Algorithms for Stochastic Lot-Sizing Problems with Backlogging

Yongpei Guan
School of Industrial Engineering, University of Oklahoma, Norman OK 73019, USA
email: yguan@ou.edu http://coll.ou.edu/

As a traditional model in the operations research and management science domain, deterministic lot-sizing problem is embedded in many application problems such as production and inventory planning and has been consistently drawing attentions from researchers. In this paper we consider basic versions of lot-sizing models in which problem parameters are stochastic and develop corresponding scenario tree based stochastic lot-sizing models. For these models, we develop production path properties and a general dynamic programming framework based on these properties. The dynamic programming framework allows us to show that the optimal value function is piecewise linear and continuous, which enables us to develop polynomial time algorithms for several different problems, including those with backlogging and varying capacities under certain conditions. Moreover, we develop polynomial time algorithms that run in $\mathcal{O}(n^2)$ and $\mathcal{O}(n^2 T \log n)$ times respectively for stochastic uncapacitated and constant capacitated lot-sizing problems with backlogging, regardless of the scenario tree structure.

Key words: lot-sizing; dynamic programming; integer programming; stochastic programming

MSC2000 subject classification: Primary: 90C39, 90C10, 90C15; Secondary: 90B30, 90B05

OR/MS subject classification: Primary: Programming: integer and stochastic; Secondary: Production/scheduling: planning

---

1. Introduction As a traditional model in the operations research and management science domain, lot-sizing problem (i.e., see Nemhauser and Wolsey [23], Hopp and Spearman [17], and Pochet and Wolsey [24]) has been consistently drawing attentions from researchers. The traditional deterministic lot-sizing problem is to determine the amount to produce in each time period over a finite discrete horizon so as to satisfy the demand for each period while minimizing total setup, production, and inventory holding costs. This fundamental model is embedded in many application problems such as production and inventory planning (i.e., see Tempelmeier and Derstroff [29], Belvaux and Wolsey [6], Belvaux and Wolsey [7], Stadtler [28], and many others). Thus, understanding and exploiting this structure has been essential in developing approaches to more complicated, real-world problems.

Polynomial time algorithms have been studied extensively for the deterministic uncapacitated lot-sizing problem (ULS) and its variants. Most efficient polynomial time algorithms are based on the Wagner-Whitin property (i.e., see Wagner and Whitin [32]): no production is undertaken if inventory is available from the previous time period. An initial dynamic programming algorithm based on this property for the ULS problem runs in $\mathcal{O}(T^2)$ time (i.e., see Wagner and Whitin [32]), where T is the number of time periods. This was improved later in $\mathcal{O}(T \log T)$ time and $\mathcal{O}(T)$ time for the case without speculative moves (i.e., see Aggarwal and Park [1], Federgruen and Tzur [12], and Wagelmans et al. [31]). Polynomial time algorithm developments for variants of ULS include the constant capacity problem (i.e., see Florian and Klein [14] and van Hoesel and Wagelmans [30]), ULS with backlogging (i.e., see Federgruen and Tzur [13]), ULS with demand time windows (i.e., see Lee et al. [21]), and ULS with inventory bounds (i.e., see Atamtürk and Kücükyavuz [3]). Polynomial time algorithms also provide likelihood to find the compact description of the convex hull of all feasible solutions of the problem. Examples include the compact descriptions of the convex hulls of ULS studied by Bárány et al. [4, 5] and ULS with backlogging studied by Kücükyavuz and Pochet [20].

In practice, the assumption of known, deterministic data parameter is not necessarily realistic. For example, the demand for each time period is unknown in advance. Capacity can also be uncertain in production planning since it is affected by such factors as machine malfunctions and variability in productivity. To deal with uncertainties, for the case on inventory control and planning problems, significant research progress has been made on developing optimal inventory policies by assuming that demands for different time periods are mutually independent and the underlying random process is Markovian (i.e., see Scarf [26] for (s, S) policies, and many others). In this paper we assume all problem parameters including demand for a particular time period can be dependent on all historical information. More specifically, we adopt a stochastic programming approach (i.e., see Ruszczyński and Shapiro [25]) to address the uncertain problem parameters. Many application problems have recently been studied that contain the stochastic lot-sizing model as a submodel. Instances include stochastic capacity expansion problems (i.e.,

see Ahmed and Sahinidis [2]), stochastic batch-sizing problems (i.e., see Lulli and Sen [22]), and stochastic production planning problems (i.e., see Beraldi and Ruszczyński [8]).

Recently efficient algorithms have been studied for several special cases of the problem. For instance, the stochastic uncapacitated lot-sizing problem (SULS) with and without setup costs by Guan and Miller [16], SULS without setup costs by Huang and Ahmed [18], and SULS with random lead times by Huang and Kücükyavuz [19]. In this paper, we will study a more general setting of the stochastic lot-sizing problem: including varying capacities and backlogging. Since our problem can be capacitated and demand can be quite high in some scenarios, allowing the possibility of backorders is necessary to ensure that the problem has a feasible solution. We develop algorithms that will take advantage of the scenario tree structure. To the best of our knowledge, these algorithms are the first exact polynomial time algorithms for stochastic lot-sizing problems with backlogging and/or varying capacities.

In the remaining part of this paper. We introduce notation and a mathematical formulation for the general stochastic capacitated lot-sizing problem (SCLS) with backlogging in Section 2. In Section 3, we analyze SULS with backlogging, the uncapacitated version of this problem. We first state the *production path property* for SULS with backlogging, a fundamental optimality condition analogous to the Wagner-Whitin property. We then use this property and the tree structure to derive a dynamic programming algorithm that runs in $\mathcal{O}(n^2)$ time to solve SULS with backlogging, regardless of the scenario tree structure, where n is the number of nodes in the tree. Comparing to the work studied in Guan and Miller [16], we allow backlogging, which is more realistic since otherwise demands for every possible scenario should be satisfied and solution is too conservative. We also improve the computational complexity from $\mathcal{O}(n^2 \log n)$ to $\mathcal{O}(n^2)$. In Section 4, we describe a dynamic programming framework for general stochastic lot-sizing problems with backlogging and varying capacities. We prove that the optimal value function is piecewise linear and continuous. For the case that each non-leaf node contains at least two children, the dynamic programming framework is sufficient to provide an $\mathcal{O}(n^4)$ time algorithm to fully describe the optimal value function. In Section 5, an $\mathcal{O}(n^2 T \log n)$ time algorithm is developed for stochastic constant capacitated lot-sizing problem with backlogging, regardless of the scenario tree structure, where T represents the number of time periods. Finally, the insights and future research are discussed in Section 6.

2. Notation and a Mathematical Formulation In our stochastic setting, we assume that the uncertain problem parameters for the lot-sizing problem evolve as a discrete time stochastic process with finite probability space. The resulting information structure can be interpreted as a scenario tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with T levels (stages) where node $i \in \mathcal{V}$ in stage t of the tree gives the state of the system that can be distinguished by information available up to stage t. Accordingly, the time period for node i is defined as $t(i)$ and the set of nodes on the path from the root node to node i is defined as $\mathcal{P}(i)$. Let the probability associated with the state represented by node i be p_i and the set of node i's children be $\mathcal{C}(i)$. The decisions corresponding to node i are assumed to be made after observing the realizations of the problem parameters along the path from the root node to node i but are non-anticipative with respect to future realizations. Each node i in the scenario tree, except the root node (indexed as $i = 1$), has a unique parent i^-. Finally, the tree nodes are indexed according to the non-decreasing sequence of the cumulative demands from the root node to each node in the tree. For instance, $d_1 = d_{11} \leq d_{12} \leq \ldots \leq d_{1n-1} \leq d_{1n}$, where $d_{1i} = \sum_{j \in \mathcal{P}(i)} d_j$ corresponding to each node $i \in \mathcal{V}$.

With the objective of minimizing expected total costs, a multi-stage stochastic integer programming formulation of SCLS with backlogging can be described as follows:

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{V}} (\alpha_i x_i + \beta_i y_i + h_i s_i^+ + b_i s_i^-) & & \\
\text{s.t.} \quad & s_{i^-}^+ + s_i^- + x_i = d_i + s_i^+ + s_{i^-}^- & \forall\, i \in \mathcal{V}, & \quad (1) \\
& x_i \leq \mu_i y_i & \forall\, i \in \mathcal{V}, & \quad (2) \\
& x_i, s_i^+, s_i^- \geq 0,\ y_i \in \{0,1\} & \forall\, i \in \mathcal{V}. & \quad (3)
\end{aligned}$$

Decision variables x_i, s_i^+, and s_i^- represent the levels of production, inventory, and backorders at the end of time period $t(i)$ corresponding to the state defined by node i, and y_i is the setup variable for node i. Parameters α_i, h_i, b_i, and β_i represent unit production, holding, and backlogging costs as well as setup cost, and d_i and μ_i represent the demand and capacity in node i. We assume β_i positive to distinguish from non-setup cases and other parameters nonnegative. For notational brevity, probability

p_i is included in problem parameters α_i, β_i, h_i, and b_i. Constraint (1) represents the inventory flow balance and constraint (2) indicates the relationship between production and setup for each node i. Since at most one of s_i^+ and s_i^- will be positive in an optimal solution, for all $i \in \mathcal{V}$, it will be convenient throughout to refer to $s_i = s_i^+ - s_i^-$. Thus, if $s_i > 0$, it represents the level of inventory; and if $s_i < 0$, then $|s_i|$ represents the quantity of backlogging.

3. A Polynomial Time Algorithm for SULS with Backlogging Corresponding to any pair (i,k) such that $i \in \mathcal{V}$ and $k \in \mathcal{V}(i) \cap \mathcal{L}$ where $\mathcal{V}(i)$ represents the set of descents of node i including itself and $\mathcal{L}$ represents the set of leaf nodes, if $\mu_i \geq d_{ik}$ where $d_{ik} = d_{1k} - d_{1i^-}$, then SCLS with backlogging becomes an instance of SULS with backlogging. We first introduce the production path property, which defines optimality conditions for SULS with backlogging.

PROPOSITION 3.1 *(SULS Production Path Property) For any instance of SULS with backlogging, there exists an optimal solution (x^*, y^*, s^*) such that for each node $i \in \mathcal{V}$,*

$$\text{if } x_i^* > 0, \text{ then } x_i^* = d_{ik} - s_{i^-}^* \text{ for some node } k \in \mathcal{V}(i). \tag{4}$$

In other words, there always exists an optimal solution such that if we produce at a node i, then we produce exactly enough to satisfy demands along the path from node i to some descendant of node i. This can be proven by showing that any optimal solution that contains a node that violates (4) is a convex combination of optimal solutions, at least one of which contains one less node that violates (4). This proposition is a special case for the later Proposition 4.1. The proof for this proposition is omitted here.

The production path property implies there are n candidates for x_1 (production at the root node) in an optimal solution and the following two propositions hold.

PROPOSITION 3.2 *There exists an algorithm that runs in linear time for SULS with backlogging with two periods.*

PROPOSITION 3.3 *For any instance of SULS with backlogging, there exists an optimal solution (x^*, y^*, s^*) such that the inventory left after node $i \in \mathcal{V}$*

$$s_i^* = d_{1k} - d_{1i} \text{ for some node } k \in \mathcal{V}.$$

PROOF. This proof can be done by an induction method starting from the root node, e.g., node 1.

Without loss of generality, we assume the initial inventory is zero. According to Proposition 3.1, we have $s_1^* = d_{1k} - d_1$ for some node $k \in \mathcal{V}$. Thus, the conclusion holds for the root node.

Assume the conclusion holds for a node $i \in \mathcal{V}$ at time period $t(i)$, for instance, $s_i^* = d_{1k} - d_{1i}$ for some node $k \in \mathcal{V}$. We only need to prove that the conclusion holds for each node $\ell \in \mathcal{C}(i)$. Based on Proposition 3.1, we verify all two possible cases for production at node ℓ as follows:

Case 1: $x_\ell^* = 0$. For this case, $s_\ell^* = s_i^* - d_\ell = d_{1k} - d_{1i} - d_\ell = d_{1k} - d_{1\ell}$ for some node $k \in \mathcal{V}$.

Case 2: $s_i^* + x_\ell^* = d_{\ell k}$ for some $k \in \mathcal{V}(\ell)$. For this case, $s_\ell^* = s_i^* + x_\ell^* - d_\ell = d_{\ell k} - d_\ell = d_{1k} - d_{1\ell}$ for some node $k \in \mathcal{V}(\ell) \subseteq \mathcal{V}$.

All above two cases indicate that the induction step holds and therefore, our claim is true. □

Let $\mathcal{H}(i,s)$ represent the optimal value function for node $i \in \mathcal{V}$ when the inventory left from previous period is s (for notational brevity, we will use s rather than s_{i^-} for the second argument of this function).

For each node $i \in \mathcal{V}$ we have two options: production or non-production. If production occurs, then the value function for this node contains 1) setup, production and inventory costs corresponding to this node and 2) the cost for later periods. (Note that we will produce no less than what is required to satisfy demand in at least the current period as shown in Proposition 3.1, and so we will not incur backlogging costs if we produce.) We will call this function the *production value function* $\mathcal{H}_{\text{P}}(i,s)$. From the production path property, the production quantity at node i is $x_i = d_{ik} - s$ for some node $k \in \mathcal{V}(i)$

such that $d_{ik} > s$. Therefore

$$\mathcal{H}_{\rm P}(i,s) = \beta_i + \min_{k\in\mathcal{V}(i):d_{ik}>s}\left\{\alpha_i(d_{ik}-s) + h_i(d_{ik}-d_i) + \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, d_{ik}-d_i)\right\}. \tag{5}$$

Based on (5), it can be observed that $\mathcal{H}_{\rm P}(i,s)$ is right continuous and piecewise linear with the same slope $-\alpha_i$ for each piece. More specifically, letting

$$\phi(i,k) = \beta_i + \alpha_i d_{ik} + h_i(d_{ik}-d_i) + \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, d_{ik}-d_i) \tag{6}$$

and assuming there are r linear pieces, each piece of the production value function can be described as follows:

$$\begin{aligned}
&\mathcal{H}_{\rm P}(i,s) = \phi(i,k_1) - \alpha_i s \text{ if } s \le d_{ik_1}, \text{ where } k_1 = \operatorname{argmin}\{\phi(i,k) : k\in\mathcal{V}(i)\};\\
&\mathcal{H}_{\rm P}(i,s) = \phi(i,k_2) - \alpha_i s \text{ if } d_{ik_1} < s \le d_{ik_2}, \text{ where } k_2 = \operatorname{argmin}\{\phi(i,k) : k\in\mathcal{V}(i) \text{ and } d_{ik} > d_{ik_1}\};\\
&\quad\vdots\\
&\mathcal{H}_{\rm P}(i,s) = \phi(i,k_r) - \alpha_i s \text{ if } d_{ik_{r-1}} < s \le d_{ik_r}, \text{ where } k_r = \operatorname{argmin}\{\phi(i,k) : k\in\mathcal{V}(i) \text{ and } d_{ik} > d_{ik_{r-1}}\}.
\end{aligned} \tag{7}$$

Otherwise, if no production occurs, then the value function for this node contains 1) either inventory holding cost or backlogging cost corresponding to this node, depending on whether $s - d_i$ is positive or negative, and 2) the cost for later periods. We will call this function the *non-production value function* $\mathcal{H}_{\rm NP}(i,s)$. This function can be expressed as

$$\mathcal{H}_{\rm NP}(i,s) = \max\{h_i(s-d_i), -b_i(s-d_i)\} + \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, s-d_i). \tag{8}$$

Note that we can only exclude the possibility of production at i if $s \ge \max_{k\in\mathcal{V}(i)} d_{ik}$. The value function under this condition can be defined $\mathcal{H}(i,s) = \mathcal{H}_{\rm NP}(i,s)$. In all other cases (including negative values of s), we must consider both production and non-production because of the possibility of backordering. Therefore for any $s \le \max_{k\in\mathcal{V}(i)} d_{ik}$, the backward recursion function can be described as

$$\mathcal{H}(i,s) = \min\{\mathcal{H}_{\rm P}(i,s), \mathcal{H}_{\rm NP}(i,s)\}. \tag{9}$$

Algorithm and computational complexity analysis For a given zero initial inventory, we calculate and store the value function $\mathcal{H}(1,0)$. Based on Propositions 3.1 and 3.3, for each node $i\in\mathcal{V}$, it is sufficient to calculate and store $\mathcal{H}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for all $k\in\mathcal{V}$. This reduces the computational complexity of the problem significantly. Furthermore, the special tree structure will also help improve the computational efficiency of our algorithm.

The value functions $\mathcal{H}(i,s)$ are calculated and stored backwards starting from the last time period T. Before exploring the details of the algorithm, in the initial step, we

(1) set relationship indicator $\delta(i,k)$ between each node $i\in\mathcal{V}$ and each node $k\in\mathcal{V}(i)$. For instance, $\delta(i,k) = 1$ if node $k\in\mathcal{V}(i)$ and $\delta(i,k) = 0$, otherwise;
(2) use $\theta(i,s)$ to store $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,s)$ for which $s = d_{1k} - d_{1i}$ for all $k\in\mathcal{V}$ and initialize them to zero.

From above, it can be observed that $\delta(i,k)$ can be obtained in $\mathcal{O}(n^2)$ time based on the tree data structure. For instance, corresponding to each node $k\in\mathcal{V}$, set $\delta(i,k) = 1$ for each $i\in\mathcal{P}(k)$ and 0 otherwise. Thus, the initial step can be completed in $\mathcal{O}(n^2)$ time.

Now we calculate and store $\mathcal{H}(i,s)$ by induction starting from the last time period T. For each node i in time period T, e.g., each leaf node, the value function is

$$\mathcal{H}(i,s) = \begin{cases} \alpha_i(d_i - s) + \beta_i & \text{if } s < \underline{d}_i,\\ b_i(d_i - s) & \text{if } \underline{d}_i \le s < d_i,\\ h_i(s - d_i) & \text{if } s \ge d_i, \end{cases}$$

where

$$\underline{d}_i = d_i - \frac{\beta_i}{b_i - \alpha_i}.$$

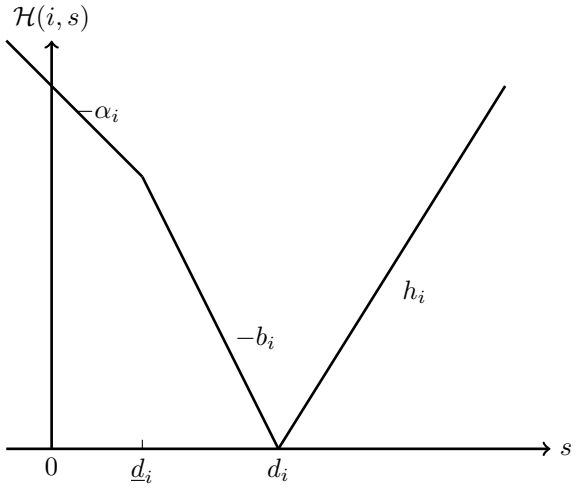

Figure 1: Value function $\mathcal{H}(i,s)$ for node i at time period T

Note here that $\underline{d}_i$ is not necessarily nonnegative. This value function is piecewise linear and continuous as shown in Figure 1. Thus, it is easy to verify that $\mathcal{H}(i,s)$ where $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$ can be calculated and stored in $\mathcal{O}(n)$ time. Meanwhile, increase $\theta(i^-, d_{1k} - d_{1i^-})$ by $\mathcal{H}(i, d_{1k} - d_{1i^-})$ for each node $k \in \mathcal{V}$. That is, each individual point in $\theta(i^-, s)$ is increased by the corresponding point value in $\mathcal{H}(i,s)$. This step takes $\mathcal{O}(n)$ time.

For each node i in the induction step, the following calculations will be executed:

Step 1: Calculate and store $\mathcal{H}_{\rm NP}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$: Note here, when we finish calculating and storing $\mathcal{H}(\ell, s)$ for each node $\ell \in \mathcal{C}(i)$, we have obtained $\theta(i,s) = \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ for $s = d_{1k} - d_{1i}$ for each node $k \in \mathcal{V}$. Thus, for each particular breakpoint $s = s'$ in $\mathcal{H}_{\rm NP}(i,s)$, the corresponding breakpoint $s = s' - d_i$ is stored in $\theta(i,s)$. Based on (8), $\mathcal{H}_{\rm NP}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$ can be obtained in $\mathcal{O}(n)$ time;

Step 2: Calculate and store $\mathcal{H}_{\rm P}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$: Since $\mathcal{H}_{\rm P}(i,s)$ is piecewise linear with the same slope $-\alpha_i$ for each piece, within each inventory interval corresponding to each piece, the smallest value $\phi(i,k)$ with $d_{ik} > s$ as indicated in (7) provides the value function of $\mathcal{H}_{\rm P}(i,s)$ in this interval. We can also observe that as shown in Step 1, $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i) = \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{1k} - d_{1i})$ is stored in $\theta(i,s)$, which is obtained when we finish calculating and storing $\mathcal{H}(\ell, s)$ for each node $\ell \in \mathcal{C}(i)$. Thus, $\phi(i,k)$ can be obtained in $\mathcal{O}(1)$ time according to (6). Now, $\mathcal{H}_{\rm P}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$ can be obtained according to non-increasing sequence of s values. We initialize $\phi = +\infty$. Starting from $s = d_{1k'} - d_{1i^-}$ where $k' = \text{argmax}\{d_{1j} : j \in \mathcal{V} \text{ and } \delta(i,j) = 1\}$, we calculate and store $\mathcal{H}_{\rm P}(i,s)$ for $s = d_{1k} - d_{1i^-}$ with $k = k', k'-1, \ldots, 1$ respectively. Corresponding to each breakpoint $s = d_{1k} - d_{1i^-}$ such that $\delta(i,k) = 1$, we first calculate $\phi(i,k)$ according to (6). Then we let $\mathcal{H}_{\rm P}(i, d_{1k} - d_{1i^-}) = \phi - \alpha_i(d_{1k} - d_{1i^-})$ if $\phi(i,k) \geq \phi$, or let $\mathcal{H}_{\rm P}(i, d_{1k} - d_{1i^-}) = \phi(i,k) - \alpha_i(d_{1k} - d_{1i^-})$ and update $\phi = \phi(i,k)$ if $\phi(i,k) < \phi$. Corresponding to each breakpoint $s = d_{1k} - d_{1i^-}$ such that $\delta(i,k) = 0$, we let $\mathcal{H}_{\rm P}(i, d_{1k} - d_{1i^-}) = \phi - \alpha_i(d_{1k} - d_{1i^-})$. This step can be completed in $\mathcal{O}(n)$ time;

Step 3: Calculate and store $\mathcal{H}(i,s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$: This step can be completed in $\mathcal{O}(n)$ time according to equation (9) since the number of breakpoints is bounded by $\mathcal{O}(n)$;

Step 4: Update $\theta(i^-, s)$ for $s = d_{1k} - d_{1i^-}$ for each node $k \in \mathcal{V}$: Increase $\theta(i^-, d_{1k} - d_{1i^-})$ by $\mathcal{H}(i, d_{1k} - d_{1i^-})$ for each node $k \in \mathcal{V}$. This step can be completed in $\mathcal{O}(n)$ time.

It can be observed from above that the total amount of work required for each node is bounded by $\mathcal{O}(n)$. Since these operations are required for each node $i \in \mathcal{V}$, the following conclusion holds.

THEOREM 3.1 *The general SULS with backlogging can be solved in $\mathcal{O}(n^2)$ time, regardless of the scenario tree structure.*

REMARK 3.1 *The general SULS with backlogging problem for which the initial inventory level is unknown and is itself a decision variable can be transformed into another general SULS with backlogging problem with zero initial inventory by adding a dummy root node* 0 *as the parent node of node* 1 *with zero production, setup and inventory costs as well as zero demand.*

REMARK 3.2 *In this paper, a more efficient algorithm for the general SULS with backlogging is developed comparing to the one studied in Guan and Miller [16] in which backlogging is not allowed and the computational complexity is* $\mathcal{O}(n^2 \max\{\mathcal{C}, \log n\})$ *where* $\mathcal{C} = \max_{i\in\mathcal{V}} |\mathcal{C}(i)|$.

4. A Dynamic Programming Framework for SCLS with Backlogging Motivated by the polynomial time algorithm development for SULS with backlogging in Section 3, we further investigate the computational complexity for general stochastic lot-sizing problems that include variant capacities and backlogging.

It is easy to verify that the SCLS problem with backlogging under the general scenario tree structure is $\mathcal{NP}$-hard since the deterministic capacitated lot-sizing problem is $\mathcal{NP}$-hard as shown in Bitran and Yanasse [9], which is a special case of the general SCLS with backlogging in which $\mathcal{C}(i) = 1$ for each node $i \in \mathcal{V}\setminus\mathcal{L}$ (e.g., one path). In this section, we develop a dynamic programming framework and analyze the computational complexity for SCLS with backlogging for the case that $\mathcal{C}(i) \geq 2$ for each node $i \in \mathcal{V}\setminus\mathcal{L}$. We focus on studying full description of the value function $\mathcal{H}(i, s)$, which provides more insights of the problem that include the sensitivity value at each individual inventory level. In our approach, we show that the value function $\mathcal{H}(i, s)$ is piecewise linear and continuous; these value functions can therefore be analyzed in terms of breakpoints (i.e., points in the domain at which slope change occurs), functional evaluations of breakpoints, and the slopes of the segments to the right of the breakpoints. We achieve the computational complexity bound by analyzing the number of breakpoints whose evaluations and right slopes must be stored at each node, and by analyzing how long these computations take. This approach will help us explore the insights on how scenario tree structure affects the computational complexity. We first derive a more generalized production path property as follows.

PROPOSITION 4.1 *(SCLS Production Path Property) For any instance of SCLS with backlogging, there exists an optimal solution* (x^*, y^*, s^*) *such that for each node* $i \in \mathcal{V}$,

$$\begin{aligned} &\text{if } 0 < x_i^* < \mu_i, \text{ then } x_i^* + s_{i^-}^* = d_{ik} - \textstyle\sum_{j\in S} \mu_j \text{ for some node } k \in \mathcal{V}(i) \text{ and } S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(i) \\ &\qquad \text{and } x_j^* = 0 \text{ or } \mu_j \text{ for each } j \in \mathcal{P}(k)\setminus\mathcal{P}(i). \end{aligned} \tag{10}$$

In other words, there always exists an optimal solution such that if we produce at a node i, then we produce enough to satisfy demands along the path from node i to some descendant of node i besides some nodes along this path producing at their capacities. For instance, set S represents the set of nodes on the path from node i to node k at which we produce at capacity. The proof details are shown in Appendix. From Proposition 4.1, it is easy to observe that

COROLLARY 4.1 *There exists an algorithm that runs in linear time for SCLS with backlogging with two periods.*

4.1 Value functions Comparing to SULS with backlogging, corresponding to each node $i \in \mathcal{V}$, there are three options for the value function of the capacitated case: 1) Non-production, denoted as $\mathcal{H}_{\text{NP}}(i, s)$, 2) Production at capacity, denoted as $\mathcal{H}_{\text{P}}^{\mu_i}(i, s)$, and 3) Production less than capacity, denoted as $\mathcal{H}_{\text{P}}^{\mu}(i, s)$.

- Non-production: In this case, the value function for this node contains inventory holding or backlogging costs corresponding to this node and the cost for later periods.

$$\mathcal{H}_{\text{NP}}(i, s) = \max\{h_i(s - d_i), -b_i(s - d_i)\} + \sum_{\ell\in\mathcal{C}(i)} \mathcal{H}(\ell, s - d_i). \tag{11}$$

- Production at capacity: In this case, the production quantity is μ_i and the value function for this node contains setup, production, and inventory holding or backlogging costs corresponding to this node and the cost for later periods.

$$\mathcal{H}_{\text{P}}^{\mu_i}(i, s) = \beta_i + \alpha_i\mu_i + \max\{h_i(\mu_i + s - d_i), -b_i(\mu_i + s - d_i)\} + \sum_{\ell\in\mathcal{C}(i)} \mathcal{H}(\ell, \mu_i + s - d_i). \tag{12}$$

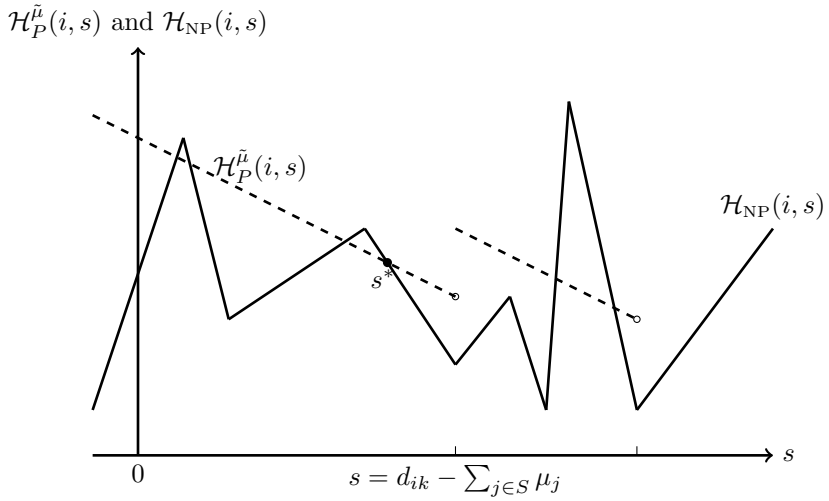

Figure 2: Value functions $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ vs $\mathcal{H}_{\rm NP}(i,s)$

- Production according to Proposition 4.1: In this case, with the consideration of production capacity, the value function is

$$\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) = \min_{k\in\mathcal{V}(i):d_{ik}-\sum_{j\in S}\mu_j-\mu_i\le s<d_{ik}-\sum_{j\in S}\mu_j \text{ and } S\subseteq\mathcal{P}(k)\setminus\mathcal{P}(i)} \mathcal{H}_{\rm P}^{k,S}(i,s), \tag{13}$$

where

$$\begin{aligned}\mathcal{H}_{\rm P}^{k,S}(i,s) &= \beta_i + \alpha_i(d_{ik} - \sum_{j\in S}\mu_j - s)\\ &+ \max\left\{h_i(d_{ik} - \sum_{j\in S}\mu_j - d_i), -b_i(d_{ik} - \sum_{j\in S}\mu_j - d_i)\right\}\\ &+ \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, d_{ik} - \sum_{j\in S}\mu_j - d_i).\end{aligned} \tag{14}$$

Therefore, for any $s \le \max_{j\in\mathcal{V}(i)} d_{ij}$, the backward recursion function can be described as

$$\mathcal{H}(i,s) = \min\left\{\mathcal{H}_{\rm P}(i,s), \mathcal{H}_{\rm NP}(i,s)\right\},$$

where $\mathcal{H}_{\rm P}(i,s) = \min\{\mathcal{H}_{\rm P}^{\mu_i}(i,s), \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)\}$.

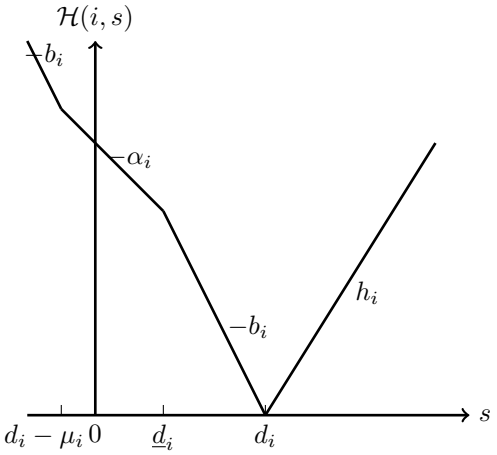

Figure 3: SCLS value function $\mathcal{H}(i,s)$ for node i at time period T

In the remaining part of this section, we analyze the property of the optimal value function $\mathcal{H}(i,s)$ for the case that $\mathcal{C}(i) \geq 2$ for each node $i \in \mathcal{V} \setminus \mathcal{L}$. We study an important property after describing two observations.

OBSERVATION 1 *The summation of piecewise linear and continuous functions is still a piecewise linear and continuous function.*

OBSERVATION 2 *The minimum of two piecewise linear and continuous functions is still a piecewise linear and continuous function.*

LEMMA 4.1 *The value function $\mathcal{H}_P^{\tilde{\mu}}(i,s)$ for each node $i \in \mathcal{V}$ is right continuous and piecewise linear with the same slope $-\alpha_i$. Corresponding to each piece of the production value function ending with $s = d_{ik} - \sum_{j\in S}\mu_j$, if it is intercrossing with $\mathcal{H}_{NP}(i,s)$ and both $\mathcal{H}_{NP}(i,s)$ and $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,S)$ are piecewise linear and continuous, then there exists a point s^* such that $\mathcal{H}_{NP}(i,s^*) = \mathcal{H}_P^{\tilde{\mu}}(i,s^*)$ and $\mathcal{H}_{NP}(i,s) < \mathcal{H}_P^{\tilde{\mu}}(i,s)$ for each $s > s^*$ within the piece.*

PROOF. It is easy to observe from (13) and (14) that $\mathcal{H}_P^{\tilde{\mu}}(i,s)$ is right continuous and piecewise linear with the same slope $-\alpha_i$. Now assume that a piece of the production line ends with the pair (k,S) where

$$k = \text{argmin}\left\{\mathcal{H}_{\text{P}}^{k,S}(i,s) | k \in \mathcal{V}(i) : d_{ik} - \sum_{j\in S}\mu_j - \mu_i \leq s < d_{ik} - \sum_{j\in S}\mu_j \text{ and } S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(i)\right\}.$$

Let $s' = d_{ik} - \sum_{j\in S}\mu_j - \epsilon$.

(1) If $k = i$, then $S = \emptyset$ and

$$\begin{aligned}\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s') - \mathcal{H}_{\text{NP}}(i,s') &= \beta_i + \alpha_i\epsilon + \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,0)\\ &- \left(b_i\epsilon + \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,-\epsilon)\right).\end{aligned}$$

(2) If $k \neq i$, then

$$\begin{aligned}\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s') - \mathcal{H}_{\text{NP}}(i,s') &= \beta_i + \alpha_i\epsilon\\ &+ \max\left\{h_i(d_{ik} - \sum_{j\in S}\mu_j - d_i), -b_i(d_{ik} - \sum_{j\in S}\mu_j - d_i)\right\}\\ &+ \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, d_{ik} - \sum_{j\in S}\mu_j - d_i)\\ &- \max\left\{h_i(d_{ik} - \sum_{j\in S}\mu_j - \epsilon - d_i), -b_i(d_{ik} - \sum_{j\in S}\mu_j - \epsilon - d_i)\right\}\\ &- \sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, d_{ik} - \sum_{j\in S}\mu_j - \epsilon - d_i).\end{aligned}$$

For both cases, it can be observed that there exists an $\epsilon > 0$ such that $\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s') - \mathcal{H}_{\text{NP}}(i,s') \geq 0$ since $\beta_i > 0$ and $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,s)$ is continuous according to the assumption. Therefore, if this piece of production line is intercrossing with $\mathcal{H}_{\text{NP}}(i,s)$, then, as shown in Figure 2, there exists a point s^* such that $\mathcal{H}_{\text{NP}}(i,s^*) = \mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s^*)$ and $\mathcal{H}_{\text{NP}}(i,s) < \mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s)$ for each $s > s^*$ within the piece since $\mathcal{H}_{\text{NP}}(i,s)$ is continuous according to our assumption. □

PROPOSITION 4.2 *The value functions $\mathcal{H}(i,s), \mathcal{H}_P^{\mu_i}(i,s)$, and $\mathcal{H}_{NP}(i,s)$ for each node $i \in \mathcal{V}$ are piecewise linear and continuous.*

PROOF. By induction.

BASE CASE: For a leaf node in time period T, we only need to consider $\mathcal{H}(i,s)$. If $\mu_i(b_i-\alpha_i)>\beta_i$, that is, to certain amount, it is better to produce instead of backlogging, then as shown in Figure 3 the value function for each node i is

$$\mathcal{H}(i,s)=\begin{cases}\beta_i+\alpha_i\mu_i+b_i(d_i-\mu_i-s) & \text{if } s<d_i-\mu_i,\\ \alpha_i(d_i-s)+\beta_i & \text{if } d_i-\mu_i\le s<\underline{d}_i,\\ b_i(d_i-s) & \text{if } \underline{d}_i\le s<d_i,\\ h_i(s-d_i) & \text{if } s\ge d_i,\end{cases}$$

where

$$\underline{d}_i=d_i-\frac{\beta_i}{b_i-\alpha_i}.$$

This value function is piecewise linear and continuous. Otherwise, if $\mu_i(b_i-\alpha_i)\le\beta_i$, that is, backlogging is always better than production, then the value function for each node i is

$$\mathcal{H}(i,s)=\begin{cases}b_i(d_i-s) & \text{if } s<d_i,\\ h_i(s-d_i) & \text{if } s\ge d_i.\end{cases}$$

This value function is also piecewise linear and continuous. Therefore, the value function for each leaf node is piecewise linear and continuous.

INDUCTIVE STEP: Using the induction hypothesis and based on (11), the non-production value function $\mathcal{H}_{\text{NP}}(i,s)$ is the sum of piecewise linear and continuous functions and is therefore itself piecewise linear and continuous according to Observation 1.

Similarly, from (12), the production value function for the case that we produce at the capacity $\mathcal{H}_{\text{P}}^{\mu_i}(i,s)$ is piecewise linear and continuous based on induction. Then, based on Observation 2, the value function $\mathcal{H}'(i,s)=\min\{\mathcal{H}_{\text{NP}}(i,s),\mathcal{H}_{\text{P}}^{\mu_i}(i,s)\}$ is piecewise linear and continuous.

We also notice that as shown in Lemma 4.1, $\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s)$ is a piecewise linear and right continuous function of s with the same slope at each piece. Moreover, the end point of each interval of the production value function is either $s=d_{ik}-\sum_{j\in S}\mu_j-\mu_i$ (i.e., production reaches capacity) or $s=d_{ik}-\sum_{j\in S}\mu_j$ such that $S\subseteq\mathcal{P}(k)\setminus\mathcal{P}(i)$ for some node $k\in\mathcal{V}(i)$. Note here it may happen that both end points of each piece are in the form $s=d_{ik}-\sum_{j\in S}\mu_j$ or both are in the form $s=d_{ik}-\sum_{j\in S}\mu_j-\mu_i$.

Now consider $\mathcal{H}(i,s)=\min\{\mathcal{H}'(i,s),\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s)\}$. We only need to consider the continuity at the jump points of $\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s)$ (i.e., end points of each piece of the production value function $\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s)$). If the jump point is in the form of $s=d_{ik}-\sum_{j\in S}\mu_j$, then according to Lemma 4.1, we have

$$\mathcal{H}'(i,s)\le\mathcal{H}_{\text{NP}}(i,s)\le\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s). \tag{15}$$

Otherwise if the jump point is in the form of $s=d_{ik}-\sum_{j\in S}\mu_j-\mu_i$, then at this point, the production reaches its capacity and

$$\mathcal{H}'(i,s)\le\mathcal{H}_{\text{P}}^{\mu_i}(i,s)=\mathcal{H}_{\text{P}}^{\tilde{\mu}}(i,s). \tag{16}$$

Therefore, we can claim that $\mathcal{H}(i,s)$ is piecewise linear and continuous. The conclusion holds. □

This result is interesting in light of the fact that the value function for a special case of the problem, SULS without backlogging, is not continuous, only right continuous (i.e., see Guan and Miller [16]).

4.2 Number of breakpoints in the value function The computational complexity of the problem depends on the number of breakpoints. We let *breakpoints* represent the s values at which the slope of the value function changes or the discontinuity happens. Since the optimal value function is piecewise linear and continuous, the value function $\mathcal{H}(i,s)$ can be computed and stored in terms of breakpoints, evaluations of breakpoints, and slopes immediately to the right of the breakpoints.

We define that a breakpoint $s=s'$ is a "convex" breakpoint of $\mathcal{H}(i,s)$ if

$$\lim_{s\to s'^-}\partial\mathcal{H}(i,s)/\partial s<\lim_{s\to s'^+}\partial\mathcal{H}(i,s)/\partial s$$

or a "concave" breakpoint, otherwise. Before describing the total number of breakpoints, we first present the following important proposition.

PROPOSITION 4.3 *All "convex" breakpoints in $\mathcal{H}(i,s)$, $\mathcal{H}_{NP}(i,s)$, and $\mathcal{H}_P^{\mu_i}(i,s)$ are in the form $s = d_{ik} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i^-)$.*

PROOF. From Figure 3, it is easy to observe that the conclusion holds for each leaf node. Then based on the induction steps, considering equation (11) for the general case, the summation operation $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell, s-d_i)$ does not generate any new "convex" breakpoints. These existing "convex" breakpoints are in the form of $s = d_{ik} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i)$, which is due to the fact that the "convex" breakpoints for each node $\ell \in \mathcal{C}(i)$ are in the form of $s = d_{\ell k} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(\ell)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i)$ based on induction hypothesis. The only new "convex" breakpoint generated for $\mathcal{H}_{\rm NP}(i,s)$ is $s = d_i$. Similarly, from (12), the only new "convex" breakpoint generated for $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ is $s = d_i - \mu_i$. Both of these breakpoints are in the form $s = d_{ik} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i^-)$.

It can be verified that no new "convex" breakpoints will be generated by taking the minimum of two piecewise linear and continuous functions. We also notice that $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ stated in (13) is a piecewise linear function with the same slope at each piece. It can also be verified that no new "convex" breakpoints will be generated by taking the minimum of a piecewise linear and continuous function and a piecewise linear function with the same slope at each piece. Therefore, all "convex" breakpoints in $\mathcal{H}(i,s) = \min\{\mathcal{H}_{\rm NP}(i,s), \mathcal{H}_{\rm P}^{\mu_i}(i,s), \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)\}$ are in the form stated in the claim and the conclusion holds. □

Let $B(i)$ represent the set of breakpoints for the value function $\mathcal{H}(i,s)$. Starting from time period T as shown in Figure 3, we consider the number of breakpoints corresponding to each node $i \in \mathcal{V}$. First, we can observe from Figure 3 that the number of breakpoints for each leaf node is at most 4 including the breakpoint $s = -\infty$. Then we can observe from (11) that

LEMMA 4.2 *The number of breakpoints of the non-production value function $|B_{NP}(i)| \le \sum_{\ell\in\mathcal{C}(i)}|B(\ell)|$ if $|\mathcal{C}(i)| \ge 2$ for each $i \in \mathcal{V}\setminus\mathcal{L}$.*

PROOF. It is due to the fact that one new breakpoint $s = d_i$ will be generated and $s = -\infty$ is the common breakpoint for each $\ell \in \mathcal{C}(i)$. □

Accordingly, we denote the set of nodes in $B_{\rm NP}(i)$ as "old" breakpoints since they are all there, except the breakpoint $s = d_i$, corresponding to the set of nodes in $B(\ell)$, $\ell \in \mathcal{C}(i)$. Besides this, from (11) and (12), we can observe that

LEMMA 4.3 *The breakpoints for $\mathcal{H}_P^{\mu_i}(i,s)$ belong to the node set in which all the "old" breakpoints in $\mathcal{H}_{NP}(i,s)$ are shifted to the left by μ_i.*

Now, we study the number of breakpoints generated by $\mathcal{H}(i,s)$. That is, calculate the total number of breakpoints for $\min\{\mathcal{H}_{\rm NP}(i,s), \mathcal{H}_{\rm P}^{\mu_i}(i,s), \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)\}$. Due to the special properties of $\mathcal{H}(i,s)$ and $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$, we can bound the number of breakpoints in $\mathcal{H}(i,s)$ to be $4|B_{\rm NP}(i)|$ as shown in the following lemma.

LEMMA 4.4 *The number of breakpoints generated by $\mathcal{H}(i,s)$ is at most $4|B_{NP}(i)|$.*

PROOF. According to Lemma 4.3, the number of breakpoints of $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ is the same as the number of breakpoints of $\mathcal{H}_{\rm NP}(i,s)$. It is noticed that both the value functions $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ are piecewise linear and continuous, and they have the same first breakpoint $s = -\infty$. It is also easy to observe that $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ reach $+\infty$ as $s = +\infty$. As shown for the leaf nodes, we do not need to store the last point $s = +\infty$. Thus, to prove the claim, in the following, we show that, within each inventory interval between two consecutive breakpoints in $\mathcal{H}_{\rm NP}(i,s)$, the number of breakpoints in $\mathcal{H}(i,s)$ is at most twice the number of breakpoints originally in $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$. Denote the breakpoints for $\mathcal{H}_{\rm NP}(i,s)$ as $s = s^{[1]}, s^{[2]}, \ldots, s^{[m]}$ such that $-\infty = s^{[1]} \le s^{[2]} \le \ldots \le s^{[m]}$, and the interval $[s^{[k]}, s^{[k+1]})$ as the k^{th} interval in $\mathcal{H}_{\rm NP}(i,s)$. We want to prove

$$|B(i)|_k \le 2(|B_{\rm NP}(i)|_k + |B_{\rm P}^{\mu_i}(i)|_k), \tag{17}$$

where $|B(i)|_k$, $|B_{\rm NP}(i)|_k$, and $|B_{\rm P}^{\mu_i}(i)|_k$ represent the number of breakpoints for the functions $\mathcal{H}(i,s)$, $\mathcal{H}_{\rm NP}(i,s)$, and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ in the interval $[s^{[k]}, s^{[k+1]})$ respectively.

First of all, according to (15) and (16) in the proof for Proposition 4.2, corresponding to each breakpoint in $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$, we have either $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \geq \mathcal{H}_{\rm NP}(i,s)$ or $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) = \mathcal{H}_{\rm P}^{\mu_i}(i,s)$. Therefore, the original breakpoints in $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ are not needed to be counted in calculating the number of breakpoints in $\mathcal{H}(i,s)$. In the following, we define the new breakpoints as the extra breakpoints generated by taking the minimum of $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ and $\min\{\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s), \mathcal{H}_{\rm NP}(i,s)\}$. We prove (17) in two cases: $\mathcal{H}_{\rm NP}(i,s^{[k]}) > \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s^{[k]})$ and $\mathcal{H}_{\rm NP}(i,s^{[k]}) \leq \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s^{[k]})$ respectively.

Case 1: $\mathcal{H}_{\rm NP}(i,s^{[k]}) > \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s^{[k]})$. For this case, we consider two scenarios in which there is only one piece of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and there are multiple pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in the interval $[s^{[k]}, s^{[k+1]})$ respectively.

Scenario 1: There is only one piece of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in this interval. We first assume that there is intercrossing between $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in the interval $[s^{[k]}, s^{[k+1]})$. Under this scenario, there is only one intercrossing point generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in this interval. Denote it as $s = s^*$ and accordingly the k^{th} interval $[s^{[k]}, s^{[k+1]})$ is splitted into two subintervals $[s^{[k]}, s^*)$ and $[s^*, s^{[k+1]})$. We describe it as the k_1^{th} and k_2^{th} intervals. According to Proposition 4.3, we have the following claim:

$$\begin{aligned}&\text{There are no ``convex'' breakpoints for } \mathcal{H}_{\rm P}^{\mu_i}(i,s) \text{ in the interval}\\ &\{s \in [s^{[k]}, s^{[k+1]}) : \mathcal{H}_{\rm P}^{\mu_i}(i,s) < \min\{\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s), \mathcal{H}_{\rm NP}(i,s)\}. \end{aligned} \tag{18}$$

Otherwise, if there is a "convex" breakpoint in this interval, i.e., $s = s'$, then there will be another piece of production line $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ connecting this "convex" breakpoint with the corresponding breakpoint $s = s' + \mu_i$ in $\mathcal{H}_{\rm NP}(i,s)$. Contradiction.

Assume there is at least one new breakpoint generated in the k^{th} interval. Otherwise the conclusion (17) is trivial. Let $|B'(i)|_{k_1}$ and $|B'(i)|_{k_2}$ be the numbers of new breakpoints in the k_1^{th} and k_2^{th} intervals respectively. Then, it is easy to verify that

(1) If $|B'(i)|_{k_2} = 0$, then either there is one new breakpoint generated in the k_1^{th} interval or there are two new breakpoints generated in the k_1^{th} interval with at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted. For the latter case, the breakpoint $s = s^*$ will not be in $\mathcal{H}(i,s)$. Then, with the consideration of the breakpoint $s = s^{[k]}$ deleted, we have $|B(i)|_k \leq 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k)$ for both cases.

(2) If $|B'(i)|_{k_2} > 0$ and $s = s^*$ is a breakpoint in $\mathcal{H}(i,s)$, then it can be verified that there is only one new breakpoint generated in the k_2^{th} interval due to (18). If $|B'(i)|_{k_1} = 0$, then $|B(i)|_k \leq |B_{\rm P}^{\mu_i}(i)|_k + 2 \leq 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k)$ since $|B_{\rm NP}(i)|_k = 1$. Otherwise if $|B'(i)|_{k_1} > 0$, then there is only one new breakpoint generated in the k_1^{th} interval and at least one original breakpoint in $B_{\rm P}^{\mu_i}(i)$ is deleted. Therefore, $|B(i)|_k \leq |B_{\rm P}^{\mu_i}(i)|_k + 3 - 1 \leq 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k)$ since $|B_{\rm NP}(i)|_k = 1$.

(3) If $|B'(i)|_{k_2} > 0$ and $s = s^*$ is not a breakpoint in $\mathcal{H}(i,s)$, then it can be verified that $|B_{\rm P}^{\mu_i}(i)|_{k_1} \geq |B'(i)|_{k_1} - 1$ and $|B_{\rm P}^{\mu_i}(i)|_{k_2} \geq |B'(i)|_{k_2} - 1$. Thus, we have

$$\begin{aligned} |B(i)|_k &\leq |B_{\rm P}^{\mu_i}(i)|_{k_1} + |B_{\rm P}^{\mu_i}(i)|_{k_2} + |B'(i)|_{k_1} + |B'(i)|_{k_2} \\ &\leq 2(|B_{\rm P}^{\mu_i}(i)|_{k_1} + |B_{\rm P}^{\mu_i}(i)|_{k_2} + 1) \\ &= 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k). \end{aligned}$$

Therefore, the conclusion (17) holds.

If there is no intercrossing between $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in the interval $[s^{[k]}, s^{[k+1]})$, then we do not need to consider $\mathcal{H}_{\rm NP}(i,s)$. There is one new breakpoint generated or there are two new breakpoints generated with at least one original breakpoint in $B_{\rm P}^{\mu_i}(i)$ deleted by taking the minimum of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$. For both cases, we have

$$|B(i)|_k \leq 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k) - 1. \tag{19}$$

Scenario 2: There are multiple pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in this interval $[s^{[k]}, s^{[k+1]})$. For instance, denote them as subintervals $r_1, r_2, \ldots, r_m$ with the r_j^{th} interval represented by $[s^{[r_j]}, s^{[r_{j+1}]})$ where $s^{[r_1]} = s^{[k]}$ and $s^{[r_{m+1}]} = s^{[k+1]}$. Without loss of generality, we can assume that all pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ are below or intercrossing with $\mathcal{H}_{\rm NP}(i,s)$, because those pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ that are above and not intercrossing with $\mathcal{H}_{\rm NP}(i,s)$ are dominated by $\mathcal{H}_{\rm NP}(i,s)$ and this subinterval can be

combined with the previous subinterval in the analysis. Under this scenario, we have $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \leq \mathcal{H}_{\rm NP}(i,s)$ at each breakpoint of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and there may be multiple intercrossing points generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in this interval $[s^{[k]}, s^{[k+1]})$ as shown in Figure 4. We also notice that at each breakpoint $s = s^{[r_j]}, 2 \leq j \leq m$, the production value function $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ reaches its capacity. If not, for example, $s = s^{[r_j]}$ for some $2 \leq j \leq m$ is not a breakpoint that $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ reaches its capacity, then according to the definition of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in equation (13), we should have $s^{[r_j]} = d_{ik} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i)$. Accordingly, we should have $\lim_{s\to s^{[r_j]-}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \leq \lim_{s\to s^{[r_j]+}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ according to the definition of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$. Then, the whole piece of production line in the r_j^{th} subinterval will be above $\mathcal{H}_{\rm NP}(i,s)$ and it does not need to be considered. Contradiction. Based on the similar argument, we have $\lim_{s\to s^{[r_j]-}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \geq \lim_{s\to s^{[r_j]+}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ for $2 \leq j \leq m$.

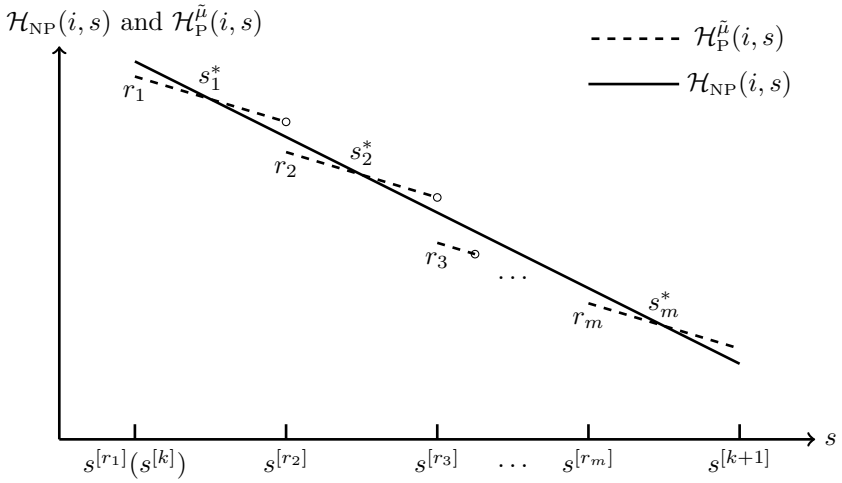

Figure 4: Multiple pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in the k^{th} interval of $\mathcal{H}_{\rm NP}(i,s)$

For the first subinterval $[s^{[r_1]}, s^{[r_2]})$ (i.e., subinterval r_1), we can follow the same argument as in Scenario 1 and $|B(i)|_{r_1} \leq 2(|B_{\rm P}^{\mu_i}(i)|_{r_1} + |B_{\rm NP}(i)|_{r_1})$. For each following subinterval, we have $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) = \mathcal{H}_{\rm P}^{\mu_i}(i,s)$ at the beginning of the subinterval (i.e., $s = s^{[r_j]}, 2 \leq j \leq m$) because these breakpoints represent that the production reaches its capacity. In the following, without loss of generality, we assume that there is at least one new breakpoint generated or $s = s_j^*$ is a breakpoint in $\mathcal{H}(i,s)$ for each subinterval r_j with $2 \leq j \leq m$. For the cases $2 \leq j \leq m-1$, we have the following analysis.

(i) If s_j^* does not exist (i.e., no intercrossing between $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$), then there is only one new breakpoint generated and $s = s^{[r_j]}$ should be an original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$. Thus

$$|B(i)|_{r_j} \leq |B_{\rm P}^{\mu_i}(i)|_{r_j} + 1 \leq 2|B_{\rm P}^{\mu_i}(i)|_{r_j} - 1 \tag{20}$$

since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \geq 2$. In the following (ii), (iii), and (iv), we assume s_j^* exists, and $|B'(i)|_{r_j}^1$ and $|B'(i)|_{r_j}^2$ are the numbers of new breakpoints in the intervals $[s^{[r_j]}, s_j^*)$ and $[s_j^*, s^{[r_{j+1}]})$ respectively.

(ii) If $|B'(i)|_{r_j}^1 > 0$, then $s = s^{[r_j]}$ should be an original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ and there is only one new breakpoint generated based on the condition described in (18) with at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted. Accordingly, $s = s_j^*$ should not be in $\mathcal{H}(i,s)$. If $|B'(i)|_{r_j}^2 = 0$, then it is easy to see that $|B(i)|_{r_j} \leq |B_{\rm P}^{\mu_i}(i)|_{r_j} + 1 - 1 \leq 2|B_{\rm P}^{\mu_i}(i)|_{r_j} - 1$ since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \geq 2$ in this case. If $|B'(i)|_{r_j}^2 > 0$, then there are at most two more new breakpoints generated in the interval $[s_j^*, s^{[r_{j+1}]})$. If there is only one new breakpoint in the interval $[s_j^*, s^{[r_{j+1}]})$, then $|B(i)|_{r_j} \leq |B_{\rm P}^{\mu_i}(i)|_{r_j} + 2 - 1 \leq 2|B_{\rm P}^{\mu_i}(i)|_{r_j} - 1$ since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \geq 2$ in this case. Otherwise if there are two new breakpoints generated in the interval $[s_j^*, s^{[r_{j+1}]})$, then there is at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted in the interval $[s_j^*, s^{[r_{j+1}]})$ and we have $|B(i)|_{r_j} \leq |B_{\rm P}^{\mu_i}(i)|_{r_j} + 3 - 2 \leq 2|B_{\rm P}^{\mu_i}(i)|_{r_j} - 1$ since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \geq 3$ in this case.

(iii) If $|B'(i)|_{r_j}^1 = 0$ and $s = s_j^*$ is a breakpoint in $\mathcal{H}(i,s)$, then $s = s^{[r_j]}$ should be an original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$. At least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ is deleted and at most one new breakpoint is generated in the interval $[s_j^*, s^{[r_{j+1}]})$ based on the condition described in (18). Then we have $|B(i)|_{r_j} \le |B_{\rm P}^{\mu_i}(i)|_{r_j} + 2 - 1 \le 2|B_{\rm P}^{\mu_i}(i)|_{r_j} - 1$ since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \ge 2$ in this case.

(iv) If $|B'(i)|_{r_j}^1 = 0$ and $s = s_j^*$ is not a breakpoint in $\mathcal{H}(i,s)$, then it can be verified that there are two new breakpoints generated in the interval $[s_j^*, s^{[r_{j+1}]})$ with at least one original breakpoint in $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ deleted due to (18) and $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i, s^{[j+1]}) = \mathcal{H}_{\rm P}^{\mu_i}(i, s^{[j+1]})$. Thus, we have $|B(i)|_{r_j} \le |B_{\rm P}^{\mu_i}(i)|_{r_j} + 2 - 1 \le 2|B_{\rm P}^{\mu_i}(i)|_{r_j}$ since $|B_{\rm P}^{\mu_i}(i)|_{r_j} \ge 1$ for this case.

For the case that $j = m$, we have the similar argument as the above $j < m$ cases except for (iv). In (iv), it can happen that there is only one new breakpoint generated in the interval $[s_m^*, s^{[k+1]})$. For this case, if $s = s^{[r_m]}$ is an original breakpoint, we have $|B(i)|_{r_m} \le 2|B_{\rm P}^{\mu_i}(i)|_{r_m}$. If $s = s^{[r_m]}$ is not an original breakpoint in $B_{\rm P}^{\mu_i}(i)$, then the r_m^{th} subinterval can be combined with the r_{m-1}^{th} subinterval to calculate the total number of breakpoints. If s_{m-1}^* exists, then only (ii) can happen in the r_{m-1}^{th} subinterval with only one new breakpoint generated in the interval $[s_{m-1}^*, s^{[k+1]})$ according to our assumption that there is at least one new breakpoint generated or $s = s_{m-1}^*$ in $\mathcal{H}(i,s)$ in each subinterval. Thus, we have $|B(i)|_{r_m} + |B(i)|_{r_{m-1}} \le 2(|B_{\rm P}^{\mu_i}(i)|_{r_m} + |B_{\rm P}^{\mu_i}(i)|_{r_{m-1}} + |B_{\rm NP}(i)|_{r_m} + |B_{\rm NP}(i)|_{r_{m-1}})$ holds. If s_{m-1}^* does not exist, then either (19) or (20) holds for the r_{m-1}^{th} subinterval. Since there is only one new breakpoint generated in the r_m^{th} interval, we also have $|B(i)|_{r_m} + |B(i)|_{r_{m-1}} \le 2(|B_{\rm P}^{\mu_i}(i)|_{r_m} + |B_{\rm P}^{\mu_i}(i)|_{r_{m-1}} + |B_{\rm NP}(i)|_{r_m} + |B_{\rm NP}(i)|_{r_{m-1}})$ holds. Therefore, in general, we have (17) holds.

Case 2: $\mathcal{H}_{\rm NP}(i, s^{[k]}) \le \mathcal{H}_{\rm P}^{\tilde{\mu}}(i, s^{[k]})$. For this case, the breakpoint $s = s^{[k]}$ can be a breakpoint for $\mathcal{H}(i,s)$.

Scenario 1: There is only one piece of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in this interval. If there is no intercrossing between $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$, then we only need to consider the breakpoints generated by $\min\{\mathcal{H}_{\rm P}^{\mu_i}(i,s), \mathcal{H}_{\rm NP}(i,s)\}$. Based on (18), there is no "convex" breakpoints in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ below $\mathcal{H}_{\rm NP}(i,s)$. Therefore, there is one new breakpoint generated (i.e., $|B(i)|_k \le |B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k + 1$), or two new breakpoints generated with at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted (i.e., $|B(i)|_k \le |B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k + 2 - 1$). Thus, for both cases, we have $|B(i)|_k \le |B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k + 1 \le 2(|B_{\rm P}^{\mu_i}(i)|_k + |B_{\rm NP}(i)|_k)$.

If there is one intercrossing point generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in this interval, i.e., denoted as $s = s_1^*$, then we consider this interval (i.e., $[s^{[k]}, s^{[k+1]})$) with the next interval (i.e., $[s^{[k+1]}, s^{[k+2]})$) together. Note here if $s = s^{[k]}$ is the last breakpoint in $\mathcal{H}_{\rm NP}(i,s)$, then there is no original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ in the interval $[s^{[k]}, s^{[k+1]})$ with $s^{[k+1]} = +\infty$ and the corresponding linear piece of $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ is parallel to $\mathcal{H}_{\rm NP}(i,s)$. Therefore, at most one new breakpoint is generated and we have (17) holds. Now we discuss several cases and prove that

$$|B(i)|_{k'} \le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}), \tag{21}$$

where k' represents the union of k^{th} and $(k+1)^{th}$ intervals.

(1) There is no intercrossing point generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in the interval $[s^{[k+1]}, s^{[k+2]})$. For this case, there are two linear pieces in $[s^{[k]}, s_1^*)$ and $[s_1^*, s^{[k+2]})$ for $\min\{\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s), \mathcal{H}_{\rm NP}(i,s)\}$. Then it is easy to verify that $|B'(i)|_{s\in[s^{[k]},s_1^*)} \le |B_{\rm P}^{\mu_i}(i)|_{s\in[s^{[k]},s_1^*)} + 1$ and $|B'(i)|_{s\in[s_1^*,s^{[k+2]})} \le |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s^{[k+2]})} + 1$.

For the case that both $|B'(i)|_{s\in[s^{[k]},s_1^*)}$ and $|B'(i)|_{s\in[s_1^*,s^{[k+2]})} > 0$, we have

$$\begin{aligned} |B(i)|_{k'} &\le |B'(i)|_{s\in[s^{[k]},s_1^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s^{[k]},s_1^*)} + |B'(i)|_{s\in[s_1^*,s^{[k+2]})} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s^{[k+2]})} + 1 \\ &\le 2(|B_{\rm P}^{\mu_i}(i)|_{s\in[s^{[k]},s_1^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s^{[k+2]})}) + 3 \\ &\le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}) - 1, \end{aligned} \tag{22}$$

where "1" in the first inequality represents the breakpoint $s = s^{[k]}$ in $\mathcal{H}_{\rm NP}(i,s)$. Note here we did not count the breakpoint $s = s_1^*$. This lies in the fact that if both $|B'(i)|_{s\in[s^{[k]},s_1^*)}$

and $|B'(i)|_{s\in[s_1^*,s^{[k+2]})} > 0$, we have at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ during the k'^{th} interval or the breakpoint $s = s_1^*$ deleted.

For the case that either $|B'(i)|_{s\in[s^{[k]},s_1^*)}$ or $|B'(i)|_{s\in[s_1^*,s^{[k+2]})} = 0$, we have either only one new breakpoint generated or two new breakpoints generated with at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted. Thus, we also have $|B(i)|_{k'} \le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}) - 1$ holds. Therefore, in general, if there is no intercrossing point generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in the interval $[s^{[k+1]}, s^{[k+2]})$, then we have the following inequality holds:

$$|B(i)|_{k'} \le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}) - 1. \tag{23}$$

(2) There is an intercrossing point generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in the interval $[s^{[k+1]}, s^{[k+2]})$. For instance, denote it as $s = s_2^*$. For this case, there are three intervals: $[s^{[k]}, s_1^*)$, $[s_1^*, s_2^*)$ and $[s_2^*, s^{[k+2]})$.

(i) If there is no new breakpoint in the interval $[s_2^*, s^{[k+2]})$, then the problem is the same as above (1) except at most one additional breakpoint $s = s_2^*$. Then, we have that inequality (21) holds.

(ii) If there is no new breakpoint in the interval $[s^{[k]}, s_1^*)$, then we can follow the similar argument as in (1). For instance, if both $|B'(i)|_{s\in[s_1^*,s_2^*)}$ and $|B'(i)|_{s\in[s_2^*,s^{[k+2]})} > 0$, then we have

$$\begin{aligned} |B(i)|_{k'} &\le |B'(i)|_{s\in[s_1^*,s_2^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s_2^*)} + |B'(i)|_{s\in[s_2^*,s^{[k+2]})} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_2^*,s^{[k+2]})} + 2 \\ &\le 2(|B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s_2^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_2^*,s^{[k+2]})}) + 4 \\ &\le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}), \end{aligned}$$

where "2" in the first inequality represents the breakpoints $s = s^{[k]}$ in $\mathcal{H}_{\rm NP}(i,s)$ and $s = s_1^*$. The last inequality holds due to the fact that $|B_{\rm NP}(i)| \ge 2$. For the case that either $|B'(i)|_{s\in[s_2^*,s^{[k+2]})}$ or $|B'(i)|_{s\in[s_1^*,s_2^*)} = 0$, we have either only one new breakpoint generated or two new breakpoints generated with one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted. Thus, we also have $|B(i)|_{k'} \le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'})$ holds.

(iii) If there is no new breakpoint in the interval $[s_1^*, s_2^*)$, then either both breakpoints $s = s_1^*$ and $s = s_2^*$ are in $\mathcal{H}(i,s)$ or neither of them in $\mathcal{H}(i,s)$. If neither of them in $\mathcal{H}(i,s)$, then for each interval $[s^{[k]}, s_1^*)$ or $[s_2^*, s^{[k+2]})$, we have either only one new breakpoint generated or two new breakpoints generated with one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ deleted, then it is easy to see that (21) holds. If both of them in $\mathcal{H}(i,s)$, then at least one original breakpoint in $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ is deleted. We have at most one new breakpoint generated during each interval $[s^{[k]}, s_1^*)$ and $[s_2^*, s^{[k+2]})$. It is also easy to verify that (21) holds.

(iv) If there is at least one new breakpoint generated for each interval, then following the similar arguments as in (22), we have

$$\begin{aligned} |B(i)|_{k'} &\le |B'(i)|_{s\in[s^{[k]},s_1^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s^{[k]},s_1^*)} + |B'(i)|_{s\in[s_1^*,s_2^*)} \\ &\quad + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s_2^*)} + |B'(i)|_{s\in[s_2^*,s^{[k+2]})} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_2^*,s^{[k+2]})} + 1 \\ &\le 2(|B_{\rm P}^{\mu_i}(i)|_{s\in[s^{[k]},s_1^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_1^*,s_2^*)} + |B_{\rm P}^{\mu_i}(i)|_{s\in[s_2^*,s^{[k+2]})}) + 4 \\ &\le 2(|B_{\rm NP}(i)|_{k'} + |B_{\rm P}^{\mu_i}(i)|_{k'}), \end{aligned}$$

where "1" in the first inequality represents the breakpoint $s = s^{[k]}$ in $\mathcal{H}_{\rm NP}(i,s)$. The last inequality holds due to the fact that $|B_{\rm NP}(i)| \ge 2$.

Scenario 2: There are multiple pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in this interval as shown in Figure 5. Then, there may be multiple intercrossing points generated by $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$ in the interval $[s^{[k]}, s^{[k+2]})$.

As described in Scenario 2 of Case 1, we can assume that all pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ are below or intercrossing with $\mathcal{H}_{\rm NP}(i,s)$ as shown in Figure 5. We also notice that at each breakpoint $s = s^{[r_j]}, 2 \le j \le m$, the production value function $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ reaches its capacity. If not, for the case that the slope for $\mathcal{H}_{\rm NP}(i,s)$ at this breakpoint is less than $-\alpha_i$ as shown in the interval $[s^{[k+1]}, s^{[k+2]})$ in Figure 5, then we can follow the same argument as in Case 1 to find the

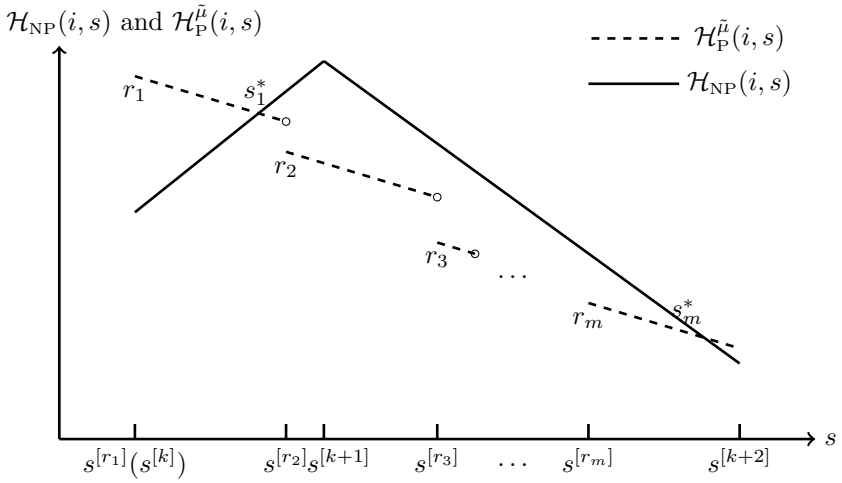

Figure 5: Multiple pieces of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ in the k^{th} and $k+1^{th}$ intervals of $\mathcal{H}_{\rm NP}(i,s)$

contradiction. For the case that the slope for $\mathcal{H}_{\rm NP}(i,s)$ at this breakpoint is larger than $-\alpha_i$ as shown in the interval $[s^{[k]}, s^{[k+1]})$ in Figure 5, if production does not reach its capacity, then we should have $s^{[r_j]} = d_{ik} - \sum_{j\in S}\mu_j$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \setminus \mathcal{P}(i)$ and accordingly $\lim_{s\to s^{[r_j]-}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \geq \lim_{s\to s^{[r_j]-}} \mathcal{H}_{\rm NP}(i,s)$ based on Lemma 4.1. Thus the whole piece of the production line in the r_{j-1}^{th} subinterval will be above $\mathcal{H}_{\rm NP}(i,s)$ and it does not need to be considered. Contradiction. Based on the similar argument, we have $\lim_{s\to s^{[r_j]-}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s) \geq \lim_{s\to s^{[r_j]+}} \mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$ for $2 \leq j \leq m$.

Now consider the total number of breakpoints. For the first piece (i.e., r_1 piece), the conclusion follows directly from our Scenario 1 in this case. For the pieces corresponding to $r_j, 2 \leq j \leq m$, we can use the similar argument as in the Scenario 2 in Case 1. The only difference is for (iv) the last piece (i.e., the r_m^{th} piece) in which $s = s^{[r_m]}$ is not an original breakpoint in $B_{\rm P}^{\mu_i}(i)$ and there is a new breakpoint generated in the interval $[s_m^*, s^{[k+2]})$. For this case, the r_m^{th} subinterval can be combined with r_{m-1}^{th} subinterval. If $m-1 \geq 2$, then the analysis is the same as in Scenario 2 in Case 1. Otherwise, if $m-1 = 1$, i.e., the first piece, then there are only two possible scenarios that can happen for the first piece of $\mathcal{H}_{\rm P}^{\tilde{\mu}}(i,s)$: 1) no new breakpoint $s = s_2^*$ generated in the interval $[s^{[k+1]}, s^{[k+2]})$ and thus (23) holds, and 2) $s = s_2^*$ exists and there is no new breakpoints generated in the interval $[s_2^*, s^{[r_m]})$. In 2), it is easy to verify that $s = s_2^*$ is not a breakpoint in $\mathcal{H}(i,s)$ and thus as indicated in (2)(i) in Scenario 1, we also have $|B(i)|_{r_1} \leq 2(|B_{\rm NP}(i)|_{r_1} + |B_{\rm P}^{\mu_i}(i)|_{r_1}) - 1$ holds. Since there is only one new breakpoint generated in the interval $[s_m^*, s^{[k+2]})$, we have $|B(i)|_{r_m} + |B(i)|_{r_{m-1}} \leq 2(|B_{\rm P}^{\mu_i}(i)|_{r_m} + |B_{\rm P}^{\mu_i}(i)|_{r_{m-1}} + |B_{\rm NP}(i)|_{r_m} + |B_{\rm NP}(i)|_{r_{m-1}})$ holds.

Therefore, the conclusion holds. Figure 6 shows an example that the bound of the breakpoints $|B(i)| \leq 4|B_{\rm NP}(i)|$ can be asymptotically tight. □

PROPOSITION 4.4 *The total number of breakpoints $|B(i)|$ is bounded by $\mathcal{O}(|\mathcal{V}(i)|^3)$.*

PROOF. For each leaf node i at time period T, $|B(i)| \leq 4 = 4|\mathcal{V}(i)|$ as shown in Figure 3. Then we prove by induction that the number of breakpoints for each node $i \in \mathcal{V}$

$$|B(i)| \leq 4^{T-t(i)+1}|\mathcal{V}(i)|.$$

Assume the claim is true for each node $\ell \in \mathcal{C}(i)$ at time period $t(\ell)$. That is, $|B(\ell)| \leq 4^{T-t(\ell)+1}|\mathcal{V}(\ell)|$. Then, based on Lemma 4.4, we have

$$|B(i)| \leq 4\sum_{\ell\in\mathcal{C}(i)} |B(\ell)| \leq 4\sum_{\ell\in\mathcal{C}(i)} 4^{T-t(\ell)+1}|\mathcal{V}(\ell)| = 4^{T-t(i)+1}\sum_{\ell\in\mathcal{C}(i)} |\mathcal{V}(\ell)| \leq 4^{T-t(i)+1}|\mathcal{V}(i)|.$$

Since every non-leaf node has at least two children, we have

$$|\mathcal{V}(i)| \geq 1 + 2^1 + \ldots + 2^{T-t(i)} = 2^{T-t(i)+1} - 1,$$

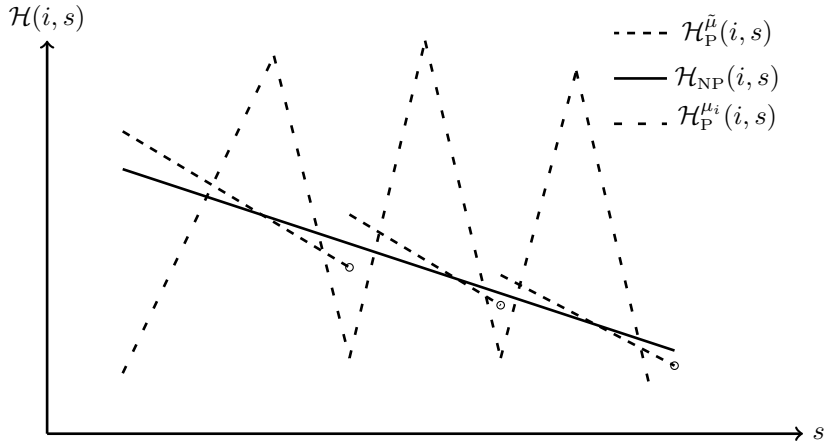

Figure 6: An example

which implies that the number of breakpoints

$$|B(i)| \le 4^{T-t(i)+1}|\mathcal{V}(i)| \le (|\mathcal{V}(i)|+1)^2|\mathcal{V}(i)|.$$

Therefore, the conclusion holds. □

According to the above proof, it is easy to see that

COROLLARY 4.2 *For the balanced tree case (i.e., $|\mathcal{C}(i)| = \mathcal{C}$ for each node $i \in \mathcal{V} \setminus \mathcal{L}$), the total number of breakpoints in $|B(i)|$ is bounded by $\mathcal{O}(|\mathcal{V}(i)|^{\log_{\mathcal{C}}^4 +1})$.*

THEOREM 4.1 *If $\mathcal{C}(i) \ge 2$ for each node $i \in \mathcal{V} \setminus \mathcal{L}$, then the optimal value function of SCLS with backlogging can be obtained in $\mathcal{O}(n^4)$ time.*

PROOF. For each node $i \in \mathcal{V}$, the value function $\mathcal{H}(i,s)$ is piecewise linear and continuous. Therefore, it can be stored in terms of breakpoint values, evaluations of breakpoints, and the right slope of each breakpoint. We construct the value functions $\mathcal{H}(i,s)$ by induction starting from each leaf node at time period T. Similarly, we use $\theta(i,s)$ to represent $\sum_{\ell\in\mathcal{C}(i)} \mathcal{H}(\ell,s)$.

The value function of each leaf node is shown in Figure 3 and it can be stored in terms of two or four breakpoints (we list the four breakpoint case as follows):

(1) four breakpoints: $s^{[1]} = -\infty$, $s^{[2]} = d_i - \mu_i$, $s^{[3]} = \underline{d_i}$, and $s^{[4]} = d_i$.

(2) evaluations at these four breakpoints: $\mathcal{H}(i,s^{[1]}) = +\infty$, $\mathcal{H}(i,s^{[2]}) = \alpha_i\mu_i + \beta_i$, $\mathcal{H}(i,s^{[3]}) = \alpha_i(d_i - \underline{d_i}) + \beta_i$ and $\mathcal{H}(i,s^{[4]}) = 0$.

(3) the right slope of each breakpoint: $r(i,s^{[1]}) = -b_i$, $r(i,s^{[2]}) = -\alpha_i$, $r(i,s^{[3]}) = -b_i$ and $r(i,s^{[4]}) = h_i$.

Then, the breakpoints in $\theta(i^-,s)$ will be generated or updated accordingly. We assume all breakpoints are stored in a non-decreasing order. To store the value function $\mathcal{H}(i,s)$ for each node $i \in \mathcal{V}$, we need to perform the following several steps:

Step 1: Calculate and store $\sum_{\ell\in\mathcal{C}(i)} \mathcal{H}(\ell,s)$: Since $\theta(i,s)$ is obtained when we finish calculating all children of node i, this step can be completed in $\mathcal{O}(n^3)$ time since the number of breakpoints for $\theta(i,s)$ is bounded by $\mathcal{O}(n^3)$ as shown in Proposition 4.4.

Step 2: Obtain and store $\mathcal{H}_{\rm NP}(i,s)$: From (11), this step can be obtained by moving $\sum_{\ell\in\mathcal{C}(i)} \mathcal{H}(\ell,s)$ to the right by d_i units plus the piecewise linear function $\max\{h_i(s-d_i), -b_i(s-d_i)\}$. It is easy to see this step can be completed in $\mathcal{O}(n^3)$ time since the number of breakpoints for node i is bounded by $\mathcal{O}(n^3)$ as shown in Proposition 4.4.

Step 3: Obtain and store $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$: From (12), we can observe that $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ can be obtained by moving $\mathcal{H}_{\rm NP}(i,s)$ to the left by μ_i plus a constant number $\beta_i+\alpha_i\mu_i$. This step can therefore be completed in $\mathcal{O}(n^3)$ time since the number of breakpoints for node i is bounded by $\mathcal{O}(n^3)$ as shown in Proposition 4.4.

Step 4: Calculate and store $\mathcal{H}'(i,s)=\min\{\mathcal{H}_{\rm NP}(i,s),\mathcal{H}_{\rm P}^{\mu_i}(i,s)\}$: Between any two consecutive breakpoints in $B_{\rm NP}(i)\cup B_{\rm P}^{\mu_i}(i)$, we compare the corresponding two pieces in $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ and obtain the minimum of the two functions based on their slopes and evaluations of breakpoints. Then we store the evaluations of the new breakpoints and the right slopes of the breakpoints. This step can be completed in $\mathcal{O}(n^3)$ time since the number of breakpoints in $B_{\rm NP}(i)\cup B_{\rm P}^{\mu_i}(i)$ is bounded by $\mathcal{O}(n^3)$ as shown in Proposition 4.4.

Step 5: Among the breakpoints generated by $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$, calculate $\phi(k,S)=$

$$\beta_i+\alpha_i(d_{ik}-\sum_{j\in S}\mu_j)+\max\left\{h_i(d_{ik}-\sum_{j\in S}\mu_j-d_i),-b_i(d_{ik}-\sum_{j\in S}\mu_j-d_i)\right\}+\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,d_{ik}-\sum_{j\in S}\mu_j-d_i)$$

for each node $k\in\mathcal{V}(i)$ and $S\subseteq\mathcal{P}(k)\setminus\mathcal{P}(i)$. Note here, these breakpoints can be labeled every time when generate new breakpoints for $\mathcal{H}_{\rm NP}(i,s)$ and $\mathcal{H}_{\rm P}^{\mu_i}(i,s)$ and stored in a separate list. For each combination of a node $k\in\mathcal{V}(i)$ and a set $S\subseteq\mathcal{P}(k)\setminus\mathcal{P}(i)$, based on the result in Step 1, the value of the function $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,d_{ik}-\sum_{j\in S}\mu_j-d_i)$ can be obtained by binary search in $\mathcal{O}(\log n^3)=\mathcal{O}(\log n)$ time. It can also be observed that there are at most $\mathcal{O}(n^2)$ possible combinations of k and $S\subseteq\mathcal{P}(k)\setminus\mathcal{P}(i)$ because corresponding to each node $k\in\mathcal{V}(i)$, the number of possible sets S is bounded by $\mathcal{O}(|\mathcal{V}(i)|)$. Therefore, this entire step can be completed in $\mathcal{O}(n^2\log n)$ time.

Step 6: Calculate and store $\mathcal{H}(i,s)=\min\{\mathcal{H}_{\rm P}(i,s),\mathcal{H}'(i,s)\}$: Sort $\phi(k,S)$ in a non-decreasing order and build a corresponding list ξ_1, which takes $\mathcal{O}(n^2\log n)$ time. We also build a list ξ_2 to store the start and end breakpoints and their evaluations for all linear pieces of $\mathcal{H}_{\rm P}(i,s)$. In ξ_2, all these linear pieces are stored according to increasing sequence of their start breakpoint inventory values. Initially $\xi_2=\emptyset$.

Starting from the first one in ξ_1, for each pair k and S, we generate the value functions $\mathcal{H}_{\rm P}(i,s)$ and $\mathcal{H}(i,s)$ for the interval $[d_{ik}-\sum_{j\in S}\mu_j-\mu_i,d_{ik}-\sum_{j\in S}\mu_j)$ in the following steps:

(1) Use binary search to find a linear piece in ξ_2 whose end breakpoint is the largest and in the interval $[d_{ik}-\sum_{j\in S}\mu_j-\mu_i,d_{ik}-\sum_{j\in S}\mu_j)$. Denote the end breakpoint of the linear piece as $s=d_{ks}$;

(2) Use binary search to find a linear piece in ξ_2 whose start breakpoint is the smallest and in the interval $[d_{ik}-\sum_{j\in S}\mu_j-\mu_i,d_{ik}-\sum_{j\in S}\mu_j)$. Denote the start breakpoint of the linear piece as $s=e_{ks}$;

(3) Due to fixed interval length μ_i for each pair, the value of $\mathcal{H}_{\rm P}(i,s)$ in the interval $[d_{ks},e_{ks})$ should be equal to $\phi(k,S)-\alpha_i s$ and the corresponding values for other parts of the interval $[d_{ik}-\sum_{j\in S}\mu_j-\mu_i,d_{ik}-\sum_{j\in S}\mu_j)$ are decided by other pairs;

(4) Obtain $\mathcal{H}(i,s)=\min\{\mathcal{H}_{\rm P}(i,s),\mathcal{H}'(i,s)\}$ for the interval $[d_{ks},e_{ks})$ and insert the linear piece of $\mathcal{H}_{\rm P}(i,s)$ for the interval $[d_{ks},e_{ks})$ to the right position in ξ_2 to maintain the increasing sequence.

Note that there are at most $\mathcal{O}(n^2)$ pairs, the binary search takes $\mathcal{O}(\log n)$ time, and the number of breakpoints in $\mathcal{H}'(i,s)$ is bounded by $\mathcal{O}(n^3)$. Thus, the maximum computational complexity for this step is decided by total operations in (4) for all pairs, which is bounded by $\mathcal{O}(n^3)$. This step can be completed in $\mathcal{O}(n^3)$ time.

Step 7: Update $\theta(i^-,s)$ by adding $\mathcal{H}(i,s)$: This step can be completed in $\mathcal{O}(n^3)$ time since the number of breakpoints is bounded by $\mathcal{O}(n^3)$ as shown in Proposition 4.4.

The maximum computational complexity for each step is $\mathcal{O}(n^3)$. Since the above operations are required for each node $i\in\mathcal{V}$, the optimal value function for SCLS with backlogging can be obtained and stored in $\mathcal{O}(n^4)$ time. □

Based on Corollary 4.2, it is easy to obtain that

COROLLARY 4.3 *The optimal value function of SCLS with backlogging for the balanced scenario tree case can be obtained in* $\mathcal{O}(n^{\log_{\mathrm{C}}^4+2})$ *time.*

5. Stochastic Constant Capacitated Lot-sizing Problem with Backlogging It is shown in the previous section that the optimal value function of SCLS with backlogging can be obtained in polynomial time when each non-leaf node contains at least two children. In this section, we investigate the computational complexity to obtain an optimal solution for stochastic constant capacitated lot-sizing (SCCLS) problem with backlogging, regardless of the scenario tree structure. That is, we derive a polynomial time algorithm to obtain an optimal solution for this problem under a general tree structure (i.e., $\mathcal{C}(i) \geq 1$ for each node $i \in \mathcal{V} \setminus \mathcal{L}$). For this case, since the capacity at each node is a constant, Proposition 4.1 can be simplified as follows:

PROPOSITION 5.1 *For any instance of SCLS with backlogging in which capacity is a constant* μ*, there exists an optimal solution* (x^*, y^*, s^*) *such that for each node* $i \in \mathcal{V}$*,*

$$\text{if } 0 < x_i^* < \mu, \text{ then } x_i^* + s_{i^-}^* = d_{ik} - m\mu \text{ for some node } k \in \mathcal{V}(i) \text{ and an integer number } m\text{: } 0 \leq m < T \text{ and } x_j^* = 0 \text{ or } \mu \text{ for each } j \in \mathcal{P}(k) \setminus \mathcal{P}(i).$$

Based on this Proposition, The value function of the problem can be simplified as follows:

- Non-production:

$$\mathcal{H}_{\mathrm{NP}}(i,s) = \max\{h_i(s-d_i), -b_i(s-d_i)\} + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s-d_i). \tag{24}$$

- Production at capacity:

$$\mathcal{H}_{\mathrm{P}}^{\mu}(i,s) = \beta_i + \alpha_i\mu + \max\{h_i(\mu+s-d_i), -b_i(\mu+s-d_i)\} + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, \mu+s-d_i). \tag{25}$$

- Production less than capacity:

$$\mathcal{H}_{\mathrm{P}}^{\tilde{\mu}}(i,s) = \min_{k \in \mathcal{V}(i): d_{ik}-(m+1)\mu \leq s < d_{ik}-m\mu} \mathcal{H}_{\mathrm{P}}^{k,m}(i,s), \tag{26}$$

where

$$\begin{aligned} \mathcal{H}_{\mathrm{P}}^{k,m}(i,s) &= \beta_i + \alpha_i(d_{ik} - m\mu - s) \\ &+ \max\{h_i(d_{ik} - m\mu - d_i), -b_i(d_{ik} - m\mu - d_i)\} \\ &+ \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - m\mu - d_i). \end{aligned} \tag{27}$$

Without loss of generality, we assume zero initial inventory and $d_{10} = 0$ where node "0" is a dummy node. We can obtain the following proposition.

PROPOSITION 5.2 *For SCCLS with backlogging, corresponding to each node* $i \in \mathcal{V}$*, we have* $s_i = d_{1k} - d_{1i} + m\mu$ *with* $-T \leq m \leq t(i)$ *for some node* $k \in \mathcal{V} \cup \{0\}$ *in an optimal solution.*

PROOF. With zero initial inventory, according to Proposition 5.1, we have $x_1 = d_{1k} - m\mu$ and $s_1 = d_{1k} - m\mu - d_1$ for some node $k \in \mathcal{V} \cup \{0\}$ and an integer member $m : 0 \leq m < T$ in an optimal solution, or $x_1 = \mu$ and $s_1 = \mu - d_1$. Thus, the conclusion holds for the root node.

Assume the conclusion holds for a node $i \in \mathcal{V}$ at time period $t(i)$. That is, $s_i = d_{1k} - d_{1i} + m\mu$ with $-T \leq m \leq t(i)$ for some node $k \in \mathcal{V} \cup \{0\}$ in an optimal solution. Then, for each node $\ell \in \mathcal{C}(i)$, based on Proposition 5.1, we verify three possible cases for production at node ℓ as follows:

Case 1: $x_\ell = 0$. For this case, $s_\ell = s_i - d_\ell = d_{1k} - d_{1i} + m\mu - d_\ell = d_{1k} - d_{1\ell} + m\mu$ for some node $k \in \mathcal{V} \cup \{0\}$ with $-T \leq m \leq t(i) < t(\ell)$.

Case 2: $x_\ell = \mu$. For this case, $s_\ell = s_i + \mu - d_\ell = d_{1k} - d_{1i} + (m+1)\mu - d_\ell = d_{1k} - d_{1\ell} + (m+1)\mu$ for some node $k \in \mathcal{V} \cup \{0\}$ with $-T < -T + 1 \le m + 1 \le t(i) + 1 = t(\ell)$.

Case 3: $s_i + x_\ell = d_{\ell k} - m\mu$ for some node $k \in \mathcal{V}(\ell)$ and an integer $m : 0 \le m < T$. Then, $s_\ell = s_i + x_\ell - d_\ell = d_{\ell k} - m\mu - d_\ell = d_{1k} - d_{1\ell} - m\mu$ for some $-T < -m \le 0 < t(\ell)$.

All above three cases indicate that the induction step holds and therefore, our claim is true. □

Proposition 5.2 will simplify the algorithm. Based on this proposition, we only need to store the breakpoints for node i that are in the form $s = s_{i^-} = d_{1k} - d_{1i^-} + m\mu$ with $-T \le m \le t(i^-)$ for all $k \in \mathcal{V} \cup \{0\}$. For the general capacitated case, based on (11), (12) and (13), the breakpoints for the value function $\mathcal{H}(i, s)$ are obtained by moving the breakpoints for the value function $\mathcal{H}(\ell, s)$ for all $\ell \in \mathcal{C}(i)$ to the right by d_i, or by $d_i - \mu_i$, plus new breakpoints generated by minimizing three piecewise linear functions. Thus, for the constant capacitated case, in order to store the breakpoints $s = d_{1k} - d_{1i^-} + m\mu$ with $-T \le m \le t(i^-)$ for all $k \in \mathcal{V} \cup \{0\}$, in the calculation of the value functions $\mathcal{H}(\ell, s)$ for all $\ell \in \mathcal{C}(i)$, we only need to store the breakpoints $s = d_{1k} - d_{1i} + m\mu = d_{1k} - d_{1\ell^-} + m\mu$ for all $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(\ell^-)$. Therefore, in the algorithm described in Theorem 4.1 to calculate the value function $\mathcal{H}(i, s)$ for each node $i \in \mathcal{V}$ backwards starting from leaf nodes, we store the breakpoints $s = d_{1k} - d_{1i^-} + m\mu$ and their evaluations for all $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(i^-)$. There are at most $\mathcal{O}(nT)$ breakpoints needed to be stored corresponding to each value function $\mathcal{H}(i, s)$.

Similar as SULS with backlogging, the scenario tree structure can help speed up the algorithm. For the initial step, we set relationship indicator $\delta(i, r)$ for each node $i \in \mathcal{V}$ and a label r. The label r is generated by each combination of a node $k \in \mathcal{V}(i)$ and an integer number m: $-T \le m \le t(i^-)$. We use $\theta(i, s)$ to store $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ for which $s = d_{1k} - d_{1\ell^-} + m\mu$ for all $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(\ell^-)$ and initialize them to zero. Note here this initial step can be completed in $\mathcal{O}(n^2 T)$ time.

Then, we sort the breakpoints for the root node $s_0 = d_{1k} + m\mu$ for each node $k \in \mathcal{V} \cup \{0\}$ and $-T < m \le T$ in a non-decreasing sequence. This can be completed in $\mathcal{O}(nT \log n)$ time. Note here these breakpoints are enough to serve as the breakpoints of the value function $\mathcal{H}(i, s)$ for each node $i \in \mathcal{V}$. For instance, corresponding to each node $i \in \mathcal{V}$, we set $s = 0$ at the breakpoint $s_0 = d_{1i^-}$. Thus, the breakpoints contained in $\mathcal{H}(i, s)$, $s = d_{1k} - d_{1i^-} + m\mu$ for each node $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(i^-)$, can be stored at the breakpoints $s_0 = d_{1k} + m\mu$ for each node $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(i^-)$. Finally, duplicated breakpoints are considered separately.

By taking advantages of both scenario tree structure and the simplified value functions (24) to (27), the optimal solution for a general scenario tree structure SCCLS with backlogging can be obtained by induction starting from the last time period T in the following approach.

For each node i in time period T, the value function is

$$\mathcal{H}(i, s) = \begin{cases} \beta_i + \alpha_i \mu + b_i(d_i - \mu - s) & \text{if } s < d_i - \mu, \\ \alpha_i(d_i - s) + \beta_i & \text{if } d_i - \mu \le s < \underline{d}_i, \\ b_i(d_i - s) & \text{if } \underline{d}_i \le s < d_i, \\ h_i(s - d_i) & \text{if } s \ge d_i, \end{cases}$$

where

$$\underline{d}_i = d_i - \frac{\beta_i}{b_i - \alpha_i}.$$

Since this function is continuous, the breakpoints $s = d_{1k} - d_{1i^-} + m\mu$ for all $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(i^-)$ of $\mathcal{H}(i, s)$ can be achieved in $\mathcal{O}(nT)$ time since the number of breakpoints to be stored is bounded by $\mathcal{O}(nT)$. The value function $\theta(i^-, s)$ is updated in a way that the function value corresponding to each breakpoint $s = d_{1k} - d_{1i^-} + m\mu$ for all $k \in \mathcal{V} \cup \{0\}$ and $-T \le m \le t(i^-)$ is increased by $\mathcal{H}(i, s)$. This procedure can also be completed in $\mathcal{O}(nT)$ time.

For each induction step for a node i, we execute the following steps:

Step 1: Calculate non-production value function $\mathcal{H}_{\text{NP}}(i, s)$: Since $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s - d_i)$ has been calculated and stored in $\theta(i, s - d_i)$, according to formulation (24), $\mathcal{H}_{\text{NP}}(i, s)$ can be obtained in $\mathcal{O}(nT)$ time since the number of breakpoints is bounded by $\mathcal{O}(nT)$. Note here the breakpoint $s = d_i$ for $\mathcal{H}_{\text{NP}}(i, s)$ is in our predetermined breakpoint list;

Step 2: Calculate production at capacity value function $\mathcal{H}_{\rm P}^{\mu}(i,s)$: Similar as in Step 1, according to (25), this step can be completed in $\mathcal{O}(nT)$ time since the number of breakpoints is bounded by $\mathcal{O}(nT)$;

Step 3: Calculate and store

$$\phi(k,m)=\beta_i+\alpha_i(d_{ik}-m\mu)+\max\{h_i(d_{ik}-m\mu-d_i),-b_i(d_{ik}-m\mu-d_i)\}+\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,d_{ik}-m\mu-d_i)$$

for each combination of $k\in\mathcal{V}$ and $m: 0\le m<T$. Note here, the breakpoint $s=d_{ik}-m\mu-d_i=d_{1k}-m\mu-d_{1\ell^-}$ is in the set of breakpoints for $\theta(i,s-d_i)$. For each combination of node $k\in\mathcal{V}$ and $m: 0\le m<T$, the value of the function $\sum_{\ell\in\mathcal{C}(i)}\mathcal{H}(\ell,d_{ik}-m\mu-d_i)$ can be obtained by binary search in $\mathcal{O}(\log n)$ time, since the number of breakpoints is bounded by $\mathcal{O}(nT)$ and $T\le n$. It can also be observed that there are $\mathcal{O}(nT)$ possible combinations of k and m. Therefore, this entire step can be completed in $\mathcal{O}(nT\log n)$ time;

Step 4: Calculate and store $\mathcal{H}_{\rm P}(i,s)$: Since $\mathcal{H}_{\rm P}(i,s)$ is piecewise linear, right continuous, and the slope for each piece is fixed (i.e., $-\alpha_i$), for each breakpoint, the qualified pair (k,m) (i.e., this breakpoint is within the interval $[d_{ik}-(m+1)\mu, d_{ik}-m\mu)$) that has the smallest $\phi(k,m)$ value leads to the corresponding $\mathcal{H}_{\rm P}(i,s)$ value for this breakpoint. To obtain $\mathcal{H}_{\rm P}(i,s)$ for each breakpoint, we can start from the breakpoint with the largest inventory value backwards to the breakpoint with the smallest inventory value. Corresponding to each breakpoint, we maintain and update a list of active pairs (k,m) according to a non-decreasing sequence of $\phi(k,m)$ values. Because each piece of $\mathcal{H}_{\rm P}(i,s)$ has the same width μ, corresponding to the breakpoint with the largest inventory value that is covered by $\mathcal{H}_{\rm P}(i,s)$, there exists only one pair (k,m) in the form as shown in (26) such that this breakpoint is within the corresponding interval. Therefore, this pair is set as both the head and the tail of the list. The value $\phi(k^*,m^*)$ and the corresponding production value function $\mathcal{H}_{\rm P}(i,s)=\phi(k^*,m^*)-\alpha_i s$ are stored, where (k^*,m^*) is denoted as the pair serving as the head of the list. Starting from this breakpoint backwards, for each breakpoint $s=d_{ik}-m\mu, 0\le m<T$, we perform the following steps:

(1) Insert the pair (k,m) into the active pair list by binary search to maintain the non-decreasing order of $\phi(k,m)$ values in the list. We also set this pair as the end of the list, because it can be observed that the pairs in the list after this pair are not needed to be considered. If $\phi(k,m)$ is smaller than the value for the current head of the list, then the pair (k,m) serves as both the head and the tail of the active list and it becomes the only pair in the active list again. This step takes $\mathcal{O}(\log n)$ time;

(2) Check if the current breakpoint value $s=d_{ik}-m\mu$ is less than $d_{ik^*}-(m^*+1)\mu$. If so, then the current head is out of range since it reaches the capacity and the next one in the active list will serve as the head. Note here duplicated breakpoints are considered separately and width for each piece of the production line is the same (i.e., μ). Thus, the next pair in the active list must be within the range and can serve as the head if it exists. This step takes $\mathcal{O}(1)$ time;

(3) Calculate and store the corresponding production value $\mathcal{H}_{\rm P}(i,s)=\phi(k^*,m^*)-\alpha_i s$. This step takes $\mathcal{O}(1)$ time.

Since there are at most $\mathcal{O}(nT)$ breakpoints, this procedure can be completed in $\mathcal{O}(nT\log n)$ time. After obtaining $\mathcal{H}_{\rm P}(i,s)$, the breakpoints $s=d_{1k}-d_{1i^-}+m\mu$ for $\mathcal{H}_{\rm P}(i,s)$ with $-T\le m\le t(i^-)$ for all $k\in\mathcal{V}\cup\{0\}$ can be calculated and stored in $\mathcal{O}(nT)$ time. Thus, this step takes $\mathcal{O}(nT\log n)$ time;

Step 5: Calculate and store $\mathcal{H}(i,s)$: For each breakpoint, take the minimum of $\mathcal{H}_{\rm P}(i,s)$, $\mathcal{H}_{\rm P}^{\mu}(i,s)$ and $\mathcal{H}_{\rm NP}(i,s)$. This step can be completed in $\mathcal{O}(nT)$ time since the number of breakpoints is bounded by $\mathcal{O}(nT)$;

Step 6: Update $\theta(i^-,s)$: Corresponding to each inventory breakpoint $s=d_{1k}-d_{1i^-}+m\mu$ for all $k\in\mathcal{V}\cup\{0\}$ and $-T\le m\le t(i^-)$, increase the value function $\theta(i^-,s)$ by $\mathcal{H}(i,s)$. This step can be completed in $\mathcal{O}(nT)$ time since the number of breakpoints is bounded by $\mathcal{O}(nT)$.

Therefore, the total amount of work required at each node is bounded by $\mathcal{O}(nT\log n)$. Since the above operations are required for each node $i\in\mathcal{V}$, the following conclusion holds.

THEOREM 5.1 *The general stochastic constant capacitated lot-sizing problem with backlogging can be solved in $\mathcal{O}(n^2T\log n)$ time, regardless of the scenario tree structure.*

The deterministic constant capacitated lot-sizing problem is the special case of our problem with only one scenario. In this case, the computational complexity in our approach is $\mathcal{O}(T^3\log T)$, which is between the complexity $\mathcal{O}(T^3)$ developed by van Hoesel and Wagelmans [30] in which backlogging is not allowed and $\mathcal{O}(T^4)$ developed by Florian and Klein [14]. Therefore, our approach also provides the best algorithm for the deterministic constant capacitated lot-sizing problem with backlogging.

Similarly as SULS with backlogging, the general SCCLS with backlogging problem for which the initial inventory level is unknown and is itself a decision variable can be transformed into another general SCCLS with backlogging problem with zero initial inventory by adding a dummy root node 0 as the parent node of node 1 with zero production, setup, and inventory costs as well as zero demand.

6. Conclusions In this paper, we studied the computational complexity for several classes of stochastic lot-sizing problems and especially, we developed a general dynamic programming framework to solve this type of problems. We first developed an algorithm that runs in $\mathcal{O}(n^2)$ time for SULS with backlogging under any general scenario tree structure. This model is more general compared to SULS studied in Guan and Miller [16] in which backlogging is not allowed. The computational complexity is also improved by a magnitude of $\mathcal{O}(\log n)$ by our new approach.

Then, we studied the general dynamic programming framework for SCLS with backlogging. We showed that the optimal value function is piecewise linear and continuous. For the case that each non-leaf node contains at least two children, based on the production path property, our dynamic programming framework provides an $\mathcal{O}(n^4)$ time algorithm to fully describe the optimal value function. The fact that SCLS with backlogging is polynomially solvable is initially surprising, given that the deterministic problem with varying capacities is $\mathcal{NP}$-hard. But it is reasonable due to the special structure of the scenario tree and the relationship of this structure to both the amount of information required to define the problem and the dynamic programming algorithms developed.

Finally, we studied SCCLS with backlogging. We showed that our algorithm runs in $\mathcal{O}(n^2T\log n)$ time for the problem under any general scenario tree structure. Note here that the deterministic constant capacitated lot-sizing problem is a a special case of our problem with only one scenario (i.e., each non-leaf node contains only one child). For this special case, our algorithm runs in $\mathcal{O}(T^3\log T)$ time and it is more efficient than the one developed by Florian and Klein [14], which runs in $\mathcal{O}(T^4)$ time. Although an $\mathcal{O}(T^3)$ algorithm for the constant capacitated lot-sizing problem was developed by van Hoesel and Wagelmans [30], backlogging is not allowed for their case.

In general, by taking advantage of the scenario tree structure, our dynamic programming framework can provide efficient algorithms for stochastic lot-sizing problems with backlogging. Our approach generalizes the analysis of the traditional deterministic lot-sizing problems into their multi-stage stochastic scenario tree settings. Most of these problems are polynomial time solvable in terms of input size (i.e., the number of nodes in the tree). Our approach also suggests that, under certain conditions, some problems, which are $\mathcal{NP}$-hard for their deterministic counterparts, are polynomial time solvable in terms of input size when they are generalized into their multi-stage stochastic scenario tree settings.

Polynomial time algorithm developments for stochastic lot-sizing problems indicate a possibility to discover strong valid inequalities to describe convex hull or linear programming formulations to provide integral solutions for the problems. Currently we are further studying the polyhedral aspects of stochastic lot-sizing problems based on the previous work by Guan et al. [15] for SULS and by Di Summa and Wolsey [11] for SCCLS. We are also studying on integrating polyhedral studies with decomposition algorithms, such as those studied by Carøe [10] and Sen and Sherali [27], among others, to solve large size stochastic integer programming problems.

Acknowledgments. The author expresses thanks to Andrew Miller for his helpful comments on an early version of this paper. This research is supported in part by National Science Foundation under Awards CMMI-0700868 and CMMI-0748204.

References

[1] A. Aggarwal and J. K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41:549–571, 1993.

[2] S. Ahmed and N. V. Sahinidis. An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research*, 51:461–471, 2003.

[3] A. Atamtürk and S. Kücükyavuz. An $O(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs. *Operations Research Letters*, 36:297–299, 2008.

[4] I. Bárány, T. van Roy, and L. A. Wolsey. Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30:1255–1262, 1984.

[5] I. Bárány, T. van Roy, and L. A. Wolsey. Uncapacitated lot-sizing: The convex hull of solutions. *Mathematical Programming Study*, 22:32–43, 1984.

[6] G. Belvaux and L. A. Wolsey. *bc − prod*: a specialized branch–and–cut system for lot-sizing problems. *Management Science*, 46:724–738, 2000.

[7] G. Belvaux and L. A. Wolsey. Modelling practical lot–sizing problems as mixed integer programs. *Management Science*, 47:993–1007, 2001.

[8] P. Beraldi and A. Ruszczyński. A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, 17:359–382, 2002.

[9] G. R. Bitran and H. H. Yanasse. Computational-complexity of the capacitated lot size problem. *Management Science*, 28:1174–1186, 1982.

[10] C. C. Carøe. *Decomposition in stochastic integer programming*. PhD thesis, University of Copenhagen, 1998.

[11] M. Di Summa and L. A. Wolsey. Lot-sizing on a tree. *Operations Research Letters*, 2008. to appear.

[12] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ time. *Management Science*, 37:909–925, 1991.

[13] A. Federgruen and M. Tzur. The dynamic lot-sizing model with backlogging-a simple $\mathcal{O}(n \log n)$ algorithm and minimal forecast horizon procedure. *Naval Research Logistics*, 40:459–478, 1993.

[14] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18:12–20, 1971.

[15] Y. Guan, S. Ahmed, G. L. Nemhauser, and A. J. Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105:55–84, 2006.

[16] Y. Guan and A. J. Miller. Polynomial time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research*, 2007. to appear.

[17] W. J. Hopp and M. L. Spearman. *Factory Physics*. McGraw-Hill, 2001.

[18] K. Huang and S. Ahmed. The value of multi-stage stochastic programming in capacity planning under uncertainty. *Operations Research*, 2008. to appear.

[19] K. Huang and S. Kücükyavuz. On stochastic lot-sizing problems with random lead times. *Operations Research Letters*, 2008. to appear.

[20] S. Kücükyavuz and Y. Pochet. Uncapacitated lot-sizing with backlogging: The convex hull. *Mathematical Programming*, 2008. to appear.

[21] C. Y. Lee, S. Cetinkaya, and A. Wagelmans. A dynamic lot-sizing model with demand time windows. *Management Science*, 47:1384–1395, 2001.

[22] G. Lulli and S. Sen. A branch-and-price algorithm for multi-stage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science*, 50:786–796, 2004.

[23] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

[24] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer, New York, 2006.

[25] A. Ruszczyński and A. Shapiro, editors. *Stochastic Programming. Handbooks in Operations Research and Management Science*, volume 10. Elsevier Science B. V., 2003.

[26] H. Scarf. The optimality of (s, S) policies for the dynamic inventory problem. In *Proceedings of the First Stanford Symposium of Mathematical Methods in the Social Sciences*. Stanford University Press, Stanford, CA, 1960.

[27] S. Sen and H. D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106:203 – 223, 2006.

[28] H. Stadtler. Multi-level lot-sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, 51:487–502, 2003.

[29] H. Tempelmeier and M. Derstroff. A Lagrangean–based heuristic for dynamic multi-level multi-item constrained lot-sizing with setup times. *Management Science*, 42:738–757, 1996.

[30] C. P. M. van Hoesel and A. Wagelmans. An $\mathcal{O}(T^3)$ algorithm for the economic lot-sizing problem with constant capacities. *Management Science*, 42:142–150, 1996.

[31] A. Wagelmans, A. van Hoesel, and A. Kolen. Economic lot sizing: An $\mathcal{O}(n \log n)$ algorithm that runs in linear time in the Wagner–Whitin case. *Operations Research*, 40:145–156, 1992.

[32] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5:89–96, 1958.

Appendix A. Proof of Proposition 4.1. By contradiction; we will show that any optimal solution that contains a node that violates (10) is a convex combination of optimal solutions, at least one of which contains one less node that violates (10).

Assume the claim is not correct. That is, for any optimal solution (x^*, y^*, s^*), there exists a node $i \in \mathcal{V}$ such that $0 < x_i^* < \mu_i$ and $x_i^* \neq d_{ik} - \sum_{j \in S} \mu_j - s_{i^-}^*$ for any combination of $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k) \backslash \mathcal{P}(i)$. Let $\Psi(i)$ be those descendants of node i that are the first nodes (except node i) with positive production quantity that is less than its capacity along a path from node i to a leaf node. Correspondingly, we use notation $\mathcal{L}_1(i)$ to represent the set of leaf nodes in $\mathcal{V}(i)$ such that there is a node (besides node i) with positive production quantity that is less than its capacity along the path from node i to each leaf node in this set, and notation $\mathcal{L}_2(i)$ to represent the set of leaf nodes in $\mathcal{V}(i)$ such that there is no node (except node i) with positive production quantity that is less than its capacity along the path from node i to any leaf node in this set. Then $\mathcal{L}_1(i) \cup \mathcal{L}_2(i) = \mathcal{L} \cap \mathcal{V}(i)$. Let

$$\Psi(i) = \cup_{\ell \in \mathcal{L}_1(i)} \text{argmin} \left\{ t(j) : j \in \mathcal{P}(\ell) \setminus \mathcal{P}(i) \text{ and } 0 < x_j^* < \mu_j \right\}.$$

Correspondingly, let $\Phi(i)$ be the set of nodes in the paths from node i to each node in $\Psi(i)$ plus nodes in the paths from node i to each leaf node in $\mathcal{L}_2(i)$; that is, $\Phi(i) = \cup_{k \in \Psi(i) \cup \mathcal{L}_2(i)} \mathcal{P}(k) \setminus \mathcal{P}(i^-)$. Let $\Lambda(i)$ be the set of nodes in $\Phi(i)$ that produce at their capacities. For instance, $\Lambda(i) = \{j \in \Phi(i) : x_j = \mu_j\}$. Define the cost corresponding to the nodes in $\mathcal{V} \setminus \Phi(i)$ as

$$\mathcal{G}(x^*, y^*, s^*) = \sum_{j \in \mathcal{V} \setminus \Phi(i)} (\alpha_j x_j^* + \beta_j y_j^* + \max\{h_j s_j^*, -b_j s_j^*\}).$$

Then, the objective value for the given optimal solution (x^*, y^*, s^*) is

$$\begin{aligned} \mathcal{F}(x^*, y^*, s^*) &= \mathcal{G}(x^*, y^*, s^*) + \alpha_i x_i^* + \beta_i y_i^* + \max\{h_i s_i^*, -b_i s_i^*\} \\ &\quad + \sum_{j \in \Phi(i) \setminus (\{i\} \cup \Psi(i))} \max\{h_j s_j^*, -b_j s_j^*\} + \sum_{j \in \Lambda(i)} (\alpha_j \mu_j + \beta_j) \\ &\quad + \sum_{j \in \Psi(i)} (\alpha_j x_j^* + \beta_j y_j^* + \max\{h_j s_j^*, -b_j s_j^*\}). \end{aligned}$$

Now consider two alternative solutions $(\bar{x}^*, \bar{y}^*, \bar{s}^*)$ and $(\hat{x}^*, \hat{y}^*, \hat{s}^*)$ in which

- $\bar{x}_i^* = x_i^* - \epsilon$, $\bar{s}_j^* = s_j^* - \epsilon$ for each $j \in \Phi(i) \setminus \Psi(i)$ and $\bar{x}_j^* = x_j^* + \epsilon$ for each $j \in \Psi(i)$,
- $\hat{x}_i^* = x_i^* + \epsilon$, $\hat{s}_j^* = s_j^* + \epsilon$ for each $j \in \Phi(i) \setminus \Psi(i)$ and $\hat{x}_j^* = x_j^* - \epsilon$ for each $j \in \Psi(i)$,

and other components are the same as (x^*, y^*, s^*). It is easy to observe that these two solutions satisfy constraints (1)–(3). Thus, they are feasible for SCLS with backlogging. According to our assumption, there must exist an $\epsilon > 0$ such that the following equation holds

$$\begin{aligned} \mathcal{F}(\bar{x}^*, \bar{y}^*, \bar{s}^*) &= \mathcal{G}(x^*, y^*, s^*) + (\alpha_i (x_i^* - \epsilon) + \beta_i y_i^* + \max\{h_i (s_i^* - \epsilon), -b_i (s_i^* - \epsilon)\}) \\ &\quad + \sum_{j \in \Phi(i) \setminus (\{i\} \cup \Psi(i))} \max\{h_j (s_j^* - \epsilon), -b_j (s_j^* - \epsilon)\} + \sum_{j \in \Lambda(i)} (\alpha_j \mu_j + \beta_j) \\ &\quad + \sum_{j \in \Psi(i)} (\alpha_j (x_j^* + \epsilon) + \beta_j y_j^* + \max\{h_j s_j^*, -b_j s_j^*\}) \\ &= \mathcal{F}(x^*, y^*, s^*) - \alpha_i \epsilon + \sum_{j \in \Phi(i) \setminus \Psi(i)} \zeta_j(\epsilon) + \sum_{j \in \Psi(i)} \alpha_j \epsilon, \end{aligned}$$

where $\zeta_j(\epsilon) = -h_j \epsilon$ if $s_j^* > 0$ and $\zeta_j(\epsilon) = b_j \epsilon$ if $s_j^* < 0$. Note here $s_j^* \neq 0$ for each $j \in \Phi(i) \setminus \Psi(i)$ according to our assumption.

Similarly, we have

$$\begin{aligned} \mathcal{F}(\hat{x}^*, \hat{y}^*, \hat{s}^*) &= \mathcal{G}(x^*, y^*, s^*) + (\alpha_i (x_i^* + \epsilon) + \beta_i y_i^* + \max\{h_i (s_i^* + \epsilon), -b_i (s_i^* + \epsilon)\}) \\ &\quad + \sum_{j \in \Phi(i) \setminus (\{i\} \cup \Psi(i))} \max\{h_j (s_j^* + \epsilon), -b_j (s_j^* + \epsilon)\} + \sum_{j \in \Lambda(i)} (\alpha_j \mu_j + \beta_j) \\ &\quad + \sum_{j \in \Psi(i)} (\alpha_j (x_j^* - \epsilon) + \beta_j y_j^* + \max\{h_j s_j^*, -b_j s_j^*\}) \\ &= \mathcal{F}(x^*, y^*, s^*) + \alpha_i \epsilon - \sum_{j \in \Phi(i) \setminus \Psi(i)} \zeta_j(\epsilon) - \sum_{j \in \Psi(i)} \alpha_j \epsilon, \end{aligned}$$

If $\alpha_i\epsilon - \sum_{j\in\Phi(i)\setminus\Psi(i)}\zeta_j(\epsilon) - \sum_{j\in\Psi(i)}\alpha_j\epsilon \neq 0$, then $\min\{\mathcal{F}(\bar{x}^*,\bar{y}^*,\bar{s}^*),\mathcal{F}(\hat{x}^*,\hat{y}^*,\hat{s}^*)\} < \mathcal{F}(x^*,y^*,s^*)$, contradicting the assumption that (x^*,y^*,s^*) is an optimal solution. Otherwise, if $\alpha_i\epsilon - \sum_{j\in\Phi(i)\setminus\Psi(i)}\zeta_j(\epsilon) - \sum_{j\in\Psi(i)}\alpha_j\epsilon = 0$, then ϵ can be increased such that $\bar{x}_i^* = d_{ik} - \sum_{j\in S}\mu_j - s_{i^-}^*$ or $\hat{x}_i^* = d_{ik} - \sum_{j\in S}\mu_j - s_{i^-}^*$ for some node $k \in \Phi(i)$ and $S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(i)$. Without loss of generality, assume $\bar{x}_i^* = d_{ik} - \sum_{j\in S}\mu_j - s_{i^-}^*$ for some node $k \in \Phi(i)$ and $S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(i)$. Then $(\bar{x}^*,\bar{y}^*,\bar{s}^*)$ is also an optimal solution.

A similar argument can be applied to $(\bar{x}^*,\bar{y}^*,\bar{s}^*)$ if there exists a node $j \in \mathcal{V}$ such that $\bar{x}_j^* \neq d_{jk} - \sum_{r\in S}\mu_r - s_{j^-}^*$ for any combination of $k \in \mathcal{V}(j)$ and $S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(j)$ and consecutively the following ones. We will eventually either obtain a solution (x,y,s) such that $x_i = d_{ik} - \sum_{j\in S}\mu_j - s_{i^-}^*$ for some node $k \in \mathcal{V}(i)$ and $S \subseteq \mathcal{P}(k)\setminus\mathcal{P}(i)$ if $0 < x_i < \mu_i$ corresponding to each node $i \in \mathcal{V}$, or find a solution with a smaller objective function value, contradicting the original assumption. Therefore, the original conclusion holds. □