstra JK, Rinnooy Kan AHG (1980) Deterministic Production Planning: Algorithms and Complexity. Management Science 26(7):669{679, URL http://dx.doi.org/10.1287/mnsc.26.7.669.

Flum J, Grohe M (2006) Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series (Springer-Verlag), ISBN 978-3-540-29952-3, URL http://dx.doi.org/10.1007/3-540-29953-X .

Gunluk O, Pochet Y (2001) Mixing mixed-integer inequalities. Mathematical Programming 90(3):429{457, ISSN 0025-5610.

Hwang HC, Ahn HS, Kaminsky P (2013) Basis paths and a polynomial algorithm for the multistage production-capacitated lot-sizing problem. Operations Research 61(2):469{482, ISSN 0030-364X, URL http://dx.doi.org/10.1287/opre.1120.1141 .

Hwang HC, Ahn HS, Kaminsky P (2016) Algorithms for the two-stage production-capacitated lot-sizing problem. Journal of Global Optimization 65(4):777{799, ISSN 0925-5001, URL http://dx.doi.org/10.1007/s10898-015-0392-2 .

Jarvis JJ, Rardin RL, Unger VE, Moore RW, Schimpeler CC (1978) Optimal design of regional wastewater systems: A xed-charge network ow model. Operations Research 26(4):538{550, ISSN 0030-364X.

Joch A (2013) Modular data centers: Weighing the pros and cons. The Business of Federal Technology URL https://fcw.com/articles/2013/04/08/prefab-data-center.aspx .

Kaminsky P, Simchi-Levi D (2003) Production and distribution lot sizing in a two stage supply chain. IIE Transactions 35(11):1065{1075, ISSN 0740-817X, 1545-8830, URL http://dx.doi.org/10.1080/07408170304401

Karimi B, Fatemi Ghomi S, Wilson J (2003) The capacitated lot sizing problem: a review of models and algorithms. Omega 31(5):365{378, ISSN 03050483, URL http://dx.doi.org/10.1016/S0305-0483(03)00059-8 .

Koca E, Yaman H, Akturk MS (2014) Lot Sizing with Piecewise Concave Production Costs. INFORMS Journal on Computing 26(4):767{779, URL http://dx.doi.org/10.1287/ijoc.2014.0597 .

Kulkarni K, Bansal M (2020) Discrete multi-module capacitated lot-sizing problems with multiple items. Technical Report .

Lenstra HW (1983) Integer programming with a xed number of variables. Mathematics of Operations Research 8(4):538{548, ISSN 0364765X, 15265471, URL http://www.jstor.org/stable/3689168 .

Li CL, Ou J, Hsu VN (2012) Dynamic lot sizing with all-units discount and resales. Naval Research Logistics (NRL) 59(3{4):230{243, URL http://dx.doi.org/10.1002/nav.21484 .

Miller AJ, Wolsey LA (2003) Tight formulations for some simple mixed integer programs and convex objective integer programs. Mathematical Programming 98(1-3):73{88, ISSN 0025-5610, 1436-4646.

Ou J (2012) Economic lot sizing with constant capacities and concave inventory costs. Naval Research Logistics (NRL) 59(7):497{501, URL http://dx.doi.org/10.1002/nav.21504 .

Ou J (2017) Improved exact algorithms to economic lot-sizing with piecewise linear production costs. European Journal of Operational Research 256(3):777{784, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2016.06.040 .

Ou J, Feng J (2019) Production lot-sizing with dynamic capacity adjustment. European Journal of Operational Research 272(1):261{269, ISSN 03772217, URL http://dx.doi.org/10.1016/j.ejor.2018.06.030 .

Pandzic H, Wang Y, Qiu T, Dvorkin Y, Kirschen DS (2015) Near-optimal method for siting and sizing of distributed storage in a transmission network. IEEE Transactions on Power Systems 30(5):2288{2300, ISSN 1558-0679, URL http://dx.doi.org/10.1109/TPWRS.2014.2364257 .

Pochet Y (1988) Valid inequalities and separation for capacitated economic lot sizing. OPERATIONS RESEARCH LETTERS 7(3):7.

Pochet Y, Wolsey LA (1993) Lot-Sizing with constant batches: Formulation and valid inequalities. Mathematics of Operations Research 18:767{785, ISSN 0364765X.

Pochet Y, Wolsey LA (2006) Production Planning by Mixed Integer Programming (Springer), ISBN 9780387299594.

Sanjeevi S, Kianfar K (2012) Mixed n-step MIR inequalities: Facets for the n-mixing set. Discrete Optimization 9(4):216{235, ISSN 1572-5286.

Shaw DX, Wagelmans APM (1998) An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. Management Science 44(6):831{838, ISSN 0025-1909, 1526-5501, URL http://dx.doi.org/10.1287/mnsc.44.6.831 .

Van Hoesel CPM, Wagelmans APM (1996) An O(T3) Algorithm for the Economic Lot-Sizing Problem with Constant Capacities. Management Science 42(1):142{150, URL http://dx.doi.org/10.1287/mnsc.42.1.142 .

Van Hoesel S, Kuik R, Salomon M, Van Wassenhove LN (1994) The single-item discrete lotsizing and scheduling problem: optimization by linear and dynamic programming. Discrete Applied Mathematics 48(3):289{303, ISSN 0166-218X, URL http://dx.doi.org/10.1016/0166-218X(92)00182-L .

van Hoesel S, Romeijn HE, Morales DR, Wagelmans APM (2005) Integrated lot sizing in serial supply chains with production capacities. Management Science 51(11):1706{1719, ISSN 0025-1909, URL http://dx.doi.org/10.1287/mnsc.1050.0378 .

Van Vyve M (2007) Algorithms for single-item lot-sizing problems with constant batch size. Mathematics of Operations Research 32(3):594{613, ISSN 0364-765X, 1526-5471, URL http://dx.doi.org/10.1287/moor.1070.0257 .

Veinott AF (1963) Unpublished Class Notes, Program in Operations Research, Stanford University, Stanford, California .

Voss S, Woodru DL (2006) Introduction to Computational Optimization Models for Production Planning in a Supply Chain (Berlin Heidelberg: Springer-Verlag), 2 edition.

Wagelmans A, Van Hoesel S, Kolen A (1992) Economic lot sizing: An o(n log n) algorithm that runs in linear time in the wagner-whitin case. Operations Research 40:S145{S156, ISSN 0030-364X.

Wagner HM, Whitin TM (1958) Dynamic Version of the Economic Lot Size Model. Management Science 5:89{96.

Xu B, Zhao J, Zheng T, Litvinov E, Kirschen DS (2017) Factoring the cycle aging cost of batteries participating in electricity markets. arXiv:1707.04567 [math] URL http://arxiv.org/abs/1707.04567, arXiv: 1707.04567.

Zangwill WI (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, ISSN 0025-1909, URL http://dx.doi.org/10.1287/mnsc.15.9.506 .

Zhang M, Kucukyavuz S, Yaman H (2012) A polyhedral study of multiechelon lot sizing with intermedi-

ate demands. Operations Research 60(4):918{935, ISSN 0030364X, URL http://dx.doi.org/10.1287/opre.1120.1058 .

Zhao M, Zhang M (2020) Multiechelon lot sizing: New complexities and inequalities. Operations Research opre.2019.1867, ISSN 0030-364X, 1526-5463, URL http://dx.doi.org/10.1287/opre.2019.1867 .

Acknowledgments

This research is funded by National Science Foundation Grant CMMI{ 1824897, which is gratefully acknowledged.

Appendices

A Proof for Theorem 1

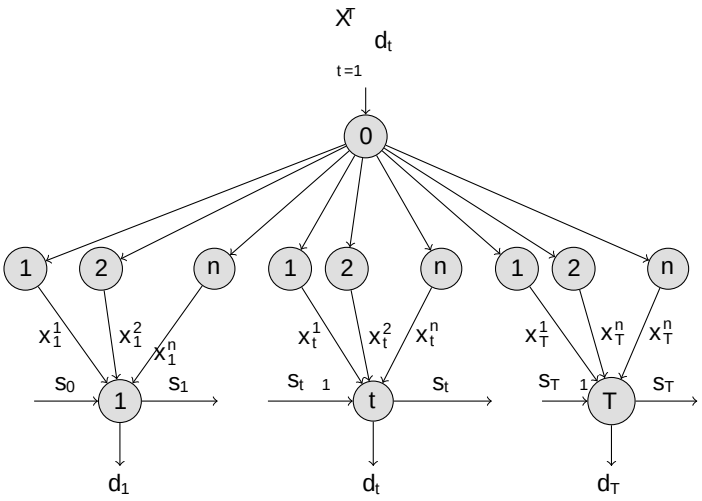

Figure 2: MCLS as a minimum cost network flow problem

Observe that MCLS can be represented as a fixed charge minimum cost network flow problem as shown in Figure 2. It is well known that when the cost functions are concave, the free arc network of the optimal solution is acyclic (Ahuja et al. 1988). A free arc is any arc in a network that carries a positive flow less that its full capacity, and a free arc network is an undirected network obtained after removing all the arcs that are not free. Using this result, we can easily prove by contradiction that there exists at most one fractional period within each regeneration interval of the optimal solution. □

B Proof of Theorem 2: Complexity of Algorithm DP-MCLS-(S)

Proof. We first analyze the time complexity of the DP algorithm for computing $_{kl}$ for an interval $[k;l]$. As we know, for calculating $_{kl}$, we need to perform three major steps: computing the minimum costs without fractional period (Section 3.1), finding all possible fractional production levels (Section 3.2), and computing the minimum costs with at most one fractional period (Section 3.3 / Section 3.4). The first step which is computing $G_k(t; ; 0)$ for all t and takes $O(T^{n+1})$ time because of the following reasons. Assuming that $G_k(k \quad 1; ; 0) = 0$ if $_i = 0$ for all $i = 1;::::n$ and $G_k(k \quad 1; ; 0) = 1$ otherwise, we successively compute $G_k(k; ; 0)$, $G_k(k+1; ; 0)$, $:::$, $G_k(l; ; 0)$, for all $2 Z_+^n$ where $_i \quad t \quad k+1 \quad T$. For each $(t;)$, $G_k(t; ; 0)$ can be computed in $O(2^n)$ time because the maximum number of subsets $S \quad f 1;::::ng$ is $O(2^n)$ and $G_k(t \quad 1; ; 0)$ has already been computed for all possible values of. There can be $O(T^{n+1})$ different possible permutations of $2 Z_+^n$ and t as $_i \quad t \quad k+1 \quad T$ for $i 2 N$ and $1 \quad k \quad t \quad l \quad T$. Hence the overall running time of this step of the algorithm is $O(T^{n+1} \quad 2^n)$ which is equal to $O(T^{n+1})$ for fixed n. The second step takes $O(T^n)$ time since there are $O(T^n)$ possible ^{u} vectors in and in the third step, for each fractional production f_u, computing $G_k^u(t; ; 1)$ for all t and also takes $O(T^{n+1})$ time (similar to the first step). Since there could be $O(T^n)$ possible fractional production levels, computing $G_k^u(t; ; 1)$ for all $(f_u; t;)$ takes $O(T^{2n+1})$ time. This implies that $_{kl}$ can be computed in $O(T^{2n+1})$ time. As there are $O(T^2)$ possible intervals and minimum cost of each interval can be computed in $O(T^{2n+1})$ time, the directed graph in phase (II) can be constructed in $O(T^{2n+3})$ time. Since this dominates the time required to solve the shortest path problem, i.e. $O(T^2)$, the overall complexity of DP-MCLS-(S) is $O(T^{2n+3})$. □

C Proof of Theorem 4: Complexity of New-DP-LSPCS

Proof. We first analyze the time complexity of the DP algorithm for computing $_{kl}$ for an interval $[k;l]$. As we know, for calculating $_{kl}$, we need to perform three major steps: computing the minimum costs without fractional period, finding all possible fractional production levels (Section 4.1), and computing the minimum costs with at most one fractional period (Section 4.2). The first step which is computing $G_k(t; ; 0)$ for all t and takes $O(T^{m+1})$ time because there can be $O(T^{m+1})$ different possible permutations of $2 Z_+^m$ and t as $_i \quad t \quad k+1 \quad T$ for $i 2 f 1;::::mg$ and $t \quad l \quad T$, and for each $(t;)$, $G_k(t; ; 0)$ can be computed in constant

time. The second step takes $O(T^m)$ time since there are $O(T^m)$ possible ^{r} vectors in and in the third step, for each fractional production f_r, computing $G^r_k(t; ; 1)$ for all t and also takes $O(T^{m+1})$ time. Since there could be $O(T^m)$ possible fractional production levels, computing $G^r_k(t; ; 1)$ for all $(f_r; t;)$ takes $O(T^{2m+1})$ time. This implies that $_{kl}$ can be computed in $O(T^{2m+1})$ time. As there are $O(T^2)$ possible intervals and minimum cost of each interval can be computed in $O(T^{2m+1})$ time, the directed graph can be constructed in $O(T^{2m+3})$ time. Since this dominates the time required to solve the shortest path problem, i.e. $O(T^2)$, the overall complexity of New-DP-LSPCS is $O(T^{2m+3})$. □

D Example of DP-MCLS

Example D.1. Consider an instance of MCLS with $n = 2$, three time periods, $C_1 = 6$, $C_2 = 9$ and demand $d = [4; 7; 6]$, setup costs $q^1 = [19; 23; 20]$ and $q^2 = [28; 35; 31]$, production costs $p = [1; 2; 1]$, and holding cost $h_t = 1$ for all $t \in \{1; 2; 3\}$.

We compute the optimal cost for a regeneration interval $[1; 3]$ i.e. $_{13}$, as follows.

Step 1: Minimum cost without fractional production. We first compute the minimum cost without the fractional period (see section 3.1). We calculate the values of $G_k(t; _1; _2; 0)$ for $k = 1$, $t = 1; \ldots; 3$, $_1 \in \{0; \ldots; {}^{\max}_1\}$, and $_2 \in \{0; \ldots; {}^{\max}_2\}$. Note that ${}^{\max}_i = \min\{T; \lfloor d_{kl}/C_i \rfloor\}$ for $i = 1; 2$, and $d_{kt} = \sum_{j=k}^{t} d_j$. Therefore, ${}^{\max}_1 = 2$ and ${}^{\max}_2 = 1$. Moreover, holding costs at the end of every period $t \in \{k; \ldots; l\}$ is equal to $h_t s_t$ where $h_t = 1$ and $s_t = {}_1C_1 + {}_2C_2 \quad d_{kt}$. Using the recursive equation in Section 3.1, we get

$$G_1(1; 0; 0; 0) = G_1(0; 1; 0; 0) = G_1(0; 0; 1; 0) = G_1(0; 1; 1; 0) = 1 \quad \text{(Infeasible cases)};$$

$$G_1(1; 1; 0; 0) = \min \begin{cases} G_1(0; 1; 0; 0) + h_1 s_1; \\ G_1(0; 0; 0; 0) + p_1 C_1 + q^1_1 + h_1 s_1 \end{cases} = \min \begin{cases} 1 \ ; \\ 0 + (1 \quad 6) + 19 + (1 \quad 2) \end{cases} = 27;$$

$$G_1(1; 0; 1; 0) = \min \begin{cases} G_1(0; 0; 1; 0) + h_1 s_1; \\ G_1(0; 0; 0; 0) + p_1 C_2 + q^2_1 + h_1 s_1 \end{cases} = \min \begin{cases} 1 \ ; \\ 0 + (1 \quad 9) + 28 + (1 \quad 5) \end{cases} = 42;$$

$$G_1(1;1;1;0) = \min \begin{cases} G_1(0;1;1;0) + h_1 s_1 = 1 \ ; \\ G_1(0;0;1;0) + p_1 C_1 + q_1^1 + h_1 s_1 = 1 \ ; \\ G_1(0;1;0;0) + p_1 C_2 + q_1^2 + h_1 s_1 = 1 \ ; \\ G_1(0;0;0;0) + p_1(C_1 + C_2) + q_1^1 + q_1^2 + h_1 s_1 = 73 \end{cases} = 73;$$

$$G_1(2;0;0;0) = G_1(2;1;0;0) = G_1(2;0;1;0) = G_1(2;3;0;0) = 1 \quad \text{(Infeasible cases)};$$

$$G_1(2;1;1;0) = \min \begin{cases} G_1(1;1;1;0) + h_2 s_2 = 77; \\ G_1(1;0;1;0) + p_2 C_1 + q_2^1 + h_2 s_2 = 42 + 35 + 4 = 81 \ ; \\ G_1(1;1;0;0) + p_2 C_2 + q_2^2 + h_2 s_2 = 27 + 53 + 4 = 84 \ ; \\ G_1(1;0;0;0) + p_2 C_1 + q_2^1 + p_2 C_2 + q_2^2 + h_2 s_2 = 1 \end{cases} = 77.$$

Similarly, $G_1(2;2;0;0) = 66$, and $G_1(3;1;1;0) = G_1(3;2;0;0) = G_1(3;0;1;0) = 1$. Below we discuss the computation of $G_1(2;1;1;0)$ and the calculations for other function values followed the similar process. According to the recursive equation in Section 3.1, we have four choices in period t = 2:

(a) To produce nothing: In this case the value of the function $G_1(2;1;1;0)$ is equal to $G_1(1;1;1;0) = 73$ plus the holding costs ($h_2 s_2 = 4$).

(b) To produce on machine of capacity C_1 at full capacity: In this case the value of the function $G_1(2;1;1;0)$ is equal to the sum of $G_1(1;0;1;0) = 42$, cost of producing on machine 1 at full capacity, i.e., $p_2 C_1 + q_2^1 = 35$, and the holding costs ($h_2 s_2 = 4$).

(c) To produce on machine of capacity C_2 at full capacity: In this case the value of the function $G_1(2;1;1;0)$ is equal to the sum of $G_1(1;1;0;0) = 25$, cost of producing on machine 2 at full capacity ($p_2 C_2 + q_2^2 = 53$), and the holding costs (= $h_2 s_2 = 4$), which is equal to 84.

(d) To produce on both machines at full capacity: In this case the value of the function $G_1(2;1;1;0)$ is equal to the sum of $G_1(1;0;0;0) = 1$, cost of producing on machines 1 and 2 at full capacity, i.e., $p_2(C_1 + C_2) + q_2^1 + q_2^2 = 88$, and the holding costs ($h_2 s_2 = 4$), which is equal to in nity.

All possible fractional production levels: Using steps in Section 3.2, we compute all possible fractional production levels in the regeneration interval [1; 3]. Note that $_1^{max} = 2$ and $_2^{max} = 1$. Set := $f(1;1);(2;0);(0;1)g$ is a set of pairs of $(_1; _2)$ that could lead to a fractional production level. For each $(_1^v; _2^v)$ 2 , we compute the values of fractional production levels

$f^v = d_{kl} \quad {}^v_1C_1 \quad {}^v_2C_2$, i.e.,

$$f^1 = \sum_{t=1}^{3} d_t \quad C_1 \quad C_2 = 2; \quad f^2 = \sum_{t=1}^{3} d_t \quad 2 \quad C_1 \quad 0 \quad C_2 = 5; \quad f^3 = \sum_{t=1}^{3} d_t \quad 0 \quad C_1 \quad C_2 = 8:$$

Therefore, the set of fractional production levels $F = f2; 5; 8g$.

Minimum Cost with fractional production: We now demonstrate how to compute the values of minimum costs with at most one fractional period in the regeneration interval. We compute the values of $G^v_k(t; \ {}_1: \ {}_2; 1)$ for where $k = 1, \ t = 1; : : : ; 3, \ {}_i \ 2 \ f 0; : : : ; \ {}^{max}_i g$ for $i = 1; 2$, and $v = 1$ (i.e. for $f^1 = 2$). Holding costs incurred at the end of every period $t \ 2 \ f k; : : : ; Ig$ is equal to $h_t s_t$ where $h_t = 1$ and $s_t = \ {}_1C_1 + \ {}_2C_2 + f^v \quad d_{kt}$. According to recursive equation provided in section 3.3, we get

$G^1_1(1; 0; 0; 0) = G_1(1; 0; 0; 1) = G^1_1(1; 1; 1; 1) = G_1(2; 0; 0; 1) = G^1_1(2; 1; 0; 1) = 1$ (Infeasible cases);

$$G^1_1(1;1;0;1) = \min \begin{cases} G^1_1(0;1;0;1) + h_1 s_1 = 1 \ ; \\ G^1_1(0;0;0;1) + p_1 C_1 + q^1_1 + h_1 s_1 = 1 \ ; \\ G_1(0;1;0;0) + p_1 f^1 + q^1_1 + h_1 s_1 = 1 \ ; \\ G_1(0;0;0;0) + p_1(C_1 + f^1) + q^1_1 + q^2_1 + h_1 s_1 = 59 \end{cases} = 59;$$

$$G^1_1(1;0;1;1) = \min \begin{cases} G^1_1(0;0;1;1) + h_1 s_1 = 1 \ ; \\ G^1_1(0;0;0;1) + p_1 C_2 + q^2_1 + h_1 s_1 = 1 \ ; \\ G_1(0;0;1;0) + p_1 f^1 + q^1_1 + h_1 s_1 = 1 \ ; \\ G_1(0;0;0;0) + p_1(C_2 + f^1) + q^1_1 + q^2_1 + h_1 s_1 = 65 \end{cases} = 65;$$

$$G^1_1(2;0;1;1) = \min \begin{cases} G^1_1(1;0;1;1) + h_2 s_2 = 65; \\ G^1_1(1;0;0;1) + p_2 C_2 + q^2_2 + h_2 s_2 = 1 \ ; \\ G_1(1;0;1;0) + p_2 f^1 + q^1_2 + h_2 s_2 = 69; \\ G_1(1;0;0;0) + p_2(C_2 + f^1) + q^1_2 + q^2_2 + h_2 s_2 = 1 \end{cases} = 65;$$

$$G_1^1(2;1;1;1) = \min \begin{cases} G_1^1(1;1;1;1) + h_2 s_2 = 1 ; \\ G_1^1(1;0;1;1) + p_2 C_1 + q_{12}^1 + h_2 s_2 = 65 + 35 + 6 = 106 ; \\ G_1^1(1;1;0;1) + p_2 C_2 + q_{12}^2 + h_2 s_2 = 59 + 53 + 6 = 118 ; \\ G_1^1(1;0;0;1) + p_2(C_1 + C_2) + q_{12}^1 + q_{12}^2 + h_2 s_2 = 1 ; \\ G_1(1;1;1;0) + p_2 f^1 + q_{12}^1 + h_2 s_2 = 73 + 27 + 6 = 106 ; \\ G_1(1;1;0;0) + p_2(f^1 + C_2) + q_{12}^1 + q_{12}^2 + h_2 s_2 = 27 + 80 + 6 = 113 ; \\ G_1(1;0;1;0) + p_2(f^1 + C_1) + q_{12}^1 + q_{12}^2 + h_2 s_2 = 42 + 74 + 6 = 122 \end{cases} = 106;$$

$$G_1^1(3;1;1;1) = \min \begin{cases} G_1^1(2;1;1;1) + h_3 s_3 = 1 ; \\ G_1^1(2;0;1;1) + p_3 C_1 + q_{13}^1 + h_3 s_3 = 65 + 26 + 0 = 91 ; \\ G_1^1(2;1;0;1) + p_3 C_2 + q_{13}^2 + h_3 s_3 = 1 ; \\ G_1^1(2;0;0;1) + p_3(C_1 + C_2) + q_{13}^1 + q_{13}^2 + h_3 s_3 = 1 ; \\ G_1(2;1;1;0) + p_3 f^1 + q_{13}^1 + h_3 s_3 = 77 + 22 + 0 = 99 ; \\ G_1(2;1;0;0) + p_3(f^1 + C_2) + q_{13}^1 + q_{13}^2 + h_3 s_3 = 1 ; \\ G_1(2;0;1;0) + p_3(f^1 + C_1) + q_{13}^1 + q_{13}^2 + h_3 s_3 = 1 \end{cases} = 91.$$

In a similar fashion, we compute the minimum costs with at most one fractional period for fractional production levels $f^2 = 5$ and $f^3 = 8$. Upon calculations, we obtain the values of $G_1^2(3;2;0;1)$ and $G_1^3(3;0;1;1)$. Finally, the overall optimal cost $_{13}$ can be computed by taking the minimum among $G_1^1(3;1;1;1)$, $G_1^2(3;2;0;1)$, and $G_1^3(3;0;1;1)$.

We now explain computation of $G_1^1(2;1;1;1)$. In period t = 2, we have the following choices.

(a) To not produce at all: In this case the function value of $G_1^1(2;1;1;1)$ is equal to $G_1^1(1;1;1;1) = 1$ plus the holding costs ($h_2 s_2 = 6$).

(b) To produce on a subset of machines at full capacity: If we choose to produce on machine 1 of capacity C_1 at full capacity in time period t = 2, then $G_1^1(2;1;1;1) = G_1^1(1;0;1;1) + p_2 C_1 + q_{12}^1 + h_2 s_2 = 106$ where $p_2 C_1 + q_{12}^1$ is the cost of producing on machine 1 in period 2 at full capacity, and h_{s2} is the holding cost incurred in period 2. Likewise, if we produce only on machine 2 at full capacity, $G_1^1(2;1;1;1) = G_1^1(1;1;0;1) + p_2 C_2 + q_{12}^2 + h_2 s_2 = 118$. We can also produce on both machines at full capacity in period 2, in which case the value of $G_1^1(2;1;1;1) = G_1^1(1;0;0;1) + p_2(C_1 + C_2) + q_{12}^1 + q_{12}^2 + h_2 s_2 = 1$ because $G_1^1(1;0;0;1)$ is

an infeasible case.

(c) To produce the fractional production and/or one of the machines at full capacity: If we choose to produce only the fractional production in period 2, the value of $G_1^1(2;1;1;1)$ is equal to the sum of $G_1(1;1;1;0)$, the cost of producing f^1 units on machine 1 ($p_2 f^1 + q_{12}^1$), and the holding costs ($h_2 s_2 = 6$) which is equal to 106. We also have a choice of producing on machine 1 or machine 2 at full capacity along with the fractional production, in which case the value of $G_1^1(2;1;1;1)$ is equal to $G_1(1;1;0;0) + p_2(f^1 + C_2) + q_{12}^1 + q_{12}^2 + h_2 s_2$ and $G_1(1;0;1;0) + p_2(f^1 + C_1) + q_{12}^1 + q_{12}^2 + h_2 s_2$ for machine 2 and machine 1, respectively.

E Comparison with DP algorithm of Koca et al. (2014)

We first briefly present the algorithm developed by Koca et al. (2014) (denoted by Old-DP-LSPCS) and discuss about the steps that differentiates their algorithm from New-DP-LSPCS. The outline of Old-DP-LSPCS is same as ours, i.e., it is comprised of two phases: computing optimal costs $_{kl}$ for all possible (semi)-regeneration intervals $[k;l]$ where $1 \quad k \quad l \quad T$ and finding the overall optimal solution using any shortest path algorithm. However, to compute $_{kl}$ for an interval $[k;l]$, Old-DP-LSPCS consists of two main steps where it computes: (1) Minimum costs for satisfying demands in each period of the (semi)-regeneration interval with no fractional period, and (2) Minimum costs for satisfying demands in each period of the regeneration interval with a fractional period. Only the first step of Old-DP-LSPCS is similar to the first step of our algorithm discussed in Section 4.1.

The key difference between Old-DP-LSPCS and New-DP-LSPCS is in the ways to compute the minimum costs for satisfying the demand in a regeneration interval $[t;l]$ with at most one fractional period; in Section 4.3 we discussed how New-DP-LSPCS computes it. In this section we present the recursive equation from Old-DP-LSPCS to demonstrate the difference with New-DP-LSPCS. We keep the notations similar to our algorithm so that readers can easily compare both algorithms. Koca et al. (2014) introduced a recursive function $G_k(t; ; ; 1)$ to compute minimum cost of satisfying the demand from period k up to t during which production of exactly b_i units has occurred $_i$ times from period k through t, production of exactly b_i units occurs $_i$ times from period $t+1$ through l, and a fractional production of $_{kl}(;)$ takes place once during the interval $[k;t]$. The forward recursion used in Koca et al. (2014) to compute the minimum cost with one fractional period is as follows:

$$G_k(t;\ ;\ ;\ 1) = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} {}_i + 1 \quad (t \quad k+1);\ \text{or } \sum_{i=1}^{m} {}_i > l \quad t;\ \text{or } {}_{kl}(\ ;\ ) \in \{0; b_1; : : : ; b_m\}; \\ & \text{or } \sum_{i=1}^{m} {}_i b_i + {}_{kl}(\ ;\ ) \quad d_{kt} \text{ for } t < l;\ \text{or } \sum_{i=1}^{m} {}_i b_i + {}_{kl}(\ ;\ ) \in d_{kl} \text{ for } t = l; \\ \min \begin{cases} G_k(t \quad 1;\ ;\ ;\ 1) + h_t \sum_{i=1}^{m} {}_i b_i + {}_{kl}(\ ;\ ) \quad d_{kt} \\ G_k(t \quad 1;\ ;\ 0) + p_t({}_{kl}(\ ;\ )) + h_t \sum_{i=1}^{m} {}_i b_i + {}_{kl}(\ ;\ ) \quad d_{kt} \\ \min_{i \in \{1;:::;m\}} G_k(t \quad 1; \quad e_i\ ; \ + e_i\ ; 1) + p_t(b_i) + \\ \qquad h_t \sum_{i=1}^{m} {}_i b_i + {}_{kl}(\ ;\ ) \quad d_{kt} \end{cases} & \text{; otherwise.} \end{cases}$$

where $_{kl}(\ ;\ ) = d_{kl} \sum_{i=1}^{m} {}_i b_i \sum_{i=1}^{m} {}_i b_i$ is a function that represents a fractional production level during time period t and is dependent on and , where is same as it is in New-DP-LSPCS and is an m-sized vector comprising of $_i$'s, where each $_i$ denotes the number of times production of exactly b_i units has occurred from period $t+1$ through l.

Notice that the function $G_k(t;\ ;\ ;\ 1)$ includes the computation of the fractional production levels using additional argument $\in Z^m$ which denotes the amount to be produced in future. In contrast, New-DP-LSPCS takes advantage of the observation that the fractional production levels are solely dependent on the values of breakpoints and the total demand in the regeneration interval and not on the future production. Upon comparing both the algorithms, one would notice that we do not keep track of possible future production amounts as it is not necessary in our algorithm, but is a key requirement in the algorithm by Koca et al. (2014) for LS-PC-S. More specically, recall that in Section 4.1, New-DP-LSPCS computes fractional production levels separately and then, for each fractional production level f^r, it computes the minimum cost of satisfying the demand from period k up to t during which production of exactly b_i units has occurred $_i$ times from period k through t, using the recursive function $G^r_k(t;\ ;\ 1)$ defined in Section 4.2. Another advantage of computing the fractional production levels beforehand is that it reduces the solution search space and allows parallelization of the algorithm. Because of the foregoing differences, our algorithm explores reduced solution space in differently, in comparison to the algorithm by Koca et al. (2014), and is computationally efficient.

We demonstrate using an example the difference between the number of value function computations carried out using the new DP algorithm for LS-PC-S, i.e. New-DP-LSPCS, and the

algorithm developed by Koca et al. (2014), i.e. Old-DP-LSPCS. For better perspective, we consider the same example as we did to showcase the computations for MCLS with $m = 2$. However, we now solve this problem as an LS-PC with $m = 3$.

Example E.1. Consider an instance of LS-PC with $m = 3$ and three time periods where the values of breakpoints are $b^1 = 6$, $b^2 = 9$, and $b^3 = 15$. Moreover, demand $d = [4; 7; 6]$, fixed costs $q^1 = [19; 23; 20]$, $q^2 = [28; 35; 31]$, and $q^3 = [47; 58; 51]$, production costs $p = [1; 2; 1]$, and holding costs $h_t = 1$ for all $t \in \{1; 2; 3\}$. We compute the optimal cost for a regeneration interval $[1; 3]$ i.e. $_{13}$, using New-DP-LSPCS and Old-DP-LSPCS.

Calculations using New-DP-LSPCS. For the sake of this example, we consider a regeneration interval $[1; 3]$ where the total demand to be satisfied is equal to 17. Note that $^{max}_1 = \min\{3; \lfloor 17/6 \rfloor\} = 2$, $^{max}_2 = \min\{3; \lfloor 17/9 \rfloor\} = 1$, and $^{max}_3 = \min\{3; \lfloor 17/15 \rfloor\} = 1$.

Step 1. Minimum Costs without a fractional production. We calculate the values of $G_1(t; \ _1; \ _2; \ _3; 0)$ for $t \in \{0; 1; \ldots; 3\}$, $_1 \in \{0; \ldots; \ ^{max}_1\}$, $_2 \in \{0; \ldots; \ ^{max}_2\}$, and $_3 \in \{0; \ldots; \ ^{max}_3\}$. Holding costs at the end of every period $t \in \{k; \ldots; I\}$ is equal to $h_t s_t$ where $h_t = 1$ and $s_t = \ _1 b_1 + \ _2 b_2 + \ _3 b_3 \ - \ d_{1t}$. Using the recursive equation provided in Koca et al. (2014) to compute minimum costs without a fractional period, we get the following values of the cost function:

- $G_1(0; 1; 0; 0; 0) = G_1(0; 0; 1; 0; 0) = G_1(0; 0; 0; 1; 0) = G_1(1; 0; 0; 0; 0) = \ 1 \ ;$
- $G_1(0; 0; 0; 0; 0) = 0; \ G_1(1; 1; 0; 0; 0) = 27; \ G_1(1; 0; 1; 0; 0) = 42;$
- $G_1(1; 0; 0; 1; 0) = 73; \ G_1(2; 0; 0; 1; 0) = 77; \ G_1(2; 2; 0; 0; 0) = 66;$
- $G_1(2; 0; 0; 0; 0) = G_1(2; 1; 0; 0; 0) = G_1(2; 0; 1; 0; 0) = \ 1 \ ;$
- $G_1(2; 1; 1; 0; 0) = \min \begin{cases} G_1(1; 0; 1; 0; 0) + p_2 b^1 + q^1_2 + h_2 s_2 = 81; \\ G_1(1; 1; 0; 0; 0) + p_2 b^2 + q^2_2 + h_2 s_2 = 84 \end{cases} = 81;$
- $G_1(3; 1; 0; 0; 0) = G_1(3; 0; 1; 0; 0) = G_1(3; 0; 0; 1; 0) = G_1(3; 2; 0; 0; 0) = \ 1 \ .$

Since $_i$ is bounded by $^{max}_i$ for $i \in \{1; \ldots; m\}$ and there are 4 time periods (including period 0), the total number of function values computed is equal to $4 \prod_{i=1}^{3} (^{max}_i + 1) = 48$.

Step 2. Computing all possible fractional production levels. We now compute the

possible fractional production levels in the regeneration interval [1;3] as discussed in Section 4.1:

$$f^1 = \sum_{t=1}^{3} d_t \quad b_1 \quad b_2 = 2; \quad f^2 = \sum_{t=1}^{3} d_t \quad 2b_1 = 5;$$
$$f^3 = \sum_{t=1}^{3} d_t \quad b_2 = 8; \quad f^4 = \sum_{t=1}^{3} d_t \quad b_3 = 2; \quad f^5 = \sum_{t=1}^{3} d_t \quad b_1 = 11:$$

Clearly, set $F = f2; 5; 8; 2; 11g$ and the computational e orts to derive this set are negligible. There are ve possible fractional production levels and each fractional quantity has an integer multiple of the breakpoints (i.e. ^{r}) associated with it. For example, for $f^1 = 2$, the vector of integer multiples ^{1} is equal to (1, 1, 0).

Step 3. Minimum Costs with a fractional period. We now demonstrate how to compute the values of minimum costs with at most one fractional period in the regeneration interval. We calculate the values of $G_1^r(t; \ _1; \ _2; \ _3; 0)$ for $t \ 2 \ f \ 0; 1; : : : ; 3g$, $_1 \ 2 \ f \ 0; : : : ; \ ^r_1g$, $_2 \ 2 \ f \ 0; : : : ; \ ^r_2g$, and $_3 \ 2 \ f \ 0; : : : ; \ ^r_3g$. Holding costs at the end of every period $t \ 2 \ f \ k; : : : ; lg$ is again equal to $h_t s_t$ where $h_t = 1$ and $s_t = \ _1b_1 + \ _2b_2 + \ _3b_3 \quad d_{1t}$. Using the recursive equation in Section 4.2, we get the following values of the cost function for $f^1 = 2$:

$$G_1^1(0; 1; 0; 0; 1) = G_1^1(0; 0; 1; 0; 1) = G_1^1(0; 0; 0; 1; 1) = G_1^1(1; 0; 0; 0; 1) = \ 1 \ ;$$
$$G_1^1(2; 0; 1; 0; 1) = \min \begin{cases} G_1^1(1; 0; 0; 0; 1) + \ p_2b^2 + \ q_2^2 + \ h_2s_2 = \ 1 \ ; \\ G_1(1; 0; 1; 0; 0) + \ p_2f^{\ 1} + \ q_2^1 + \ h_2s_2 = 69 \end{cases} = 69;$$
$$G_1^1(3; 1; 1; 0; 1) = \min \begin{cases} G_1^1(2; 1; 0; 0; 1) + \ p_3b^2 + \ q_3^2 + \ h_3s_3 = \ 1 \ ; \\ G_1^1(2; 0; 1; 0; 1) + \ p_3b^1 + \ q_3^1 + \ h_3s_3 = 103; \\ G_1(2; 1; 1; 0; 0) + \ p_3f^{\ 1} + \ q_2^1 + \ h_3s_3 = 103 \end{cases} = 103.$$

Likewise we compute the values of cost functions for the remaining fractional production levels. The number of value function computations required to calculate these values is equal to

$$\sum_{r=1}^{|F|} 4 \quad \prod_{i=1}^{3} (\ ^r_i + 1) = 52 :$$

Therefore, the total number of function values calculations for computing the minimum cost of a regeneration interval [1; 3] is 100,

Calculations using Old-DP-LSPCS We now solve the same problem using Old-DP-LSPCS developed by Koca et al. (2014). Our objective is to show the e ectiveness of computing the fractional productions levels beforehand. As discussed before, computing $_{kl}$ using Old-DP-LSPCS comprises of two steps and we show some of the calculations below:

Step 1: Minimum costs without fractional period. This step is very similar to Step 1 of New-DP-LSPCS and hence to avoid repetition, we skip the calculations for minimum costs without fractional period. Essentially we compute the values of $G_1(t; \alpha_1; \alpha_2; \alpha_3; 0)$ for $t = 0; 1; \ldots; 3$, $\alpha_1 \in \{0; \ldots; 3\}$, $\alpha_2 \in \{0; \ldots; 3\}$, and $\alpha_3 \in \{0; \ldots; 3\}$. It should be noted that for $i \in \{1; \ldots; m\}$, α_i is bounded by T, as opposed to being bounded by $\min(T; \alpha_i^{max})$ in case of New-DP-LSPCS. As the result, the total number of function value calculations performed during this step is equal to $4 \times 4 \times 4 \times 4 = 256$ (in comparison to 48 in Step 1 of New-DP-LSPCS).

Step 2: Minimum costs with fractional period. Using the recursive equation of Koca et al. (2014), we compute the values of $G_k(t; \alpha_1; \alpha_2; \alpha_3; \beta_1; \beta_2; 1)$, where $k = 1$, $\beta_i \in \{0; \ldots; T\}$ for $i \in \{1; \ldots; m\}$, $\alpha_i \in \{0; \ldots; 3\}$ for $i \in \{1; 2\}$, and $t \in \{0; \ldots; 3\}$:

- $G_1(0; 0; 0; 0; 0; 0; 1) = G_1(0; 1; 0; 0; 0; 0; 1) = G_1(0; 0; 1; 0; 0; 0; 1) = 1$;
- $G_1(1; 1; 1; 0; 0; 0; 1) = G_1(1; 1; 0; 0; 0; 1; 1) = G_1(1; 0; 0; 0; 1; 1; 1) = G_1(1; 0; 1; 0; 1; 0; 1) = 1$;
- $G_1(0; 0; 0; 0; 0; 0; 1) = 65$;
- $G_1(2; 1; 0; 0; 0; 0; 1) = \min \begin{cases} G_1(1; 0; 0; 0; 1; 0; 1) + p_2 b_2^1 + q_{i2}^1 + h_2 s_2 = 106; \\ G_1(1; 1; 0; 0; 0) + p_{213}(\ ;\ ) + q_{i2}^2 + h_2 s_2 = 113 \end{cases} = 106$;
- $G_1(2; 0; 1; 0; 1; 0; 1) = \min \begin{cases} G_1(1; 0; 0; 0; 1; 1; 1) + p_2 b_2^2 + q_{i2}^2 + h_2 s_2 = 1; \\ G_1(1; 0; 1; 0; 0) + p_{213}(\ ;\ ) + q_{i2}^1 + h_2 s_2 = 69 \end{cases} = 69$;
- $G_1(3; 1; 1; 0; 0; 0; 1) = \min \begin{cases} G_1(2; 1; 0; 0; 0; 1; 1) + p_3 b_3^2 + q_{i3}^2 + h_3 s_3 = 1; \\ G_1(2; 0; 1; 0; 1; 0; 1) + p_3 b_3^1 + q_{i3}^1 + h_3 s_3 = 95; \\ G_1(2; 1; 1; 0; 0) + p_{313}(\ ;\ ) + q_{i2}^1 + h_3 s_3 = 108 \end{cases} = 95$.

Note that we compute the values of cost functions for all possible values of $t; (\alpha_1; \alpha_2; \alpha_3; \beta_1; \beta_2)$ where each of them can have $T + 1 = 4$ possible values. As a result, the number of function value calculations for computing minimum costs with fractional period is equal $4^6 = 4096$, in comparison to 52 in Step 3 of New-DP-LSPCS.

Figures

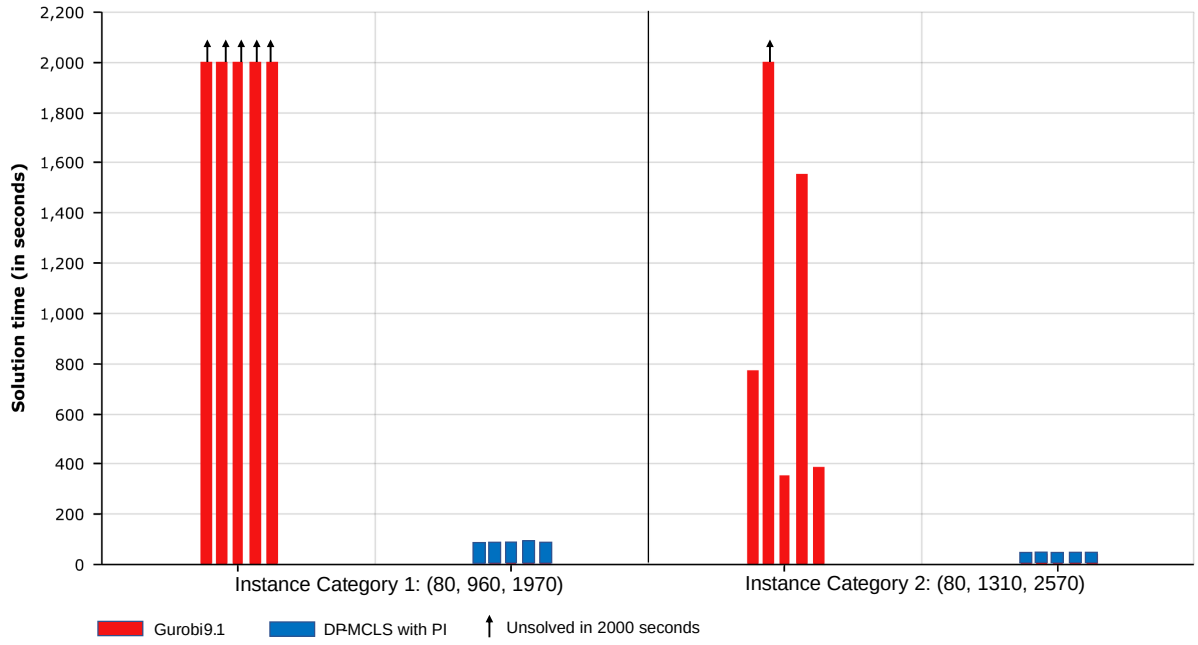

Figure 3: Gurobi versus DP-MCLS with PI for MCLS with n = 2: Solution Time Variations (from Table 2)

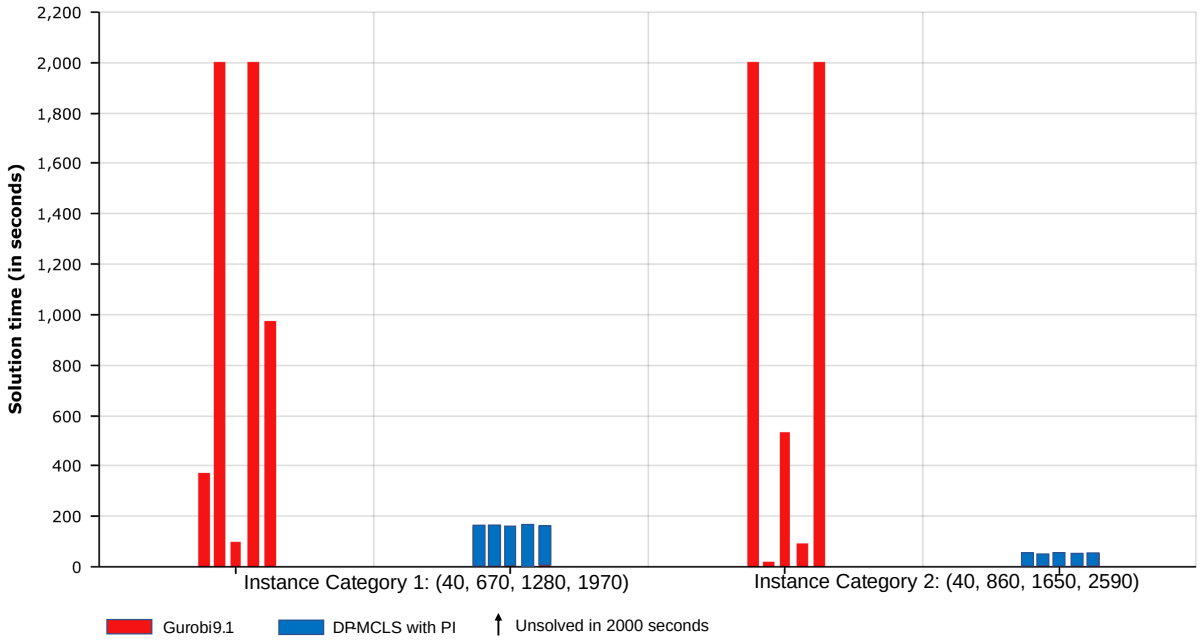

Figure 4: Gurobi versus DP-MCLS with PI for MCLS with n = 3: Solution Time Variations (from Table 3)

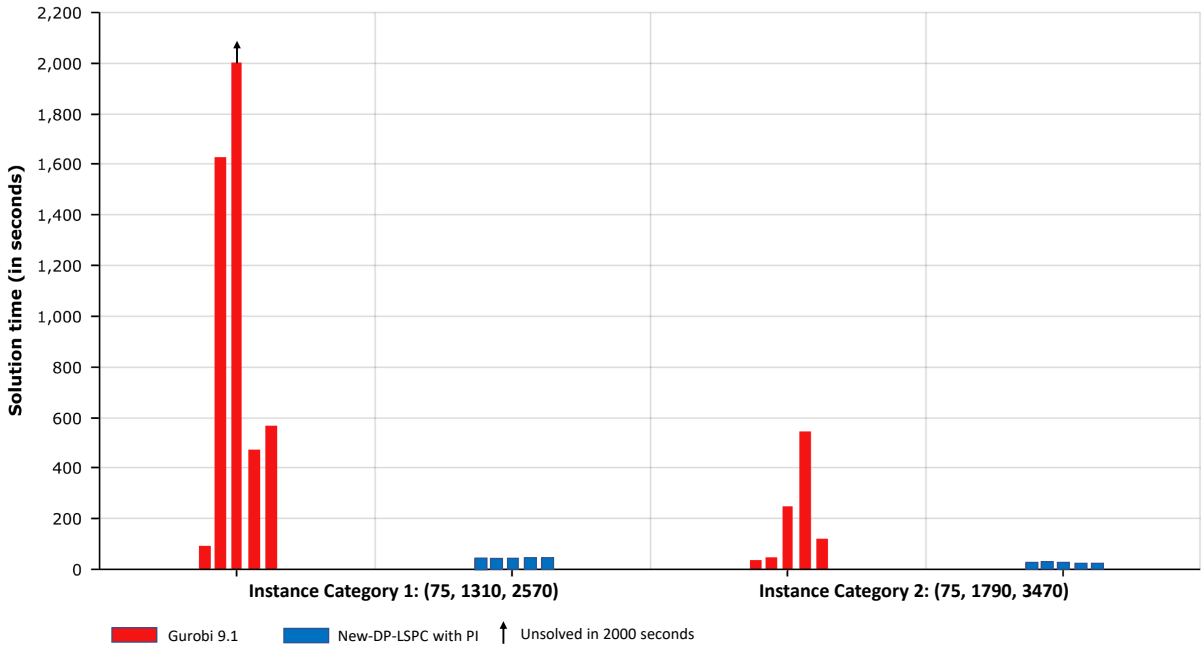

Figure 5: Gurobi versus New-DP-LSPC with PI for LS-PC with m = 2: Solution Time Variations (from Table 6)

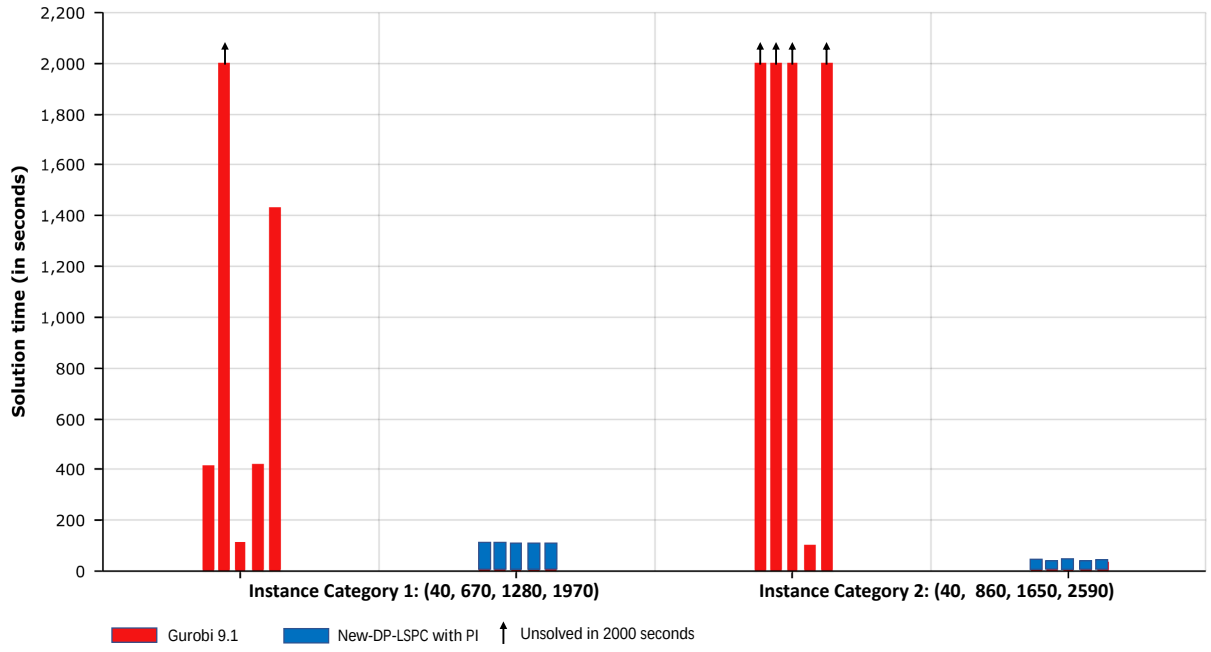

Figure 6: Gurobi versus New-DP-LSPC with PI for LS-PC with m = 3: Solution Time Variations (from Table 7)