Operations Research Letters 50 (2022) 168–175

Contents lists available at ScienceDirect

Operations Research Letters

www.elsevier.com/locate/orl

Discrete multi-module capacitated lot-sizing problems with multiple items

Kartik Kulkarni, Manish Bansal *

Department of Industrial and Systems Engineering, Virginia Tech, 250 Durham Hall, 1145 Perry Street, Blacksburg, VA 24060, United States of America

ARTICLE INFO

Article history:
Received 25 October 2020
Received in revised form 27 September 2021
Accepted 4 January 2022
Available online 11 January 2022

Keywords:
Multi-module capacitated lot-sizing with backlogging
Multi-item discrete lot-sizing problem
Fixed-parameter tractable algorithm
Concave production and holding costs
Dynamic programming

ABSTRACT

We study single-item discrete multi-module capacitated lot-sizing problems without and with backlogging where the amount produced in each time period is equal to the summation of binary multiples of the capacities of n available different modules (or machines). For fixed $n \geq 2$, we develop fixed-parameter tractable (polynomial) exact algorithms that generalize the algorithms of van Vyve (2007) for $n = 1$. We utilize these algorithms within a Lagrangian decomposition framework to solve their multi-item versions. Our algorithms are computationally efficient and stable, in comparison to Gurobi 9.0.

1. Introduction

Lot-sizing problem is a well-known combinatorial optimization problem that arises in production planning, retailing, and logistics to provide a production/procurement/shipping and inventory policy for a given planning horizon such that demand for each time period is satisfied at minimum total cost [6,19]. One of the various types of lot-sizing problems is the *discrete lot-sizing problem with a single module of constant capacity* (DLS-CC) that assumes "all-or-nothing" production in each time period, i.e., a module (or a machine) either produces at its full capacity or it produces nothing. In this paper, we study generalizations of the single- and multi-item DLS-CC without and with backlogging where the production in each time period of the planning horizon is the summation of binary multiples of the capacities of n available modules, and production and holding costs are concave. The modules represent machines, trucks, or suppliers of different capacities in the context of production, transportation, or procurement, respectively (see Section 1.1 for more details). We refer to these problems as m-Item Discrete Multi-module Capacitated Lot-Sizing Problem without and with Backlogging where m is the number of items, and we denote them by m-DLS-MC-WB and m-DLS-MC-B, respectively. Mathematically, the m-DLS-MC-B is defined as follows. Given a finite planning horizon $\mathcal{T} := \{1, \ldots, T\}$, a set of items $\mathcal{M} := \{1, \ldots, m\}$, demand d_t^i for period $t \in \mathcal{T}$ and item $i \in \mathcal{M}$, and a set of modules, $\mathcal{N} := \{1, \ldots, n\}$, of time invariant capacities $C^1, \ldots, C^n$ with setup cost $q_t^{i,j}$ for the module of capacity C^j, $j \in \mathcal{N}$ associated with item $i \in \mathcal{M}$, in period $t \in \mathcal{T}$, the m-DLS-MC-B is formulated as:

$$\text{Minimize} \sum_{i \in \mathcal{M}} \sum_{t \in \mathcal{T}} \left(p_t^i(x_t^i) + h_t^i(s_t^i) + b_t^i(r_t^i) + \sum_{j=1}^{n} q_t^{i,j} y_t^{i,j} \right) \tag{1a}$$

$$\text{s.t. } x_t^i + s_{t-1}^i - r_{t-1}^i = d_t^i + s_t^i - r_t^i, \ t \in \mathcal{T}, \ i \in \mathcal{M}, \tag{1b}$$

$$x_t^i = \sum_{j=1}^{n} C^j y_t^{i,j}, \ t \in \mathcal{T}, \ \ i \in \mathcal{M}, \tag{1c}$$

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

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

where x_t^i is the amount of production of item $i \in \mathcal{M}$ in period $t \in \mathcal{T}$, s_t^i is the inventory of item $i \in \mathcal{M}$ at the end of period $t \in \mathcal{T}$, r_t^i is the amount of backlog at the end of period t, $s_0^i = 0$ and $r_0^i = 0$ are the inventory and backlog, respectively, of item $i \in \mathcal{M}$ at the beginning of the planning horizon, and $y_t^{i,j}$ is a binary variable that determines whether the module $j \in \mathcal{N}$ of capacity C^j is utilized for item $i \in \mathcal{M}$ in period $t \in \mathcal{T}$ or not. Constraints (1b) are

* Corresponding author. Postal Address: 227 Durham Hall, 1145 Perry Street, Blacksburg, VA 24061, United States of America.
E-mail addresses: kartikrf@vt.edu (K. Kulkarni), bansal@vt.edu (M. Bansal).

https://doi.org/10.1016/j.orl.2022.01.002
0167-6377/

the classical inventory balance constraints. Constraints (1c) are the capacity constraints that ensure the amount of item $i \in \mathcal{M}$ produced in time period $t \in \mathcal{T}$ is equal to the sum of binary multiples of capacities $C^1, C^2, \ldots, C^n$, i.e., sum of a subset of $\{C^1, \ldots, C^n\}$, where $C^1 < C^2 < \ldots < C^n$ without loss of generality. Moreover, we assume that production costs $p_t^i(\cdot)$, holding costs $h_t^i(\cdot)$, and backlogging costs $b_t^i(\cdot)$ are concave functions. Constraints (1d) ensure that at most one item is produced on module $j \in \mathcal{N}$, in each time period. Such problems are also referred to as *small bucket problems*, i.e., each time period is so short that only one product can be produced on a module during a given period. The formulation for m-DLS-MC-WB is equivalent to formulation (1) with $r_t^i = 0$ for all $i \in \mathcal{M}$ and $t \in \mathcal{T}$. For convenience, often in this paper, we use m-DLS-MC-(W)B to denote m-DLS-MC-B and m-DLS-MC-WB together.

Below we provide a list of special cases of m-DLS-MC-B and m-DLS-MC-WB:

DLS-MC-WB: Single-item discrete lot-sizing problem with multiple capacitated modules and without backlogging, i.e., m-DLS-MC-WB with $m = 1$;

DLS-MC-B: Single-item discrete lot-sizing problem with multiple capacitated modules and backlogging;

DLS-CC-WB: Single-item discrete lot-sizing problem with a single constant capacitated module and without backlogging, i.e., m-DLS-MC-WB with $m = 1$ and $n = 1$ [17,24];

DLS-CC-B: Single-item discrete lot-sizing with a single constant capacitated module and backlogging [24];

***m*-DLS-CC-WB**: Multi-item discrete lot-sizing with a single constant capacitated module and without backlogging [17];

***m*-DLS-CC-B**: Multi-item discrete lot-sizing with single constant capacitated module and backlogging [17].

1.1. Applications of m-DLS-MC-WB and m-DLS-MC-B

In addition to production planning where modules essentially represent the machines for production, the m-DLS-MC-(W)B problems are also relevant in making tactical decisions in inventory management, supply chain, and logistics. The terms machines and setup costs used in a production planning setting can be interchanged with suppliers or trucks of different capacities and fixed ordering or fixed transportation costs, respectively. The order quantities from each supplier can be interpreted as the capacity of each supplier. Essentially, the problem objective would be to decide the quantities to be ordered from each supplier in each time period such that the overall demands are met and the total ordering and inventory costs are minimized. Furthermore, the following three key features of the m-DLS-MC-(W)B problems appear in a variety of applications:

(a) *Multiple capacitated modules in each time period.* Unlike DLS-CC-(W)B in which a single module with a time invariant capacity is taken into account, in the real-world applications, the total production (or transportation) capacity in each time period is obtained by summing up the capacity of a subset of n available modules, i.e., machines (or trucks), of different capacities.

(b) *All-or-Nothing Assumption.* This assumption is pertinent for applications where the setup cost of a module is significantly higher than the variable costs, i.e., production costs, and as a result, it is beneficial to utilize the module at full capacity. For example, in the tire manufacturing industry, the tire molds setup during a given time period are run at full capacity. Likewise, in the freight transportation and international shipping industries, trucks/containers carry goods at full capacity in order to maximize their utilization and thus, considering all-or-nothing assumption is reasonable.

(c) *Small Bucket Property.* Note that Constraints (1d) are redundant for DLS-MC-(W)B. However, for $m \geq 2$, the small bucket property ensures that in each time period, a truck (for example, an oil, water, or milk tanker) can carry at most one type of item.

1.2. Contributions of this paper

We develop dynamic programming based exact algorithms for DLS-MC-WB and DLS-MC-B. For $n = 1$, our algorithm for DLS-MC-B has the same worst-case complexity as the algorithm proposed by van Vyve [24], i.e., $O(T^2)$, and for $n \geq 2$, it is the first polynomial time algorithm for DLS-MC-B with a fixed n that takes $O(T^{n+1})$ time. This algorithm belongs to the class of fixed-parameter tractable algorithms (Downey and Fellows [7], Flum and Grohe [12]) because for a fixed n, it is a polynomial time algorithm for DLS-MC-B (an NP-hard problem if n is a part of the input). Interestingly, the DLS-MC-WB can be reformulated as a discrete lot-sizing problem with piecewise concave production cost functions (denoted by DLS-PC-WB). The breakpoints of these piecewise functions are determined by taking all the possible binary combinations of the capacities of the n available modules. In DLS-PC-WB, the 'discreteness' implies that the production in each time period is equal to either zero or one of the breakpoints. Note that the number of breakpoints, α, can vary from n (when $C^j = C$ for all $j \in \mathcal{N}$) to $2^n - 1$ depending on the value of capacities. Koca et al. [15] and Kulkarni and Bansal [16] developed algorithms for solving DLS-PC-WB without the assumption of all-or-nothing production. These algorithms can be slightly modified to solve DLS-PC-WB in $O(T^{\alpha+1})$ time which is equal to $O(T^{2^n})$ time in the worst case. Therefore, solving DLS-MC-WB with a fixed $n > 1$ using our algorithm is more efficient than reformulating and solving it as a DLS-PC-WB. For an example, when $n = 20$, the former takes $O(T^{21})$ and the latter takes $O(T^{1048576})$ time.

We also utilize the aforementioned algorithms for DLS-MC-WB and DLS-MC-B in solving m-DLS-MC-WB and m-DLS-MC-B, respectively, for $m \geq 2$ using a Lagrangian decomposition approach. More specifically, we first consider a Lagrangian relaxation of formulation (1) in which constraints (1d) are relaxed and introduced in the objective function with some nonnegative weights (Lagrangian multipliers) to enforce a penalty for violating these constraints. Thereafter, we solve this relaxation by decomposing it into m subproblems where each problem is a DLS-MC-B corresponding to each item, thereby providing a lower bound to the original problem. It is an iterative approach where Lagrangian multipliers are updated using a cutting-plane based approach [4].

We carry out computational experiments to evaluate the efficiency of our algorithms for solving DLS-MC-WB and DLS-MC-B. Our computational results show that these algorithms significantly outperform the state-of-the-art mathematical programming solver Gurobi 9.0 used for solving the mixed-binary programming formulation (1). In particular, out of the 240 total randomly generated instances of DLS-MC-WB and DLS-MC-B with $n = 2$, 3, and 4, Gurobi (with its default settings) was unable to solve 81 instances within a time limit of 2000 seconds whereas our algorithms were able to solve all these (unsolved) instances in less than 360 seconds, and in 88 seconds on average. For the remaining 159 instances that were solved by Gurobi within the time limit, the average solution times using Gurobi and our algorithms are 812 and 83 seconds, respectively.

Organization of this paper. In Section 2, we present a review of the literature on problems related to m-DLS-MC-WB and m-DLS-MC-B. In Section 3, we present exact dynamic programming (DP) algorithms that solve DLS-MC-WB and DLS-MC-B. In Section 4, we propose a Lagrangian decomposition approach to obtain a strong lower bound for m-DLS-MC-(W)B with $m \geq 2$. In Section 5, we report the computational results for m-DLS-MC-(W)B, $m = 1, 2, 3$, and discuss the efficiency of our proposed methods. Finally, in Section 6, we provide some concluding remarks.

2. Literature review

In this section, we discuss the literature on problems that are either closely related to or special cases of m-DLS-MC-WB and m-DLS-MC-B.

Discrete Capacitated Lot-Sizing Problems. Florian et al. [11] and Bitran and Yanasse [5] have independently shown that the single-item discrete capacitated lot-sizing problem with time varying capacities is NP-Hard. For DLS-CC-WB and DLS-CC-B with a time invariant capacity, i.e., DLS-MC-(W)B with $n = 1$, van Vyve [24] presented polynomial $O(T \log T)$ and $O(T^2)$ time algorithms, respectively. Miller and Wolsey [17] studied the m-DLS-CC-WB and m-DLS-CC-B and provided a full description of the convex hull of the feasible region for these problems by adding Gomory fractional cuts and mixed integer rounding inequalities, respectively. Bansal et al. [2] studied stochastic DLS-CC-(W)B and presented tight second stage formulations for these problems.

Several authors have also studied the variants of discrete lot-sizing problems where they considered start-up costs or sequence-dependent changeovers [9,21–23]. van Eijl [21] showed that m-DLS-CC-WB with start-up costs is NP-hard even when production costs are zero, and startup and holding costs for each item are time invariant. Note that in the context of logistics, startup costs are equivalent to the cost of placing orders which are negligible especially when orders are placed online.

Capacitated Lot-Sizing Problems with Non-discrete Capacity Constraints. For the completeness of the literature review, we also briefly review DLS-MC-WB without the all-or-nothing assumption (or with non-discrete capacity constraints), i.e., Problem (1) where constraints (1c) are replaced by $x_t^i \le \sum_{j=1}^n C^j y_t^{i,j}$ for all $t \in \mathcal{T}$ and $i \in \mathcal{M}$. Florian and Klein [10] presented a DP algorithm, which runs in $O(T^4)$ time, for this problem with $n = 1$. Lately, Kulkarni and Bansal [16] introduced a fixed parameter tractable algorithm for fixed $n \ge 2$ that takes $O(T^{2n+3})$ time. They also provided a new DP algorithm for DLS-PC-WB without the discreteness assumption, which is computationally 16 times (on average) faster than the algorithm by Koca et al. [15]. Researchers have also considered variants of this problem that allow any non-negative integer number of modules of a given capacity in each time period, i.e., $y_t^{1,j} \in \mathbb{Z}_+$ for all $t \in \mathcal{T}$ and $j \in \mathcal{N}$, and referred to these problems as multi-module capacitated lot-sizing (MMLS) problems. For MMLS with $n = 1$, Pochet and Wolsey [18] presented a DP algorithm that takes $O(T^3)$ time and introduced so-called (k, l, S, I) valid inequalities. For MMLS without and with backlogging, Sanjeevi and Kianfar [20] and Bansal and Kianfar [3] derived valid inequalities using n-mixing and continuous multi-mixing cut-generation procedure, respectively. Bansal [1] presented sufficient conditions under which the (k, l, S, I) inequalities and inequalities of [20] are facet-defining.

3. Exact algorithms for DLS-MC-WB and DLS-MC-B

In this section, we present dynamic programming algorithms to exactly solve DLS-MC-WB and DLS-MC-B with $n \ge 2$ capacitated modules/machines. We demonstrate that, for a fixed value of the parameter n, these algorithms run in polynomial time and thus belong to the class of fixed-parameter tractable algorithms. Since we are only considering the single-item DLS-MC-(W)B in this section, we omit the index i from the notations in formulation (1) to improve the readability of the section.

3.1. Dynamic programming algorithm for DLS-MC-WB with fixed $n \ge 2$

Let e_j be a unit vector of size n whose jth element is one and the remaining elements are zeros. We also define a vector $\tau = (\tau^1, \tau^2, \ldots, \tau^n) \in \mathbb{Z}_+^n$ where τ^j, $j \in \mathcal{N}$, denotes the number of times module j of capacity C^j has been set up. Furthermore, we define d_{1t} as the cumulative demand from period 1 up to period t, i.e., $d_{1t} = \sum_{j=1}^t d_j$ for $t \ge 1$. For $t \in \mathcal{T}$, let $\mathcal{H}(t, \tau)$ be a function that denotes the value of minimum cost solution for periods $1, 2, \ldots, t$ during which module $j \in \mathcal{N}$ runs τ^j times at full capacity. Clearly, the total amount produced up to period t is $\sum_{j=1}^n \tau^j C^j$.

We set $\mathcal{H}(t, \tau)$ to infinity if $\tau^j > t$ for any $j \in \mathcal{N}$ as there are only t periods from 1 to t. Since backlogging is not permitted, it is important to note that if $\sum_{j=1}^n \tau^j C^j < d_{1t}$ for $t \le T$, then the demand is not satisfied and in such case, we set the value of function $\mathcal{H}(t, \tau)$ to infinity. In each time period $t \in \mathcal{T}$, our objective is to choose among two possible decisions: (a) to not produce at all and (b) to set up some subset $S \subseteq \mathcal{N}$ of n available modules and produce at full capacity on each of the $|S|$ modules. If we choose option (a), i.e., if we choose to not produce at all during time period t, the value of the overall cost function is equal to the sum of $\mathcal{H}(t-1, \tau)$, and the inventory holding cost at the end of period t, i.e. $h_t\left(\sum_{j=1}^n \tau^j C^j - d_{1t}\right)$. On the other hand, if we choose to produce in period t using a subset S of the n available modules at full capacity, then the value of function $\mathcal{H}(t, \tau)$ is obtained by summing the function value at $(t-1, \tau - \sum_{j \in S} e_j)$, the cost of setting up the set S of modules and producing at their full capacity, and the holding costs at the end of period t i.e., $\mathcal{H}(t, \tau) = \mathcal{H}(t-1, \tau - \sum_{j \in S} e_j) + p_t\left(\sum_{j \in S} C^j\right) + \sum_{j \in S} q_t^j + h_t\left(\sum_{j=1}^n \tau^j C^j - d_{1t}\right)$, and then minimizing this summation over all possible subsets $S \subseteq \mathcal{N}$. We use these observations to derive the following forward recursion to compute $\mathcal{H}(t, \tau)$ for all possible values of τ and $t \in \mathcal{T}$:

$$\mathcal{H}(t,\tau) = \begin{cases} \infty, & \text{if } \tau^j > t \text{ for any } j \in \mathcal{N} \\ & \text{or } \sum_{j=1}^n \tau^j C^j < d_{1t}, \\ \min\limits_{S \subseteq \mathcal{N}} \left\{ \mathcal{H}(t-1, \tau - \sum\limits_{j \in S} e_j) + p_t\left(\sum\limits_{j \in S} C^j\right) \right. & \\ \left. + \sum\limits_{j \in S} q_t^j + h_t\left(\sum\limits_{j=1}^n \tau^j C^j - d_{1t}\right) \right\}, & \text{otherwise.} \end{cases} \tag{2}$$

3.2. Dynamic programming algorithm for DLS-MC-B

We now extend recursive equation (2) to obtain an exact algorithm to solve DLS-MC-B with a fixed $n \ge 2$. For reader's convenience, we keep the same notations τ, $\mathcal{H}(t, \tau)$, and e_j as the ones defined for DLS-MC-WB. For $z \in \mathbb{R}$, we denote $\max(0, z)$ by $(z)^+$. Note that in a time period $t \in \mathcal{T}$, it is now possible for $\sum_{j=1}^n \tau^j C^j$ to be less than d_{1t}. Hence, we do not set $\mathcal{H}(t, \tau)$ to infinity in such cases. However, at the end of every period $t \in \mathcal{T}$, either a backlogging cost of $b_t\left(d_{1t} - \sum_{j=1}^n \tau^j C^j\right)$ or a holding cost of $h_t\left(\sum_{j=1}^n \tau^j C^j - d_{1t}\right)$ is incurred. This leads to the summation of $h_t\left(\sum_{j=1}^n \tau^j C^j - d_{1t}\right)^+$ and $b_t\left(d_{1t} - \sum_{j=1}^n \tau^j C^j\right)^+$ as the total holding/backlogging costs during every time period. Again, similar to our approach for DLS-MC-WB, we have two choices in every time period: either to not produce at all, or to produce on a subset S of modules at full capacity. If we choose to produce nothing in time period t, the minimum costs of producing $\sum_{j=1}^n \tau^j C^j$ units from period 1 through t would be equal to the sum of $\mathcal{H}(t-1, \tau)$

and the holding/backlogging costs. However, if we choose to utilize a subset S of modules at full capacity, the function value would be equal to the sum of $\mathcal{H}(t-1, \tau - \sum_{j\in S} e_j)$, the cost of setting up the subset S of modules and producing on them at full capacity i.e. $p_t\big(\sum_{j\in S} C^j\big) + \sum_{j\in S} q_t^j$, and the holding/backlogging costs. Below, we present the recursive equation that computes the minimum costs over all possible τ and for all $t \in \mathcal{T}$ for DLS-MC-B:

$$\mathcal{H}(t,\tau) = \begin{cases} \infty, & \text{if } \tau^j > t \text{ for any } j \in \{1,\ldots,n\} \\ \min\limits_{S\subseteq\mathcal{N}} \Big\{ \mathcal{H}(t-1, \tau - \sum\limits_{j\in S} e_j) + p_t\Big(\sum\limits_{j\in S} C^j\Big) + \sum\limits_{j\in S} q_t^j \\ \quad + h_t\Big(\sum\limits_{j=1}^{n} \tau^j C^j - d_{1t}\Big)^+ + b_t\Big(d_{1t} - \sum\limits_{j=1}^{n} \tau^j C^j\Big)^+ \Big\}, & \text{otherwise.} \end{cases}$$

3.3. Optimal solution and complexity

The overall minimum costs (denoted by *OPT*) for both DLS-MC-WB and DLS-MC-B can be computed by using the expression below where we find the minimum value of the cost function $\mathcal{H}(t,\tau)$ at time period T and among all possible τ vectors.

$$OPT = \min_{\tau\in\{0,1,2,\ldots,T\}^n} \mathcal{H}(T,\tau).$$

We denote the above DP algorithms to solve DLS-MC-WB and DLS-MC-B by `DP-DLS-MC-WB` and `DP-DLS-MC-B`, respectively.

Theorem 1. *For a fixed number of modules $n \in \mathbb{Z}_+$ where $n \geq 1$, algorithms `DP-DLS-MC-WB` and `DP-DLS-MC-B` solve DLS-MC-WB and DLS-MC-B, respectively, in $O(T^{n+1})$ time, where T is the number of time periods in the planning horizon.*

Proof. Notice that in order to compute the optimal solution value, *OPT*, for a given DLS-MC-WB instance, we compute $\mathcal{H}(T,\tau)$ for all possible values of τ, using the recursive equation (2). Since $\mathcal{H}(T,\tau)$ is dependent on $\mathcal{H}(T-1,\tau)$ and $\mathcal{H}(T-1,\tau-\sum_{j\in S} e_j)$ for all $S \subseteq \{1,\ldots,n\}$, we compute the values of $\mathcal{H}(t,\tau)$ for all $t \in \mathcal{T}$ and $\tau \in \{0,\ldots,T\}^n$. For a given $t \in \mathcal{T}$ and $\tau \in \{0,\ldots,T\}^n$, $\mathcal{H}(t,\tau)$ can be computed in $O(2^n)$ time since there are 2^n possible subsets of $\{1,\ldots,n\}$ modules. Since we assume that n is a fixed and time invariant parameter, $\mathcal{H}(t,\tau)$ can be computed in constant time for a given t and τ. There are $O(T)$ time periods in the planning horizon and $O(T^n)$ possible τ vectors for each period $t \in \mathcal{T}$. As a result, in the worst case, the overall running time of algorithm `DP-DLS-MC-WB` is $O(2^n \times T^{n+1})$ which is equal to $O(T^{n+1})$ for a fixed n. Same follows for `DP-DLS-MC-B`. □

Remark 1. For DLS-MC-B with $n = 1$, van Vyve [24] provided an $O(T^2)$ time algorithm. Note that for DLS-MC-B with $n = 1$, 2, 3, the `DP-DLS-MC-B` takes $O(T^2)$, $O(T^3)$, and $O(T^4)$, respectively.

Remark 2. The algorithms developed by Koca et al. [15] and Kulkarni and Bansal [16] for lot-sizing problems with piecewise concave production costs can also be modified slightly and utilized to solve DLS-MC-WB. However, as mentioned in Section 1.2, depending on the value of the capacities of n available modules, the running time of these algorithms can be as much as $O(T^{2^n})$.

Remark 3. Observe that enumerating all the possible solutions of DLS-MC-WB takes at least $O(2^{nT})$ time, even for a fixed n. This is because in each time period t, there are $O(2^n)$ possible ways in which n modules can be set up, and as a result, an exhaustive enumeration of solutions over the entire planning horizon takes at least $O(2^n \times 2^n \times \ldots \times 2^n) = O(2^{nT})$ time. In contrast, the `DP-DLS-MC-WB` provides an optimal solution, if it exists, in $O(T^{n+1})$ time. The same is true for `DP-DLS-MC-B`.

4. Lagrangian decomposition for *m*-DLS-MC-(W)B with $m \geq 2$

We present a solution approach based on a Lagrangian relaxation scheme to solve the m-DLS-MC-(W)B with $m \geq 2$ items and n modules of time invariant capacities. We view the small bucket constraints (1d) as complicating constraints and upon relaxing these constraints, we obtain the following Lagrangian relaxation formulation:

$$P(\lambda) = \text{Min} \sum_{i\in\mathcal{M}} \sum_{t\in\mathcal{T}} \Big(p_t^i(x_t^i) + h_t^i(s_t^i) + b_t^i(r_t^i) + \sum_{j=1}^{n} q_t^{i,j} y_t^{i,j} \Big) + \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{N}} \lambda_{jt} \Big(\sum_{i=1}^{m} y_t^{i,j} - 1 \Big)$$

$$\text{s.t. (1b), (1c), and (1e) hold,}$$

where $\lambda_{jt} \geq 0$ is a Lagrange multiplier corresponding to module j and time period t. We denote a vector of all λ_{jt} for $j \in \mathcal{N}$ and $t \in \mathcal{T}$ by λ. Upon simplification, we get $P(\lambda) = \sum_{i\in\mathcal{M}} P^i(\lambda) - K_1$ where $K_1 = \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{N}} \lambda_{jt}$ and $P^i(\lambda)$ is a (single item) DLS-MC-(W)B problem corresponding to item $i \in \mathcal{M}$, i.e.,

$$P^i(\lambda) = \text{Minimize} \sum_{t\in\mathcal{T}} \Big(p_t^i(x_t^i) + h_t^i(s_t^i) + b_t^i(r_t^i) \Big) + \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{N}} \Big(q_t^{i,j} + \lambda_{jt} \Big) y_t^{i,j}$$

$$\text{s.t. } x_t^i + s_{t-1}^i - r_{t-1}^i = d_t^i + s_t^i - r_t^i,\ t \in \mathcal{T},$$

$$x_t^i = \sum_{j=1}^{n} C^j y_t^{i,j},\ t \in \mathcal{T},$$

$$y_t^i \in \{0,1\}^n, s_0^i = r_0^i = 0, x_t^i, s_t^i, r_t^i \geq 0,\ t \in \mathcal{T}.$$

Observe that for a given set of Lagrange multipliers λ, the Lagrangian relaxation can be decomposed into m DLS-MC-(W)B problems and we can compute $P^i(\lambda)$ using our algorithms presented in Section 3.

For a given λ, the Lagrangian relaxation provides an optimal solution for the original problem in case constraints (1d) are satisfied; otherwise, it provides a lower bound for the optimal objective value of m-DLS-MC-(W)B for $m \geq 2$. Our objective is to find the best lower bound over all the possible values of the Lagrange multipliers. This is done by solving the Lagrangian dual problem which is given by $z_{LD} = \max_{\lambda\geq 0} P(\lambda) = \max\{\phi : \phi \leq P(\lambda), \lambda \geq 0\}$. It is well known that $P(\lambda)$ is a concave non-smooth function (Section 7.5.3, pg. 717 of [4]). Therefore, we consider a cutting-plane based approach to solve the aforementioned maximization problem. We initialize this iterative approach by setting λ^1 equal to zero vector, $\phi^1 = -\infty$, and iteration counter $k = 1$. At each iteration k, we compute $\phi^k = P(\lambda^k)$ by solving m single-item sub-problems of type DLS-MC-(W)B in parallel and computing $P^i(\lambda^k)$ for all $i \in \mathcal{M}$ along with an optimal solution $\{\hat{y}_t^{i,k}, \hat{x}_t^{i,k}, \hat{s}_t^{i,k}, \hat{r}_t^{i,k}\}_{i,t}$ of the DLS-MC-(W)B associated with item $i \in \mathcal{M}$ and λ^k. Then, we derive an outer approximation of the concave function $P(\lambda)$ using a cutting-plane, i.e., $P(\lambda) \leq P(\lambda^k) + \Big(\sum_{i=1}^{m} \hat{y}_t^{i,j,k} - 1\Big)(\lambda - \lambda^k)$, where

$\left(\sum_{i=1}^{m} \hat{y}_t^{i,j,k} - 1\right)$ is a subgradient at $\lambda = \lambda^k$. This results in a linear program:

$$\text{Max}\left\{\phi : \phi \leq P(\lambda^l) + \left(\sum_{i=1}^{m} \hat{y}_t^{i,j,l} - 1\right)(\lambda - \lambda^l),\ l = 1, \ldots, k\right\}, \tag{3}$$

which we use to obtain a new Lagrange multiplier λ^{k+1} and a lower bound ϕ^{k+1}. In case $\frac{\phi^{k+1} - \phi^k}{\phi^{k+1}} < \epsilon$ (a pre-defined tolerance), we terminate this algorithm. For more details on the finite convergence of this method, we refer the reader to Section 7.5.3 of Bertsekas et al. [4].

Remark 4. The aforementioned Lagrangian decomposition approach can be used at each node of the branch-and-bound or branch-and-cut algorithms to obtain stronger lower bound for the node problems, thereby contributing to the performance of these approaches for solving m-DLS-MC-(W)B [8].

5. Computational results for m-DLS-MC-(W)B

In this section, we examine the computational efficiency and effectiveness of our exact and approximation algorithms for DLS-MC-(W)B and m-DLS-MC-(W)B for $m \geq 2$, respectively. We consider linear production, holding, and backlogging costs to compare the solution times of our algorithms with the solution times of Gurobi 9.0 with and without mixed integer rounding (MIR) cuts [25] and pairing inequalities [13] (a subset of mixing inequalities [14]). For each instance of DLS-MC-WB, we also compare the run time of our exact algorithm with the time taken to solve it as a DLS-PC-WB using algorithms proposed by [15,16]. We implemented the mixed binary formulation (1) using Gurobi 9.0 and our DP algorithms to solve the DLS-MC-(W)B using Python 2.7. For m-DLS-MC-WB with $m \geq 2$, our Lagrangian decomposition approach is also implemented in Python 2.7 that calls Gurobi to solve the linear program (3) and our implementation for DLS-MC-(W)B to solve each subproblem in parallel. We set a time limit of 2000 seconds for each experiment. It should also be noted that all the experiments were performed on a workstation with an Intel Xeon E5-1660 processor and 32 GB RAM.

5.1. Instance and cut generation for m-DLS-MC-(W)B with $m \geq 1$

For our computational experiments, we generated m-DLS-MC-WB and m-DLS-MC-B instances for $m = 1, \ldots, 4$, as follows. For each time period $t \in \mathcal{T}$ and for each item $i \in \{1, \ldots, m\}$, demand d_t^i is a random integer drawn from *Uniform*[400, 600]. Moreover, per unit production cost p_t^i is a random number drawn from *Uniform*[0.5, 1.0]. The per-unit holding cost and per-unit backlogging costs are assumed to be 0.05 and 0.15, respectively, for all items and all periods in the planning horizon. For DLS-MC-WB and DLS-MC-B, we perform experiments with two, three, and four capacities. For each set of experiments of DLS-MC-WB and DLS-MC-B with fixed n capacities, $n \in \{2, 3, 4\}$, we consider four sets of n capacities: $(C^1, \ldots, C^n)$. For each item $i \in \mathcal{M}$, setup costs of modules with capacities C^1, C^2, C^3, and C^4, i.e., $q_t^{i,1}$, $q_t^{i,2}$, $q_t^{i,3}$, and $q_t^{i,4}$, are random integers drawn from *Uniform*[2850, 3150], *Uniform*[5850, 6150], *Uniform*[8850, 9150], and *Uniform*[11850, 12150], respectively. All the generated instances are available at https://github.com/Bansal-ORGroup/Multi-Item-Discrete-MCLS. We also utilize MIR, pairing, and mixing cut-generation procedures to generate valid inequalities for m-DLS-MC-(W)B. These inequalities are added at the root node of the branch-and-cut search tree used by Gurobi for solving the problem instances. The average cut-generation time in our computational experiments is 0.02 seconds; please refer to the online appendix for more details.

5.2. Computational results for DLS-MC-WB and DLS-MC-B instances

We evaluate the computational efficiency of DLS-MC-WB and DLS-MC-B instances with two, three, and four capacities and report the results of these experiments in Tables 1 and 2, respectively. Columns labeled n and T denote the number of modules and number of periods in the planning horizon. We consider four sets of capacities $(C^1, \ldots, C^n)$ for each $n \in \{2, 3, 4\}$. Each row in these tables represents an instance category for a given n, T, and $(C^1, \ldots, C^n)$. For each instance category, we generate ten instances and solve them using Gurobi with default settings (labeled as GRB-DEF), Gurobi with aforementioned MIR and pairing cuts (labeled as GRB-CUTS), and `DP-DLS-MC-(W)B`, i.e., our DP algorithm for DLS-MC-(W)B. We also solve DLS-MC-WB instances using the algorithm of [15,16] for DLS-PC-WB (labeled as `DP-DLS-PC`). Columns labeled as Avg. and S.Dev. provide the average and standard deviation of the solution times (in seconds) for the instances that were solved to optimality within 2000 seconds. Moreover, columns labeled as #SolInst and #USI provide the number of solved and unsolved instances (out of 10), respectively, within the time limit, using the corresponding solution approach. In case there is no #USI column for an algorithm, it implies that #USI= 0, i.e., the algorithm solved all instances within the time limit. Based on the results of the 240 considered instances of DLS-MC-WB and DLS-MC-B with $n = 2$, $n = 3$, and $n = 4$ in Tables 1 and 2, we make the following observations.

(i) For DLS-MC-WB with $n = 2$, $n = 3$, and $n = 4$, Gurobi with default settings (GRB-DEF) was unable to solve 38%, 27%, and 25% of the instances, respectively. On the other hand `DP-DLS-MC-WB` was able to solve all the corresponding instances in less than 20, 75, and 342 seconds, respectively. For the remaining instances of DLS-MC-WB with $n = 2$, $n = 3$, and $n = 4$ that GRB-DEF was able to solve within the time limit, the average solution times of GRB-DEF are 725, 836, and 751 seconds, respectively, whereas the average time taken to solve the corresponding instances of DLS-MC-WB with $n = 2$, $n = 3$, and $n = 4$ using `DP-DLS-MC-WB` is 13.4, 49, and 186 seconds, respectively.

(ii) Similar to the results of DLS-MC-WB, we observe that for DLS-MC-B with $n = 2$, $n = 3$, and $n = 4$, GRB-DEF was unable to solve 40%, 38%, and 35% of the instances, respectively, whereas `DP-DLS-MC-B` solved all the corresponding instances in less than 30, 88, and 356 seconds, respectively. Among the instances that GRB-DEF was able to solve, the average solution times of GRB-DEF for DLS-MC-B instances with $n = 2$, $n = 3$, and $n = 4$ are 762, 840, and 936 seconds, respectively. On the other hand the average time taken by `DP-DLS-MC-B` to solve the same DLS-MC-B instances with $n = 2$, $n = 3$, and $n = 4$ are 18, 53, and 185.8 seconds, respectively.

(iii) Upon addition of the MIR cuts and the mixing/pairing cuts, inequalities (6)–(8) in the online appendix, to the problem and solving it using Gurobi (denoted by GRB-CUTS), we observed that GRB-CUTS was able to solve the problem instances faster than GRB-DEF. However, notice that our DP algorithms still outperform the solver. More specifically, for DLS-MC-WB instances with $n = 2$, $n = 3$, and $n = 4$, GRB-CUTS was unable to solve about 33%, 25%, and 23%, respectively, of the total instances. For the remaining instances that GRB-CUTS was able to solve within the time limit, the average time taken to solve DLS-MC-WB instances with $n = 2$, $n = 3$,

Table 1
Computational Results for DLS-MC-WB Instances.

n	T	$(C^1, \ldots, C^n)$	DLS-MC-WB													
			GRB-DEF					GRB-CUTS					DP-DLS-MC-WB		DP-DLS-PC		
			Solution Time (in s)			USI		Solution Time (in s)			USI		Solution Time (in s)		Solution Time (in s)		#USI
			Avg.	S.Dev.	#SolInst	#USI	Gap (%)	Avg.	S.Dev.	# SolInst	#USI	Gap (%)	Avg.	S.Dev.	Avg.	S.Dev.	
2	300	(670, 1280)	478.3	293.5	5	5	0.58	381.5	221.6	5	5	0.38	19.1	0.6	880.2	3.6	0
		(850, 1590)	826	282.4	7	3	0.73	696.8	214.7	7	3	0.51	15	0.7	789.1	1.9	0
		(960, 1970)	711.4	391.1	8	2	0.28	730.3	501.0	9	1	0.31	12.2	0.5	560.9	1.7	0
		(1310, 2570)	850.8	352.2	5	5	0.42	868.3	449.8	6	4	0.37	7.1	0.6	289.8	1.2	0
3	100	(670, 1050, 1420)	490.6	649.1	6	4	0.71	413.0	283.8	6	4	0.58	74.2	0.9	–	–	10
		(790, 1150, 1570)	1033.2	455.8	8	2	0.47	923.8	438.1	8	2	0.31	62.5	1.2	–	–	10
		(870, 1450, 1920)	950.0	460.2	7	3	0.86	853.3	431.6	7	3	0.57	41.0	0.9	–	–	10
		(970, 1690, 2620)	812.2	517.5	8	2	0.71	833.5	579.9	9	1	0.98	24.5	0.9	–	–	10
4	50	(470, 850, 1220, 1510)	987.4	389.8	7	3	0.98	839.0	299.2	7	3	0.70	340.5	1.5	–	–	10
		(670, 1050, 1420, 1790)	577.1	435.4	8	2	0.72	498.8	342.8	8	2	0.55	195.3	2.3	–	–	10
		(790, 1150, 1570, 1950)	724.4	666.6	6	4	0.50	780.8	610.3	7	3	0.43	146.6	2.7	–	–	10
		(870, 1450, 1920, 2290)	737.9	503.9	9	1	1.00	655.5	408.0	9	1	0.62	84.8	2.5	–	–	10

Table 2
Computational Results for DLS-MC-B Instances.

n	T	$(C^1, \ldots, C^n)$	DLS-MC-B											
			GRB-DEF					GRB-CUTS					DP-DLS-MC-B	
			Solution Time (in s)			USI		Solution Time (in s)			USI		Solution Time (in s)	
			Avg.	S.Dev.	#SolInst	#USI	Gap (%)	Avg.	S.Dev.	# SolInst	#USI	Gap (%)	Avg.	S.Dev.
2	300	(670, 1280)	544.5	567.6	6	4	0.57	659.4	598.1	7	3	0.50	28.1	0.6
		(850, 1590)	699.3	311.2	5	5	0.64	607.2	241.6	5	5	0.43	19.1	0.6
		(960, 1970)	850.9	497.7	8	2	0.21	916.0	498.1	10	0	–	14.7	0.6
		(1310, 2570)	944.9	527.9	5	5	0.86	829.4	383.7	5	5	0.64	9.9	0.6
3	100	(670, 1050, 1420)	684.5	419.1	6	4	0.81	588.0	496.2	6	4	0.57	86.3	6.4
		(790, 1150, 1570)	999.3	504.8	5	5	1.24	924.8	397.2	6	4	0.85	66.5	11.5
		(870, 1450, 1920)	910.4	688.4	7	3	0.57	915.1	457.1	9	1	0.37	41.7	8.9
		(970, 1690, 2620)	783.8	286.4	7	3	0.38	807.9	521.9	9	1	0.66	24.4	17.7
4	50	(470, 850, 1220, 1510)	1255.9	510.0	5	5	0.94	1170.7	416.8	5	5	0.47	353.8	2.1
		(670, 1050, 1420, 1790)	663.1	431.0	7	3	1.05	676.3	419.4	8	2	0.67	207.2	1.7
		(790, 1150, 1570, 1950)	1205.9	476.6	6	4	0.97	1080.6	427.6	6	4	0.66	154.8	1.5
		(870, 1450, 1920, 2290)	772.8	463.9	8	2	0.32	849.7	455.2	10	0	–	84.9	1.3

and $n = 4$ was 669, 755, and 694 seconds, respectively. Likewise, for DLS-MC-B with $n = 2$, $n = 3$, and $n = 4$, GRB-CUTS could not solve 33%, 25%, and 28% of the instances, and the average solution times of GRB-CUTS for the remaining solved instances of DLS-MC-B with $n = 2$, $n = 3$, and $n = 4$ are 753, 810, and 944 seconds, respectively. On an average, we observed DP-DLS-MC-WB and DP-DLS-MC-B to be 9 times and 10 times, respectively, faster than GRB-CUTS.

(iv) In addition to high solution times, we also observed that the variation of the solution times of Gurobi with or without cuts is significantly high (ranging from 39 seconds to more than 2000 seconds), whereas the solution times of our DP algorithms for DLS-MC-(W)B are highly stable and consistent among all ten instances. This characteristic makes our algorithms even more reliable for solving the DLS-MC-WB and DLS-MC-B instances.

(v) We also compare the time taken to solve DLS-MC-WB instances using DP-DLS-MC-WB with the solution times of DP-DLS-PC. For DLS-MC-WB with $n = 2$, we observe that DP-DLS-MC-WB was able to solve all instances within 20 seconds and in 13.4 seconds on average. In contrast, DP-DLS-PC solved all DLS-MC-WB instances within 890 seconds and 630 seconds on average. For DLS-MC-WB with $n = 3$ and $n = 4$, no instance was solved by DP-DLS-PC within the time limit of 2000 seconds whereas DP-DLS-MC-WB was able to solve all DLS-MC-WB instances within 78 seconds for $n = 3$, and 360 seconds for $n = 4$. This is because as discussed before, for $n = 2$, $n = 3$, and $n = 4$, the DP-DLS-MC-WB takes $O(T^3)$, $O(T^4)$, and $O(T^5)$ time, respectively, whereas DP-DLS-PC takes $O(T^4)$, $O(T^8)$, and $O(T^{16})$ time, respectively.

5.3. Computational results for m-DLS-MC-WB and m-DLS-MC-B

We evaluate the effectiveness of embedding our DP algorithms for DLS-MC-(W)B within a Lagrangian decomposition (LD) approach to obtain lower bounds for m-DLS-MC-(W)B instances with $m \geq 2$. For each instance, we also solve formulation (1) using Gurobi (labeled as GRB-DEF), and formulation (1) with additional MIR and pairing cuts using Gurobi (labeled as GRB-CUTS). For both GRB-DEF and GRB-CUTS, we record the best integer bound (BIB), i.e., the best upper bound, provided by Gurobi within 2000 seconds along with the lower bound provided by linear programming (LP) relaxation of m-DLS-MC-(W)B without and with cuts, respectively. We perform experiments for m-DLS-MC-(W)B with $n \in \{2, 3\}$ modules and $m \in \{2, 3, 4\}$ items, and report the results (average over 10 randomly generated instances) in Tables 3 and 4. Specifically, we report the following: (a) integrality gap at the root node, i.e., $100 \times (\text{BIB} - \text{LB}_{LP})/\text{BIB}$ where LB_{LP} denotes optimal LP relaxation solution value, in columns labeled as RootGap%, (b) number of instances (out of the 10 instances) not solved by GRB-DEF and GRB-CUTS to optimality within 2000 seconds, denoted by #USI, (c) remaining integrality gap that Gurobi reports at the end of 2000 seconds (labeled as Gap%) for the unsolved instances, (d) number of iterations (#Iter) performed by the LD approach, and (e) integrality gap with respect to the lower bound provided by the LD approach, LB_{LD} and the best integer bound obtained using GRB-DEF and GRB-CUTS, i.e., LD-Gap%:= $100 \times (\text{BIB} - \text{LB}_{LD})/\text{BIB}$.

Since the BIB provided by GRB-DEF and GRB-CUTS are different, we report the LD-Gap% with respect to each of the procedures and denote them by DEF-LD Gap% and CUTS-LD Gap%, respectively. To compare the lower bounds obtained using the LD approach with the lower bounds provided by LP relaxation of m-DLS-MC-(W)B using GRB-DEF and GRB-CUTS, we provide gap improvements in columns labeled as DEF-Gap Improv% and CUTS-Gap Improv%, respectively, which is equal to $100 \times (\text{BIB} - \text{LB}_{LD})/(\text{BIB} - \text{LB}_{LP})$.

Table 3
Computational Results for m-DLS-MC-WB instances.

n	T	m	$(C^1, \ldots, C^n)$	GRB-DEF				GRB-CUTS				Lagrangian Decomposition					
				RootGap%	Avg. BIB	USI		RootGap%	Avg. BIB	USI		Time (in s)	#Iter	DEF-LD Gap%	DEF-Gap Improv%	CUTS-LD Gap%	CUTS-Gap Improv%
						#USI	Gap%			#USI	Gap%						
2	100	2	(1270, 2120)	7.70	252809.9	9	1.33	6.96	252600.1	7	1.14	10.2	6	3.86	49.8	3.78	45.8
			(1310, 2570)	7.32	236759.9	8	0.65	6.53	236488.3	7	0.62	11.5	7	3.01	58.8	2.95	54.7
		3	(1270, 2120)	8.42	340510.9	6	1.82	7.91	340356.9	6	1.12	13.9	7	4.58	45.6	4.54	42.6
			(1310, 2570)	8.38	317418.0	8	1.61	7.82	317204.1	8	0.98	13.4	7	4.34	48.2	4.28	45.3
		4	(1270, 2120)	6.64	408180.0	9	1.77	6.20	407934.8	8	1.14	18.5	8	3.76	43.2	3.71	40.0
			(1310, 2570)	6.87	387664.4	10	1.62	6.42	387418.6	7	1.47	18.2	8	3.00	56.3	2.94	54.1
3	50	2	(970, 1690, 2620)	13.19	97452.7	7	1.06	10.22	95224.3	7	0.88	13.0	6	6.58	50.0	4.39	57.0
			(1310, 1750, 2120)	15.87	84165.8	8	1.28	14.54	82712.8	6	1.25	10.5	6	7.16	55.4	5.53	62.0
		3	(970, 1690, 2620)	10.39	152235.5	10	1.74	8.77	148845.9	10	0.72	11.2	6	6.09	40.7	3.95	54.9
			(1310, 1750, 2120)	10.28	137256.8	10	1.40	7.15	136707.4	8	1.18	15.4	10	4.49	56.8	4.11	42.5
		4	(970, 1690, 2620)	9.96	225616.9	10	1.56	7.36	224471.1	7	0.37	18.9	9	3.79	62.1	2.42	67.1
			(1310, 1750, 2120)	9.54	213418.7	10	1.44	6.99	208800.7	5	1.05	17.2	9	5.32	44.4	3.22	53.9

5.3.1. Computational results for m-DLS-MC-WB with $n = 2$ and $n = 3$

For $m \in \{2, 3, 4\}$ and $n = 2$, we consider 100 periods in the planning horizon and two sets of capacities (C^1, C^2): (1270, 2120), and (1310, 2570). Similarly, for $m \in \{2, 3, 4\}$ and $n = 3$, we consider $T = 50$ periods and two sets of capacities (C^1, C^2, C^3): (970, 1690, 2620), and (1310, 1750, 2120). For each set of capacities $(C^1, \ldots, C^n)$, we generate ten random instances using the procedure mentioned in Subsection 5.1, and report the results in Table 3 where each row is an average of results for ten instances. From Table 3, we observe that for m-DLS-MC-WB instances with $n = 2$ and $n = 3$, the initial LP gap with respect to the best integer bound obtained using GRB-DEF is 7.53% and 11.54% on average. In contrast, the LD approach was able to obtain lower bounds that reduced this gap to 3.77% for $n = 2$ and 5.5% for $n = 3$, which is 50% and 52%, respectively, improvement over the initial LP gap. Moreover, the remaining integrality gap reported by GRB-DEF at the end of 2000 seconds is 1.3% (on average) for $n = 2$ and 1.41% (on average) for $n = 3$. Similar comparisons can be made between GRB-CUTS and the LD approach. For m-DLS-MC-WB instances with $n = 2$ and $n = 3$, the initial LP gap with respect to the best integer bound obtained using GRB-CUTS is 6.9% and 9.2% on average. On the other hand, the LD approach was able to reduce these gaps to 3.7% for $n = 2$ and 3.9% for $n = 3$. Note that GRB-DEF and GRB-CUTS were unable to solve 88% and 72% of the total 120 instances whereas the LD approach took 15 seconds time (on average) to provide a strong lower bound. We observe similar results for m-DLS-MC-B with $n \in \{2, 3\}$; refer to the online appendix for more details.

6. Conclusion

In this paper, we introduced single- and multi-item discrete lot-sizing problems without and with backlogging where in each time period a subset of n available modules (machines or trucks) are used at full capacity. For single-item versions of the problems, we developed fixed parameter tractable algorithms that run in $O(T^{n+1})$ time for $n \geq 2$. This implies that for a fixed $n \in \mathbb{Z}_+$, the problems are solved in polynomial time. Our computational results showed that these algorithms are efficient and stable in comparison to using the state-of-the-art solver, Gurobi 9.0. To solve the multi-item versions, we embedded the foregoing algorithms within a Lagrangian decomposition framework where the Lagrange multipliers are updated iteratively using a cutting-plane based method. We observed that this decomposition method is able to provide stronger lower bounds within a few seconds.

Acknowledgements

This research is funded by the Automotive Research Center (ARC) of University of Michigan, Ann Arbor in accordance with Cooperative Agreement W56HZV-19-2-0001 U.S. Army CCDC Ground Vehicle Systems Center (GVSC) Warren, MI, and NSF CMMI - 1824897. We also would like to thank the area editor, associate editor, and reviewer for their constructive feedback.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.orl.2022.01.002.

References

[1] M. Bansal, Facets for single module and multi-module capacitated lot-sizing problems without backlogging, Discrete Appl. Math. 255 (2019) 117–141.
[2] M. Bansal, K.L. Huang, S. Mehrotra, Tight second-stage formulations for two-stage stochastic mixed integer programming problems, SIAM J. Optim. 28 (2018) 788–819.
[3] M. Bansal, K. Kianfar, n-step cycle inequalities: facets for continuous multi-mixing set and strong cuts for multi-module capacitated lot-sizing problem, Math. Program. 154 (1) (2015) 113–144.
[4] D.P. Bertsekas, W. Hager, O. Mangasarian, Nonlinear Programming, Athena Scientific, Belmont, MA, 1998.
[5] G.R. Bitran, H.H. Yanasse, Computational Complexity of the Capacitated Lot Size Problem, Vol. 28, Management Science, 1982, pp. 1174–1186.
[6] N. Brahimi, N. Absi, S. Dauzère-Pérès, A. Nordli, Single-item dynamic lot-sizing problems: an updated survey, Eur. J. Oper. Res. 263 (3) (2017) 838–863.
[7] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer Science & Business Media, 2012.
[8] M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems, Manag. Sci. 50 (12) (2004) 1861–1871.
[9] B. Fleischmann, The discrete lot-sizing and scheduling problem, Eur. J. Oper. Res. 44 (3) (1990) 337–348.
[10] M. Florian, M. Klein, Deterministic Production Planning with Concave Costs and Capacity Constraints, Vol. 18, Management Science, 1971, pp. 12–20.
[11] M. Florian, J.K. Lenstra, A.H.G. Rinnooy Kan, Deterministic Production Planning: Algorithms and Complexity, Vol. 26, Management Science, 1980, pp. 669–679.
[12] J. Flum, M. Grohe, Parameterized Complexity Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, ISBN 978-3-540-29952-3, 2006.
[13] Y. Guan, S. Ahmed, G.L. Nemhauser, Cutting planes for multistage stochastic integer programs, Oper. Res. 57 (2) (2008) 287–298.
[14] O. Günlük, Y. Pochet, Mixing mixed-integer inequalities, Math. Program. 90 (3) (2001) 429–457.
[15] E. Koca, H. Yaman, M.S. Aktürk, Lot sizing with piecewise concave production costs, INFORMS J. Comput. 26 (4) (2014) 767–779.
[16] K. Kulkarni, M. Bansal, Multi-module capacitated lot-sizing problem, and its generalizations with two-echelons and piecewise concave production costs, Technical Report, 2019, http://www.optimization-online.org/DB_HTML/2019/07/7294.html.
[17] A.J. Miller, L.A. Wolsey, Tight MIP formulation for multi-item discrete lot-sizing problems, Oper. Res. 51 (4) (2003) 557–565.
[18] Y. Pochet, L.A. Wolsey, Lot-sizing with constant batches: formulation and valid inequalities, Math. Oper. Res. 18 (1993) 767–785.
[19] Y. Pochet, L.A. Wolsey, Production Planning by Mixed Integer Programming, Springer, ISBN 9780387299594, 2006.
[20] S. Sanjeevi, K. Kianfar, Mixed n-step MIR inequalities: facets for the n-mixing set, Discrete Optim. 9 (4) (2012) 216–235.
[21] C. van Eijl, A polyhedral approach to the discrete lot-sizing and scheduling problem, PhD thesis, Technische Universiteit Eindhoven, 1996.
[22] C. van Eijl, C. van Hoesel, On the discrete lot-sizing and scheduling problem with Wagner-Whitin costs, Oper. Res. Lett. 20 (1) (1997) 7–13.
[23] S. Van Hoesel, R. Kuik, M. Salomon, L.N. Van Wassenhove, The single-item discrete lotsizing and scheduling problem: optimization by linear and dynamic programming, Discrete Appl. Math. 48 (3) (1994) 289–303.
[24] M. van Vyve, Algorithms for single-item lot-sizing problems with constant batch size, Math. Oper. Res. 32 (3) (2007) 594–613.
[25] L.A. Wolsey, Integer Programming, Wiley, New York, USA, 1998.

ONLINE APPENDIX

Appendix A: Cut-Generation Procedures

We apply the cut-generation procedures on so-called base inequalities that are derived as follows. We eliminate x_t^i, $r_{t\ \ 1}^i$, and s_t^i from (1b) using (1c) and (1e) to get

$$s_{t\ \ 1}^i + r_t^i + \sum_{j=1}^{n} C^j y_t^{i,j} \geq d_t^i, \quad t \in \mathcal{T}, i \in \mathcal{M}. \tag{4}$$

Since $C^1 < C^2 < \ldots < C^n$, we get the following valid base inequalities for m-DLS-MC-(W)B:

$$s_{t\ \ 1}^i + r_t^i + C^1 \left(\sum_{j=1}^{n} \lceil C^j / C^1 \rceil y_t^{i,j} \right) \geq d_t^i, \tag{5}$$

for $t \in \mathcal{T}$ and $i \in \mathcal{M}$, where $s_{t\ \ 1}^i + r_t^i \geq 0$ and $\sum_{j=1}^{n} \lceil C^j / C^1 \rceil y_t^{i,j} \in \mathbb{Z}_+$. For each $t \in \mathcal{T}$ and $i \in \mathcal{M}$, we apply a MIR procedure on the base inequality (5) to get MIR cuts for m-DLS-MC-(W)B, i.e.,

$$s_{t\ \ 1}^i + r_t^i + \beta_{it}^{(1)} \left(\sum_{j=1}^{n} \lceil C^j / C^1 \rceil y_t^{i,j} - \lfloor d_t^i / C^1 \rfloor \right) \geq \beta_{it}^{(1)}, \tag{6}$$

where $\beta_{it}^{(1)} = d_t^i - C^1 \lfloor d_t^i / C^1 \rfloor$. Now for each $i \in \mathcal{M}$ and pair $(k, k+1)$ where $k \in \{1, \ldots, T-1\}$, we apply the pairing or mixing procedures to get the following valid inequalities for m-DLS-MC-(W)B:

$$\eta_k^i \geq \beta_{i,k}^{(1)} \left(\left\lceil \frac{d_k^i}{C^1} \right\rceil - \sum_{j=1}^{n} \left\lceil \frac{C^j}{C^1} \right\rceil y_k^{i,j} \right) + \left(\beta_{i,k+1}^{(1)} - \beta_{i,k}^{(1)} \right) \left(\left\lceil \frac{d_{k+1}^i}{C^1} \right\rceil - \sum_{j=1}^{n} \left\lceil \frac{C^j}{C^1} \right\rceil y_{k+1}^{i,j} \right), \text{ if } \beta_{i,k+1}^{(1)} \geq \beta_{i,k}^{(1)}, \tag{7}$$

$$\eta_k^i \geq \beta_{i,k+1}^{(1)} \left(\left\lceil \frac{d_{k+1}^i}{C^1} \right\rceil - \sum_{j=1}^{n} \left\lceil \frac{C^j}{C^1} \right\rceil y_{k+1}^{i,j} \right) + \left(\beta_{i,k}^{(1)} - \beta_{i,k+1}^{(1)} \right) \left(\left\lceil \frac{d_k^i}{C^1} \right\rceil - \sum_{j=1}^{n} \left\lceil \frac{C^j}{C^1} \right\rceil y_k^{i,j} \right), \text{ if } \beta_{i,k+1}^{(1)} < \beta_{i,k}^{(1)}, \tag{8}$$

where $\eta_k^i = s_{k\ \ 1}^i + s_k^i + r_k^i + r_{k+1}^i$.

Remark 5. *Based on our preliminary computational experiments, we observed that adding a subset of the pairing inequalities for $k \in \{1, 3, 5, \ldots\}$ is more effective in reducing the overall solution time. Note that the mixing procedure can also be applied on base inequalities (5) corresponding to each subset of $\mathcal{T}$, but it leads to an exponential number of inequalities. Furthermore, since inequalities (4) are "knapsack-type" constraints, various cut-generation procedures known in the literature for knapsack problems can be utilized to derive cutting planes for m-DLS-MC-(W)B. We consider MIR and pairing cuts (a subset of mixing inequalities) so that $O(mT)$ number of cuts are added at the root node. This led to an average cut-generation time of 0.02 seconds in our computational experiments.*

Appendix B: Computational Results for m-DLS-MC-B with $n = 2$ and $n = 3$

We also perform experiments for m-DLS-MC-B with $n \in \{2, 3\}$ and $m \in \{2, 3, 4\}$ where the set of capacities, time periods, and all other input parameters are generated in the same way as done for m-DLS-MC-WB instances. We report the results for m-DLS-MC-B with $n = 2$ and $n = 3$ in Table 4. Based on these results, we observe that the average initial LP gap using GRB-DEF for instances with $n = 2$ and $n = 3$ is about 7.8% and 12.8%, respectively. On the other hand, the average initial LP gap using GRB-CUTS is 7% for instances with $n = 2$ and 11.6% for instances with $n = 3$. In comparison to GRB-DEF, the LD approach reduced these gaps to 4.6% for $n = 2$ and 6.5% for $n = 3$, which is about 49% improvement in 33 seconds. The LD approach also led to 46% gap improvement in comparison to the initial LP gaps obtained using GRB-CUTS. Again, GRB-DEF and GRB-CUTS were unable to solve 83% and 71% of the instances within a time limit of 2000 seconds, whereas the LD approach took about half a minute to significantly reduce the integrality gap.

Table 4: Computational Results for m-DLS-MC-B instances

n	T	m	$(C^1, \ldots, C^n)$	**GRB-DEF**				**GRB-CUTS**				**Lagrangian Decomposition**					
				RootGap%	**Avg. BIB**	**USI**		**RootGap%**	**Avg. BIB**	**USI**		**Time (in s)**	**#Iter**	**DEF-LD Gap%**	**DEF-Gap Improv%**	**CUTS-LD Gap%**	**CUTS-Gap Improv%**
						#USI	**Gap%**			**#USI**	**Gap%**						
2	100	2	(1270, 2120)	5.53	231231.1	7	1.04	4.52	230470.6	7	0.53	12.6	5	2.40	56.4	2.10	53.6
			(1310, 2570)	5.34	223747.3	8	1.16	4.58	223308.8	7	0.98	10.3	7	2.00	62.9	1.80	60.7
		3	(1270, 2120)	6.60	321389.5	8	1.20	5.69	320958.9	6	0.93	15.8	6	3.28	51.0	3.16	44.5
			(1310, 2570)	15.20	321389.5	10	1.50	13.44	320711.6	7	1.15	13.1	4	12.29	18.9	12.11	9.9
		4	(1270, 2120)	6.86	400703.9	9	1.04	6.57	399970.7	9	0.65	17.4	6	3.70	45.7	3.55	45.9
			(1310, 2570)	7.33	381103.3	8	1.09	7.03	380650.8	8	1.08	17.0	8	3.90	46.1	3.83	45.4
3	50	2	(970, 1690, 2620)	14.59	91644.7	7	0.93	13.86	91116.4	5	0.6	12.8	6	7.75	47.2	7.22	48.0
			(1310, 1750, 2120)	18.24	79170.9	7	1.08	17.12	78406.7	7	0.75	9.4	5	11.17	38.8	10.30	39.8
		3	(970, 1690, 2620)	13.20	150777.1	8	1.60	11.49	150137.5	8	1.17	13.6	7	8.12	38.3	7.73	32.7
			(1310, 1750, 2120)	10.04	132045.6	9	1.46	8.50	131320.0	6	0.58	14.9	9	3.60	64.4	3.07	63.9
		4	(970, 1690, 2620)	10.85	221631.9	10	1.47	9.91	221170.1	8	1.12	16.7	9	4.10	62.1	3.90	60.6
			(1310, 1750, 2120)	9.71	207126.9	8	1.21	8.92	206636.7	8	0.77	18.1	8	4.50	53.8	4.27	52.1