Isotonic Optimization with Fixed Costs

Xin Chen[1], Shuai Li[1], Weijun Xie[1]
[1] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA
xin.chen@isye.gatech.edu, sli884@gatech.edu, wxie@gatech.edu

This paper introduces a generalized isotonic optimization framework over an arborescence graph, where each node incurs state-dependent convex costs and a fixed cost upon strict increases. We begin with the special case in which the arborescence is a directed path and develop a dynamic programming (DP) algorithm whose DP stage, excluding preprocessing, has an initial complexity of $O(n^3)$, which we improve to $O(n^2 \log n)$ by exploiting problem structure, and further to $O(n)$ under isotonic minima and constant-time block-data queries assumptions via the Monge property. For general arborescences, where the path-based DP is inapplicable, we identify structural conditions under which the problem remains tractable. In particular, we develop polynomial-time algorithms for cases with piecewise linear convex costs, dominating fixed costs at parent nodes, or arborescences of logarithmic height or bounded Horton–Strahler number, provided that sums of node costs can be minimized and their intersections with horizontal lines can be computed exactly in polynomial time. For general arborescences, we show tight worst-case bounds of $n^{\Theta(\log n)}$ on the number of convex pieces required to represent the parametric DP value functions. Numerical experiments on synthetic instances demonstrate the computational advantages of our proposed algorithms over the big-M baselines.

Key words: isotonic optimization, fixed costs, dynamic programming, arborescence

1. Introduction

We consider a mixed integer nonlinear optimization problem on an arborescence graph. Specifically, we define an arborescence $\mathcal{T} = (N_{\mathcal{T}}, A_{\mathcal{T}})$ as a directed tree where all edges are directed away from the root. The node set and edge set are given by $N_{\mathcal{T}} = 0 \cup [n]$ and $A_{\mathcal{T}} = \{(p(i), i) \mid i \in [n]\}$, where $[n] = \{1, ..., n\}$ and $p(i)$ denote the parent node of node i. Without loss of generality, we assume that the root node 0 is only connected to node 1. For each $i \in [n]$, we attach it with an associated one-dimensional state x_i and a corresponding convex state cost $f_i(x_i)$. We impose isotonic constraints for variables whose indices form an arc in $A_{\mathcal{T}}$, i.e., $x_{p(i)} \leq x_i$ for all $i \in [n]$. In addition, if $x_{p(i)} < x_i$, a fixed cost $K_i \geq 0$ is incurred. The objective is to minimize the overall state costs and fixed costs incurred by all nodes in $\mathcal{T}$. The isotonic requirements and the need to incorporate a fixed cost whenever a strict increase is made arise in many applications. We refer to Section 1.1 for several more detailed motivating examples. The above generalized isotonic optimization model can be formulated

as follows:

$$\begin{aligned} \min_{(\boldsymbol{x},\boldsymbol{y})\in\mathbb{R}^n\times\{0,1\}^n} & \sum_{i\in[n]} K_i y_i + \sum_{i\in[n]} f_i(x_i) \\ \text{s.t.} \quad & (x_{p(i)} - x_i)(1-y_i) = 0, \quad \forall i \in [n], \\ & x_{p(i)} \le x_i, \quad \forall i \in [n], \end{aligned} \tag{Arborescence}$$

where x_0 is a given parameter, and for each node $i \in [n]$. A strict increase $x_i > x_{p(i)}$ requires $y_i = 1$. Since $K_i \ge 0$, an optimal solution can be chosen with $y_i = 0$ whenever $x_i = x_{p(i)}$. In the Arborescence problem, the first set of constraints ensures that a fixed cost is incurred whenever a strict increase occurs from a parent node. The second set of constraints enforces the isotonic inequalities on the arborescence. When $\mathcal{T}$ is the directed path $0 \to 1 \to \cdots \to n$, hereafter simply called a path, the problem simplifies to the following:

$$\begin{aligned} \min_{(\boldsymbol{x},\boldsymbol{y})\in\mathbb{R}^n\times\{0,1\}^n} & \sum_{i\in[n]} K_i y_i + \sum_{i\in[n]} f_i(x_i) \\ \text{s.t.} \quad & (x_i - x_{i-1})(1-y_i) = 0, \quad \forall i \in [n], \\ & x_0 \le x_1 \le x_2 \le \cdots \le x_n. \end{aligned} \tag{Path}$$

Throughout, we assume that each convex cost function f_i has a finite smallest minimizer. Without loss of optimality, all x_i can be restricted to a closed bounded interval containing these minimizers and x_0 (when finite). Both Arborescence and Path problems therefore have an optimal solution, since the restricted feasible set is compact and the objective is continuous. These problems are mixed-integer nonlinear programs (MINLPs), which are, in general, hard to solve. The purpose of this paper is to design efficient polynomial-time algorithms for the Path problem and a broad family of the Arborescence problem.

1.1. Motivating Examples

This subsection presents three motivating examples from the existing literature, each of which can be viewed as a special case of the Arborescence and Path problems.

Generalized Reduced Isotonic Regression: Isotonic regression is a well-established class of isotonic optimization problems (see, e.g., Barlow and Brunk 1972, Robertson and Wright 1975, Barlow and Ubhaya 1971, Bacchetti 1989). Given a sequence of univariate data points, the objective of the isotonic regression problem is to determine the best monotone nondecreasing sequence that optimally fits the data with respect to a specified convex function. Formally, suppose there are n data points, and let $f_i(x_i) : \mathbb{R} \to \mathbb{R}_+$ be a convex function that measures the discrepancy between the fitted output x_i and the observation. The isotonic regression problem can be expressed in the following general form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \sum_{i\in[n]} f_i(x_i) \,|\, x_1 \le x_2 \le \cdots \le x_n \right\}. \tag{1}$$

Classical isotonic regression (1) does not penalize the number of fitted levels. A strictly increasing fit $x_1 < x_2 < \cdots < x_n$ has n distinct levels and $n-1$ strict increases. Allowing many fitted levels can lead to overfitting (Niculescu-Mizil and Caruana 2005, Salanti and Ulm 2003, Schell and Singh 1997). Besides, a fitted step functions with too many steps occupy a lot of storage space on the computer. To address the overfitting issue in the isotonic optimization (1), the *reduced isotonic regression* (Schell and Singh 1997) is introduced to restrict the number of strict increases to be within a threshold $s \in \mathbb{N}$; i.e., an additional constraint $\sum_{i=2}^{n} \mathbb{1}\{x_i > x_{i-1}\} \le s$ is added. The authors show that the reduced isotonic regression can be solved in polynomial time, and subsequent research has improved the time complexity (Haiminen et al. 2008, Hardwick and Stout 2014). Here, we adopt another formulation to impose sparsity. Instead of restricting the number of changes to be within a cardinality, we add a penalty term $\sum_{i=2}^{n} K_i \mathbb{1}\{x_i > x_{i-1}\}$ to the objective of the isotonic optimization (1) with appropriate penalty parameters $K_i \in \mathbb{R}_+$. This leads to the following penalized formulation:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{y}\in\{0,1\}^{n-1}} \quad & \sum_{i\in[n]\setminus\{1\}} K_i y_i + \sum_{i\in[n]} f_i(x_i) \\ \text{s.t.} \quad & (x_i - x_{i-1})(1-y_i) = 0, \quad \forall i \in [n]\setminus\{1\}, \\ & x_1 \le x_2 \le \cdots \le x_n, \end{aligned} \tag{GIR}$$

where $K_i \in \mathbb{R}_+$'s are fixed costs which are only incurred when $x_i > x_{i-1}$. We call (GIR) the *generalized reduced isotonic regression* problem, which is a special case of the Path problem. This formulation allows location-dependent penalties K_i. These penalties let the regularization strength vary across regions according to application requirements. For this application, we focus on developing algorithms that accommodate general convex functions within the Path framework.

Open-Loop Stochastic Lot-Sizing: The stochastic lot-sizing problem is extensively studied in the inventory management literature (see a comprehensive overview in Tempelmeier 2013). Consider a periodic review open-loop stochastic lot-sizing problem over a time horizon of length T. Let the random variable $d_t \in \mathbb{R}^+$ represent the stochastic demand at time t, with a cumulative distribution function $F_t(\cdot)$ for each $t \in [T]$. Let the decision variable $x_t \in \mathbb{R}^+$ denote the ordering quantity placed at time t for each $t \in [T]$. At time t, three types of costs are incurred. The fixed ordering cost $K_t \in \mathbb{R}^+$ applies whenever $x_t > 0$. The holding cost $h_t \in \mathbb{R}^+$ is associated with inventory storage, while the backlog cost $b_t \in \mathbb{R}_+$ accounts for unmet demand. The unit ordering costs can be subsumed into the holding and backlog costs and hence normalized to zero without loss of generality (Levi et al. 2007, Section 4.6), provided that the adjusted cost coefficients remain positive. The stochastic lot-sizing problem can be formulated as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}_+^T} \sum_{t\in[T]} \left(K_t \mathbb{1}\{x_t > 0\} + \mathbb{E}\left[h_t \left(\sum_{i\in[t]} x_i - \sum_{i\in[t]} d_i \right)^+ + b_t \left(\sum_{i\in[t]} d_i - \sum_{i\in[t]} x_i \right)^+ \right] \right). \tag{2}$$

It is worth noting that Vargas (2009) designed a dynamic programming algorithm to solve a special case of the stochastic lot-sizing problem, where the ratio of backlog and holding costs must be non-decreasing. The general stochastic lot-sizing problem can be treated as a special case of the Path problem. To see this, we define $y_t = \mathbb{1}\{x_t > 0\} \in \{0,1\}$ as an indicator variable denoting whether an order is placed at time t. We further define $D_t = \sum_{i\in[t]} d_i$ as the cumulative stochastic demand from time 1 to t and $X_t = \sum_{i\in[t]} x_i$ as the cumulative ordering quantity over the same period. The cost can then be represented in terms of the cumulative ordering quantity X_t and the cumulative demand D_t. By definition, the cumulative ordering quantities satisfy $0 = X_0 \le X_1 \le X_2 \le \cdots \le X_T$. If $y_t = 1$, a fixed cost K_t is incurred at time t, and the cumulative ordering quantity X_t can increase strictly beyond X_{t-1} by placing an order. Additionally, we define the function $f_t(X_t) := \mathbb{E}[h_t(X_t - D_t)^+ + b_t(D_t - X_t)^+]$, which is convex. Thus, the stochastic lot-sizing problem (2) can be equivalently expressed as:

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^T, \boldsymbol{y}\in\{0,1\}^T} \quad & \sum_{t\in[T]} \Big(K_t y_t + f_t(X_t) \Big) \\ \text{s.t. } & (X_t - X_{t-1})(1-y_t) = 0, \quad \forall t \in [T], \\ & 0 = X_0 \le X_1 \le X_2 \le \ldots \le X_T, \end{aligned} \tag{SLS-O}$$

which is a special case of the Path problem.

Closed-Loop Stochastic Lot-Sizing: The aforementioned SLS-O is a two-stage open-loop model; that is, the order quantities are determined at the beginning of the planning horizon and do not rely on a particular realization of random demands. One multi-stage *closed-loop* stochastic lot-sizing model based on a scenario tree has been studied by Guan and Miller (2008), Guan (2011). Different from SLS-O, in the scenario-tree-based model, the order quantity decision at the current stage depends on past realizations. Suppose that there is a T-stage scenario tree $\mathcal{T} = (N_\mathcal{T}, A_\mathcal{T})$, where each node $i \in N_\mathcal{T} := \{0\} \cup [n]$ corresponds to a particular realization. Each node $i \in N_\mathcal{T}$ represents a realization at some stage, containing: a set of its child nodes $\mathcal{C}(i)$, a realization probability p_i (note that $p_i = \sum_{j\in\mathcal{C}(i)} p_j$ for each nonleaf node i), a demand realization d_i, a fixed ordering cost K_i, a unit holding cost h_i, and a unit backlog cost b_i. For notational simplicity, we absorb the probability p_i into all cost parameters associated with node i. Let a binary decision $y_i \in \{0,1\}$ indicate whether to order at node i or not, and let the continuous variable $x_i \in \mathbb{R}_+$ denote the order quantity at node i. For a node i and a descendant j (possibly $j = i$), let $\mathcal{P}(i,j)$ denote the set of nodes on the unique directed path from i to j. At each node i, there are fixed ordering cost $K_i y_i$ and the inventory cost $h_i(\sum_{\ell\in\mathcal{P}(1,i)} x_\ell - \sum_{\ell\in\mathcal{P}(1,i)} d_\ell)^+ + b_i(\sum_{\ell\in\mathcal{P}(1,i)} d_\ell - \sum_{\ell\in\mathcal{P}(1,i)} x_\ell)^+$. We can also incorporate a linear ordering cost $c_i x_i$ at each node. The total ordering cost satisfies $\sum_{i\in[n]} c_i x_i = \sum_{i\in[n]} \left(c_i - \sum_{j\in\mathcal{C}(i)} c_j\right) \sum_{\ell\in\mathcal{P}(1,i)} x_\ell$. These costs can be absorbed into the inventory costs by adding $c_i - \sum_{j\in\mathcal{C}(i)} c_j$ to h_i and subtracting it from b_i, up to the additive constant $\sum_{i\in[n]} \left(c_i - \sum_{j\in\mathcal{C}(i)} c_j\right) \sum_{\ell\in\mathcal{P}(1,i)} d_\ell$ which is independent of the

ordering decisions. Provided that the adjusted coefficients are positive, each modified node cost has a finite smallest minimizer. We henceforth use h_i and b_i for these adjusted coefficients. Similar to SLS-O model, by defining $X_i = \sum_{\ell \in \mathcal{P}(1,i)} x_\ell$ and $D_i = \sum_{\ell \in \mathcal{P}(1,i)} d_\ell$, the optimization problem can be reformulated as:

$$\begin{aligned} \min_{(\boldsymbol{X},\boldsymbol{y}) \in \mathbb{R}^n \times \{0,1\}^n} & \sum_{i \in [n]} K_i y_i + \sum_{i \in [n]} \left(h_i (X_i - D_i)^+ + b_i (D_i - X_i)^+ \right) \\ \text{s.t.} \quad & (X_{p(i)} - X_i)(1 - y_i) = 0, \quad \forall i \in [n], \\ & X_{p(i)} \le X_i, \quad \forall i \in [n], \\ & X_0 = 0. \end{aligned} \tag{SLS-C}$$

Our framework applies whether or not ordering decisions use the current demand observation; when they do not, the inventory costs are replaced by their conditional expectations. We observe that the SLS-C problem is a special case of Arborescence problem where each f_i is piecewise linear convex with only one kink point.

Joint Inventory Control and Pricing with Costly Price Markdown: Let us consider a deterministic joint inventory control and pricing model with finite horizon T studied by Chen and Hu (2012). At the beginning of each time $t \in [T]$, the retailer decides the ordering quantity $x_t \in \mathbb{R}_+$ and the price $p_t \in \mathbb{R}_+$ of the product. Let $c_t \in \mathbb{R}_+$ denote the unit ordering cost and d_t denote the price-dependent demand at time t. Let the remaining inventory at the beginning of time t be I_{t-1}. At time t, the cost $h_t(I_{t-1} + x_t - d_t)^+ + b_t(d_t - I_{t-1} - x_t)^+$ is incurred, where $h_t \in \mathbb{R}+$ and $b_t \in \mathbb{R}+$ are the unit holding and backlog costs, respectively, with $b_t > c_t$. With zero initial inventory and no ordering capacity constraints, each unit of demand can be fulfilled independently. Since $b_t > c_t$, ordering in period t is cheaper than backlogging its demand. Thus, the minimum unit fulfillment cost for period t is $\widetilde{c}_t = \min_{1 \le s \le t} \{c_s + \sum_{\ell=s}^{t-1} h_\ell\}$, which depends only on the cost parameters. The minimum total ordering and inventory cost is therefore $\sum_{t \in [T]} \widetilde{c}_t d_t$. Let the demand function be $D_t(p_t) = \alpha_t D(p_t)$, where $\alpha_t > 0$ is the time-dependent multiplier and $D : [0, \overline{p}] \to \mathbb{R}_+$ is the time-independent demand function, assumed continuous and strictly decreasing with $D(\overline{p}) = 0$. Prices are restricted to $p_t \in [0, \overline{p}]$. Then we can equivalently express the price function in demand d_t as $P_t(d) = D^{-1}(d_t/\alpha_t)$ and the revenue function as $R_t(d_t) = d_t \times P_t(d_t)$. If we let $d_t/\alpha_t = \widetilde{d}_t$, then $R_t(d_t) = \alpha_t R(\widetilde{d}_t)$, where $R(\widetilde{d}_t) = \widetilde{d}_t \times D^{-1}(\widetilde{d}_t)$ and is assumed to be a concave function on $[0, D(0)]$. We assume the price markdown is allowed, i.e., we impose the constraint $p_1 \ge p_2 \ge \ldots \ge p_T$, which can be equivalently expressed as $\widetilde{d}_1 \le \widetilde{d}_2 \le \ldots \le \widetilde{d}_T$. We further assume that price change is costly, and the costs consist of menu cost

and management cost. Assume a fixed cost $A_t \in \mathbb{R}_+$ is charged when $p_t < p_{t-1}$ or $\widetilde{d}_{t-1} < \widetilde{d}_t$. Hence, we can formulate the model as:

$$\begin{aligned} \min_{\widetilde{\boldsymbol{d}} \in [0, D(0)]^T, \boldsymbol{z} \in \{0,1\}^T} \quad & \sum_{t \in [T]} A_t z_t + \sum_{t \in [T]} \alpha_t \left(\widetilde{c}_t \widetilde{d}_t - R(\widetilde{d}_t) \right) \\ \text{s.t.} \quad & (\widetilde{d}_t - \widetilde{d}_{t-1})(1 - z_t) = 0, \quad \forall t \in [T], \\ & 0 = \widetilde{d}_0 \le \widetilde{d}_1 \le \widetilde{d}_2 \le \ldots \le \widetilde{d}_T. \end{aligned} \tag{CPM}$$

Each node cost $\alpha_t(\widetilde{c}_t \widetilde{d}_t - R(\widetilde{d}_t))$ is continuous and convex on the compact interval $[0, D(0)]$, and hence has a finite smallest minimizer. Thus, CPM has the Path structure.

1.2. Literature Review

There are three streams of relevant literature: isotonic regression, general isotonic optimization problems, and application problems with isotonic constraints. Most existing works do not incorporate fixed costs into their models and can be solved in polynomial time using well-developed algorithms.

The literature on isotonic regression primarily focuses on problems of the form (1), where the convex function $f_i(x_i)$ is typically of the form $w_i|x_i - b_i|^p$ for $1 \le p < \infty$, with $w_i \ge 0$ for each $i \in [n]$. When $p = 1$, the problem is referred to as isotonic median regression (Robertson and Wright 1973, Chakravarti 1989). When $p = 2$, the problem corresponds to the standard isotonic regression problem, which has been widely studied in the literature (Barlow and Brunk 1972, Best and Chakravarti 1990, Dykstra 1981). For general convex functions, Best et al. (2000) showed that the Pool Adjacent Violators (PAV) algorithm can solve the isotonic regression problem optimally. Subsequent research has aimed at improving the computational efficiency of solving isotonic regression problems. For example, for isotonic median regression on a path, Ahuja and Orlin (2001) developed a scaling algorithm, while Stout (2013) developed a partitioning algorithm, both achieving $O(n \log n)$ time. Beyond these standard formulations, the literature has explored alternative convex and non-convex objectives for problem (1). For instance, L_∞-norm isotonic regression considers an objective of the form $\max_{i \in [n]} w_i|\widehat{y}_i - y_i|$ and has been studied in Stout (2018, 2014, 2015). Similarly, L_0-norm isotonic regression minimizes the number of misclassified values, with the objective $\sum_{i \in [n]} w_i \delta(\widehat{y}_i \ne y_i)$, where $\delta(\widehat{y}_i \ne y_i) = 1$ if $\widehat{y}_i \ne y_i$ and 0, otherwise (Stout 2021a,b). For a comprehensive overview of the fastest known algorithms for these isotonic regression problems, readers may refer to Stout (2019, 2023). However, the algorithms reviewed above address isotonic regression models without fixed costs for strict increases considered in this paper.

Existing literature also studies isotonic optimization problems in more general settings, extending the classical isotonic regression formulations. One extension involves "soft" isotonic models, where isotonic constraints can be violated with a penalty cost. Examples of such works include nearly

isotonic regression (Tibshirani et al. 2011), the convex s-excess problem (Hochbaum 2001), and the convex dual minimum-cost network flow problem (Ahuja et al. 2003). Another line of research enforces isotonic constraints on the edges of general directed acyclic graphs (DAGs). For example, Stout (2008) studied unimodal regression, where the non-decreasing constraints in (1) are replaced with unimodal constraints of the form $\widehat{y}_1 \leq \ldots \leq \widehat{y}_m \geq \widehat{y}_{m+1} \geq \ldots \geq \widehat{y}_n$ for some $m \in [n]$. Other works investigate isotonic regression on arborescences (Pardalos and Xue 1999, Stout 2013, 2015) and on general DAGs (Wang et al. 2022). Additionally, some studies apply soft isotonic models to more general structures, such as paths (Yu et al. 2023) and directed trees (Chen et al. 2024), designing optimal dynamic programming or PAV-based algorithms to efficiently solve these problems.

Beyond isotonic regression, isotonic constraints play a significant role in modeling various classical operations research problems. One notable example is the extension of the Economic Order Quantity (EOQ) model to multistage production systems, including serial, assembly, and distribution systems, as well as more general system structures (Schwarz and Schrage 1975, Crowston et al. 1973, Love 1972, Maxwell and Muckstadt 1985). Beyond multistage inventory systems, isotonic constraints also arise in other operational problems, such as the center location problem (Kaufman and Tamir 1993, Tamir 1993) and various other applications surveyed in Hochbaum (2004). However, the cited applications do not impose fixed costs on strict increases along the isotonic constraints as in our model.

1.3. Summary of Contributions

We present our main contributions in detail below. Table EC.1 in Appendix E.1 lists the special and general cases considered in this paper and specifies which algorithm applies to each.

- *Modeling Power*: The proposed Arborescence problem and Path models can cover many problems from the literature, including generalized reduced isotonic regression, stochastic lot sizing, joint inventory control, and pricing with costly price markdowns.
- *Efficient Algorithms for the Path Problem*: To solve the Path problem efficiently, we develop a dynamic programming (DP) algorithm that leverages optimality conditions. While a naive DP implementation has a time complexity of $O(n^3)$, we improve it to $O(n^2 \log n)$ by exploiting the problem's structural properties. Furthermore, when the convex cost functions exhibit isotonic minima, we prove that the DP algorithm can be further improved to achieve a linear runtime $O(n)$. These bounds exclude block-data preprocessing; the linear bound additionally assumes constant-time queries for block costs and minimizers.
- *Efficient Algorithms for the Arborescence Problem*: The DP algorithms developed for the Path problem may not be directly applicable to the Arborescence problem. To address this, we design new DP algorithms specifically for solving the Arborescence problem. In particular, we show that when cost functions are convex piecewise linear with a constant number of pieces, the Arborescence

problem can be solved in $O(n^2)$ time. This result recovers the best-known complexity for the scenario-tree-based lot-sizing problem but with a much simpler algorithm. For general convex cost functions, we prove that the corresponding value-to-go functions are piecewise convex in their states. We further show that the number of convex pieces is polynomial in n when $K_i \geq \sum_{j \in \mathcal{C}(i)} K_j$ for every node i, or when the arborescence has logarithmic height or bounded Horton–Strahler number. These bounds yield polynomial-time algorithms provided that the required sums of node costs can be minimized over intervals and their intersections with horizontal lines computed exactly in polynomial time.

- *Tight Bounds for General Arborescences*: We provide a tight worst-case bound of the order $n^{\Theta(\log n)}$ on the minimum number of convex interval pieces needed to represent the parametric DP value functions for the general Arborescence problem, combining a general upper bound with a constructive lower bound. Although the general problem may admit a polynomial-time algorithm using a different approach, our tight quasi-polynomial bound shows that explicitly maintaining the parametric DP value functions requires superpolynomial time and space in the worst case. This demonstrates a fundamental computational obstacle in extending this approach to general arborescences.
- *Effective Numerical Experiments*: On the tested synthetic instances, our proposed algorithms efficiently find optimal solutions, whereas the big-M formulation solved by Gurobi reaches the time limit on most instances. For general arborescences, the observed piece counts remain modest in the tested random instance family despite the worst-case quasi-polynomial bound. Additional experiments examine sensitivity to the Horton–Strahler number, which is a measure of the tree's branching complexity, and evaluate the algorithms on open-loop and closed-loop stochastic lot-sizing instances derived from real sales data.

Paper Structure, Notations, and Computational Convention: The structure of the paper is as follows. Section 2 investigates a DP algorithm for solving the Path problem and enhances its time complexity for both the general case and a special case where the functions f_i exhibit isotonic minima. Section 3 and Section 4 extend the study to the more general Arborescence problem and develop efficient polynomial-time DP algorithms for special cases, where Section 3 considers two special cost structures and Section 4 considers special arborescence structures. Section 5 provides tight quasi-polynomial bounds on convex-piece counts for general arborescences. Section 6 compares the computational performance of the proposed algorithms with that of the tested big-M baselines. Finally, Section 7 summarizes the findings and concludes the paper.

In this paper, all vectors are written in boldface, while scalars and matrices are not. $[n] := \{1, \ldots, n\}$ for any $n \in \mathbb{Z}_{++}$. $\mathbb{1}\{\text{condition}\}$ equals 1 if condition = YES and equals 0 if condition = NO. Denote $f'_-(x)$ and $f'_+(x)$ as the left and right derivatives of a real-valued function $f : \mathbb{R} \to \mathbb{R}$ at x if they exist. A function $f(n)$ is said to be $O(g(n))$ as $n \to \infty$ if there exist positive constants M and n_0 such that $|f(n)| \leq M|g(n)|$ for all $n \geq n_0$. For an arborescence graph $\mathcal{T} = (N_\mathcal{T}, A_\mathcal{T})$, let $\mathcal{T}(i) = (N_{\mathcal{T}(i)}, A_{\mathcal{T}(i)})$

be the sub-arborescence of $\mathcal{T}$ rooted at $i \in N_{\mathcal{T}}$, and, for $j \in N_{\mathcal{T}(i)}$, let $\mathcal{P}(i,j)$ denote the set of nodes on the unique directed path from i to j. We denote $\mathcal{L}$ as the set of leaf nodes in $\mathcal{T}$, and $\mathcal{L}(i)$ as the set of leaf nodes in $\mathcal{T}(i)$. All the proofs in this paper are delegated in the appendix.

We assume that arithmetic operations, comparisons, and exact evaluations of a single cost function take constant time. Evaluating a sum of m cost functions therefore takes $O(m)$ time. Preprocessing costs are specified separately. Space complexity counts the numbers and indices the algorithm stores, excluding input and oracle storage. These bounds do not measure bit complexity.

2. DP Algorithms for the Path Problem

This section first presents a DP algorithm to solve the Path problem, leveraging its optimality condition. We then improve this DP algorithm by eliminating redundant computations to increase efficiency. Additionally, we introduce a new DP algorithm with significantly improved time complexity for a special case.

The algorithms and complexity bounds in this section are stated for $x_0 = -\infty$. At the model level, when $x_0 = \ell$ instead of $-\infty$, we can insert an index $0'$ between 0 and 1 and associate a fixed cost $K_{0'} = 0$ and a convex cost function $f_{0'}(x_{0'}) = \delta(x_{0'} = \ell)$ to the index, where $\delta(A) = 0$ if $A =$ YES and ∞ otherwise, to recover the original (Path) problem. The preservation of the relevant algorithmic assumptions and complexity bounds under this reformulation should be verified case by case.

2.1. A DP Algorithm and Its Efficient Implementation

We first provide an optimality condition for the Path problem, which serves as the foundation for designing a DP algorithm to solve it. To begin with, we introduce key notations that are frequently used throughout this paper. For each pair (i,j) satisfying $1 \le i \le j \le n$, define $\overline{x}^*_{i,j}$ as (i,j)-opt solution and $c_{i,j}$ as its associated cost, which admit the following expressions:

$$\overline{x}^*_{i,j} = \min\left\{x^* | x^* \in \arg\min \sum_{\ell=i}^{j} f_\ell(x)\right\}, c_{i,j} = K_i + \sum_{\ell=i}^{j} f_\ell(\overline{x}^*_{i,j}). \tag{3}$$

Any optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the Path problem partitions $[n]$ into distinct blocks, each of which consists of a subset of consecutive indices, and the corresponding x^*_i values in a block are the same. Denote $B(i,j)$ to be the block containing indices from i to j. For any two consecutive blocks $B(i,j)$ and $B(j+1,k)$ in the partition, the values must satisfy the monotonicity condition $x^*_i = \ldots = x^*_j < x^*_{j+1} = \ldots = x^*_k$. Thus, $\overline{x}^*_{i,j}$ can be interpreted as the smallest optimal solution of the unconstrained problem for the block $B(i,j)$, with $c_{i,j}$ representing its corresponding cost. This result is formally summarized below.

THEOREM 1. *Suppose* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *is an optimal solution for the Path problem chosen with* $y^*_i = \mathbb{1}\{x^*_i > x^*_{i-1}\}$ *for all* $i \in [n]$*, such that* $y^*_i = 1$ *if* $i \in \{i_0, i_1, i_2, \ldots, i_R\} \subseteq [n]$ *where* $1 = i_0 < i_1 < i_2 < \ldots < i_R < i_{R+1} = n+1$ *and* $y^*_i = 0$ *otherwise. Then*

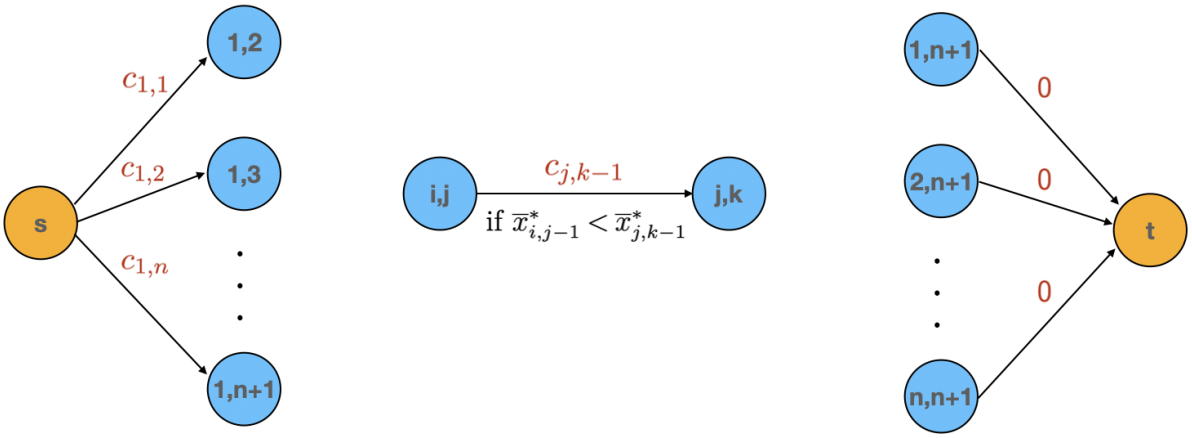


Figure 1 Illustration of the graph where finding the shortest (s,t)-path in it is equivalent to solving (DP-original). Each blue node (i,j) with $1 \le i < j \le n+1$ represents the block $B(i,j-1)$ and two yellow nodes are s and t. The arcs pointing out from s only towards $(1,2),(1,3),\ldots,(1,n+1)$ whose costs are $c_{1,1},c_{1,2},\ldots,c_{1,n}$, and arcs pointing to t only from $(1,n+1),(2,n+1),\ldots,(n,n+1)$ whose costs are all 0. Between any two blue nodes $(i,j),(j,k)$, an arc can only be placed between them if they satisfy $\overline{x}^*_{i,j-1} < \overline{x}^*_{j,k-1}$, and the cost is $c_{j,k-1}$.

(a) $x^*_i = \overline{x}^*_{i_{r-1},i_r-1}$ *for all* $i \in B(i_{r-1}, i_r - 1)$*; and*

(b) The strict inequality $\overline{x}^*_{i_{r-1},i_r-1} < \overline{x}^*_{i_r,i_{r+1}-1}$ *must hold for any* $r \in [R]$.

The optimality condition in Theorem 1 allows us to design an exact DP algorithm. The algorithm begins by precomputing all values of $\overline{x}^*_{i,j}$ and $c_{i,j}$. The goal of the DP algorithm is to determine the optimal partition of the set $[n]$ into blocks $B(1,i_1-1), B(i_1,i_2-1),\ldots,B(i_R,n)$, such that the sequence satisfies $\overline{x}^*_{1,i_1-1} < \overline{x}^*_{i_1,i_2-1} < \ldots < \overline{x}^*_{i_R,n}$, while minimizing the total cost $\sum_{r\in[R+1]} c_{i_{r-1},i_r-1}$.

For each (j,k) pair such that $1 \le j < k \le n+1$, we define the value-to-go function $V(j,k)$ as the minimum cost incurred by the first $k-1$ indexes in the Path problem, given that the last block in the partition of the index set $[k-1]$ is $B(j,k-1)$, or equivalently, the last index that has a strict increase before k is j. That is, we consider

$$V(j,k) = \begin{cases} c_{1,k-1}, & \text{if } 1 = j < k \le n+1, \\ c_{j,k-1} + \min_{i:1\le i<j,\, \overline{x}^*_{i,j-1} < \overline{x}^*_{j,k-1}} V(i,j), & \text{if } 2 \le j < k \le n+1. \end{cases} \qquad \text{(DP-original)}$$

The value $\min_{j\in[n]} V(j,n+1)$ corresponds to the optimal value of the Path problems. Note that the formulation (DP-original) can also be interpreted as finding the (s,t) shortest path of the graph illustrated in Figure 1.

The formulation (DP-original), assuming all $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s are precomputed, has a time complexity of $O(n^3)$. This complexity arises because there are $O(n^2)$ states, and for each state, the

naive computation of the value-to-go function requires $O(n)$ arithmetic operations (i.e., additions and comparisons). To accelerate the naive implementation, we observe that for each $j = 2, \ldots, n$, the evaluations of $V(j,k)$ for any $k = j+1, \ldots, n+1$ are structurally similar, differing only in the placement of different upper bounds when selecting the optimal i to evaluate $\min_{i:1\le i<j, \overline{x}^*_{i,j-1}<\overline{x}^*_{j,k-1}} V(i,j)$. Specifically, for any $k = j+1, \ldots, n+1$, computing $V(j,k)$ requires identifying the optimal preceding index i such that $V(i,j)$ is minimized over all i's with $\overline{x}^*_{i,j-1}$ strictly less than $\overline{x}^*_{j,k-1}$. We observe that although the valid i's for each k depend on $\overline{x}^*_{j,k-1}$, they are all drawn from a common pool. Algorithm 1 takes advantage of this by first sorting the combined set of values $\{\overline{x}^*_{i,j-1}\}_{i=1}^{j-1} \cup \{\overline{x}^*_{j,k-1}\}_{k=j+1}^{n+1}$ in ascending order. Suppose $\overline{x}^*_{j,k-1}, \overline{x}^*_{j,k'-1}$ are two consecutive elements in the sorted list with $\overline{x}^*_{j,k-1} \le \overline{x}^*_{j,k'-1}$ and $k, k' > j$. When $V(j,k)$ has already been computed, to compute $V(j,k')$ we only need to take the minimum over all $V(i,j)$'s with i's such that $\overline{x}^*_{j,k-1} \le \overline{x}^*_{i,j-1} < \overline{x}^*_{j,k'-1}$, and compare it with $\min_{i:1\le i<j, \overline{x}^*_{i,j-1}<\overline{x}^*_{j,k-1}} V(i,j)$ which has already been computed and recorded when computing $V(j,k)$. This prevents us from scanning and doing comparisons over $V(i,j)$ for the same i multiple times, and will ultimately save computation. The implementation details are provided in Algorithm 1, with correctness and time complexity analyzed in Proposition 1. Notably, this optimization reduces the time complexity from $O(n^3)$ to $O(n^2 \log n)$, significantly improving computational efficiency. Additionally, the time complexity of precomputing $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s must be considered when evaluating the overall complexity of the algorithm. This aspect will be analyzed in the subsequent subsections.

PROPOSITION 1. *Suppose that $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s are precomputed. Then, the DP recursion* (DP-original) *can be efficiently implemented using Algorithm 1 in $O(n^2 \log n)$ time.*

2.2. Acceleration with Isotonic Minima

In this subsection, we study a special case of the Path problem that satisfies the following assumption.

ASSUMPTION 1 **(Isotonic Minima)**. *Let $\overline{x}^*_{i,i} = \min\{x \mid x \in \arg\min f_i(x)\}$ for each $i \in [n]$. We refer to the sequence $\{\overline{x}^*_{i,i}\}_{i\in[n]}$ as isotonic minima if it satisfies the condition $\overline{x}^*_{1,1} \le \overline{x}^*_{2,2} \le \ldots \le \overline{x}^*_{n,n}$.*

According to Assumption 1 and the convexity of the functions $f_i(x)$, it follows that $\overline{x}^*_{i,j-1} \le \overline{x}^*_{j,j} \le \overline{x}^*_{j,k-1}$ for any $1 \le i < j < k \le n+1$. This holds because $\overline{x}^*_{i,j-1}$ minimizes a sum of convex functions whose individual minimizers are all less than or equal to $\overline{x}^*_{j,j}$, while $\overline{x}^*_{j,k-1}$ minimizes a sum of convex functions with individual minimizers are all greater than or equal to $\overline{x}^*_{j,j}$. This property allows us to simplify the DP recursion by reducing the state space. Specifically, using this property and the

Algorithm 1 Efficient Implementation of DP Recursion (DP-original)

Input: $\overline{x}^*_{i,j}$'s, $c_{i,j}$'s , for (i,j) such that $1 \le i \le j \le n$
for $k = 2, \ldots, n+1$ **do**
    $V(1,k) \leftarrow c_{1,k-1}$
end for
for $j = 2, \ldots, n$ **do**
    Index $\overline{x}^*_{1,j-1}, \ldots, \overline{x}^*_{j-1,j-1}, \overline{x}^*_{j,j}, \ldots, \overline{x}^*_{j,n}$ to be $1, \ldots, j-1, j, \ldots, n$
    Sort the indexed entries in ascending order of their values. For equal values, place indices at least j before indices below j. Let $\pi(\ell)$ denote the index of the ℓ-th entry.
    $V_{\min}(0) \leftarrow \infty$
    for $\ell = 1, \ldots, n$ **do**
        if $\pi(\ell) \ge j$ **then**
            $V_{\min}(\ell) \leftarrow V_{\min}(\ell-1)$
            $V(j, \pi(\ell)+1) \leftarrow c_{j,\pi(\ell)} + V_{\min}(\ell)$
        else
            $V_{\min}(\ell) \leftarrow \min\{V(\pi(\ell), j), V_{\min}(\ell-1)\}$
        end if
    end for
end for
Output: $\min_{j \in [n]} V(j, n+1)$ and a solution towards it

observation that in the formulation (DP-original), the strict inequality $\overline{x}^*_{i,j-1} < \overline{x}^*_{j,k-1}$ can be replaced by a less than or equal to inequality, the value-to-go function $V(j,k)$ can be simplified as

$$V(j,k) = \begin{cases} c_{1,k-1}, & \text{if } 1 = j < k \le n+1, \\ c_{j,k-1} + \min_{i \in [j-1]} V(i,j), & \text{if } 2 \le j < k \le n+1. \end{cases}$$

Setting $\hat{V}(0) = 0$ and defining $\hat{V}(j-1) := \min_{i \in [j-1]} V(i,j)$ for each $j = 2, \ldots, n+1$, we can rewrite the formulation (DP-original) as

$$\hat{V}(j) = \begin{cases} 0, & j = 0, \\ \min_{i:1 \le i \le j} \{c_{i,j} + \hat{V}(i-1)\} & j \in [n]. \end{cases} \qquad \text{(DP-isotonic)}$$

This is equivalent to the DP recursion for finding the shortest (s,t)-path in the graph shown in Figure 2, where each node represents an index. The terminal cost $\hat{V}(n)$ corresponds to the optimal value of the Path problem.

A simple observation is that the new DP recursion (DP-isotonic) can be computed in $O(n^2)$ time, improving upon the $O(n^2 \log n)$ complexity from the previous subsection. Furthermore, we can show that the DP recursion (DP-isotonic) admits a significantly better $O(n)$ time complexity by leveraging the Monge property of the costs $c_{i,j}$'s.

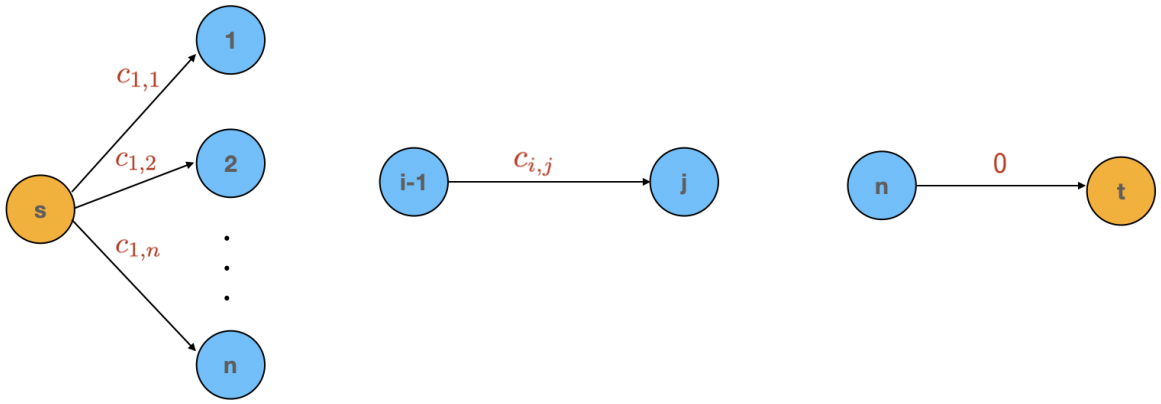


Figure 2 Illustration of the graph where finding the shortest (s,t)-path is equivalent to solving the DP recursion (DP-isotonic). Each index i with $1 \le i \le n$ is represented by a blue node. The only two yellow nodes are s and t, with arcs pointing out from s only towards $1,2,\ldots,n$ whose distances are $c_{1,1}, c_{1,2}, \ldots, c_{1,n}$, and arcs pointing to t only from n with distance 0. Between any two blue nodes $i-1$ and j with $i \le j$, an arc is directed from $i-1$ to j with distance $c_{i,j}$.

Definition 1 (Monge Property of a Matrix). An $m \times n$ matrix M is *Monge* if for any $i \in [m-1], j \in [n-1]$:

$$M_{i,j} + M_{i+1,j+1} \le M_{i+1,j} + M_{i,j+1};$$

i.e., for any $i < i' \in [m]$ and $j < j' \in [n]$, a Monge matrix M must satisfy $M_{ij} + M_{i',j'} \le M_{i',j} + M_{i,j'}$.

The Monge property, in fact, implies *total monotonicity*. The $m \times n$ matrix M is *monotone* if $i_M(j) \le i_M(j')$ for any $j < j'$ where $i_M(j)$ is the smallest row index of the minimum element in the jth column of the matrix M, and is *totally monotone* if every 2×2 submatrix of M is monotone. When $\boldsymbol{M}$ is Monge, we have $M_{i',j'} - M_{i,j'} \le M_{i',j} - M_{i,j}$ for any $i < i'$ and $j < j'$, which implies that if $M_{i',j} \le M_{i,j}$, then $M_{i',j'} \le M_{i,j'}$. That is, a Monge matrix must be totally monotone. For a general matrix M, finding all column minima requires $O(mn)$ time since finding the minimum of one column takes $O(m)$ time and there are n columns. When the matrix $\boldsymbol{M}$ is Monge and therefore totally monotone, Aggarwal et al. (1986) showed that the *SMAWK algorithm* can be used to find all column minima in $O(m+n)$ time. The SMAWK algorithm applies two reductions, named *REDUCE* and *INTERPOLATE*, recursively and alternatively, whose efficient implementations rely on the total monotonicity property. Readers can refer to the original paper for details of the algorithm.

We show that the Monge property can accelerate our DP algorithm. For each $i, j \in [n]$, let us define $\widetilde{c}_{i,j}$ as

$$\widetilde{c}_{i,j} = \begin{cases} c_{i,j} & i \le j, \\ \infty & i > j, \end{cases} \tag{4}$$

and $u_{i,j} = \widetilde{c}_{i,j} + \hat{V}(i-1)$. The DP formulation (DP-isotonic) can be equivalently transformed to be:

$$\hat{V}(j) = \begin{cases} 0, & j = 0 \\ \min_{i:1\le i\le j} u_{i,j}, & j \in [n]. \end{cases}$$

Once $\hat{V}(i-1)$ is available, evaluating $u_{i,j}$ for $i \le j$ requires one query to $c_{i,j}$ and $O(1)$ additional arithmetic operations. Specifically, at step j, the value $\hat{V}(i)$ is available if its index $i \le j$. Conversely, if $i > j$, the entry $u_{i,j}$ is set to ∞. Thus, computing the values $\hat{V}(j)$ is equivalent to recursively identifying all column minima of the matrix $\boldsymbol{U} = (u_{i,j})_{i\in[n],j\in[n]}$. In particular, if the matrix $\boldsymbol{U}$ satisfies the Monge property, and all the elements of $\boldsymbol{U}$ are known in advance, the SMAWK algorithm (Aggarwal et al. 1986) can be applied here to compute all column minima in $O(n)$ time. However, it is important to note that the original SMAWK (Aggarwal et al. 1986) algorithm is not directly applicable in our DP setting because the evaluation of $u_{i,j}$ depends on the prior computation of $\hat{V}(i-1)$ when $i \le j$. Nevertheless, we can still use the modified SMAWK algorithm (Galil and Park 1990) to evaluate the DP using $O(n)$ block-cost queries and $O(n)$ additional operations by guessing and correcting value-to-go functions along the way.

It remains to show that the matrix $\boldsymbol{U}$ is Monge, i.e., to prove that $u_{i,j} + u_{i+1,j+1} \le u_{i+1,j} + u_{i,j+1}$ for every $i \in [n-1], j \in [n-1]$, which is equivalent to

$$\begin{aligned} & \widetilde{c}_{i,j} + \hat{V}(i-1) + \widetilde{c}_{i+1,j+1} + \hat{V}(i) \le \widetilde{c}_{i+1,j} + \hat{V}(i) + \widetilde{c}_{i,j+1} + \hat{V}(i-1) \\ \iff \quad & \widetilde{c}_{i,j} + \widetilde{c}_{i+1,j+1} \le \widetilde{c}_{i+1,j} + \widetilde{c}_{i,j+1}. \end{aligned}$$

Hence, the matrix $\boldsymbol{U}$ is Monge if and only if the matrix $\widetilde{\boldsymbol{C}} = (\widetilde{c}_{i,j})_{i\in[n],j\in[n]}$ is. In fact, we show that the matrix $\widetilde{\boldsymbol{C}}$ indeed has Monge property under the isotonic minima assumption (i.e., Assumption 1 holds).

THEOREM 2. *Under Assumption 1, matrix $\widetilde{C} = (\widetilde{c}_{i,j})_{i\in[n],j\in[n]}$ defined by (4) satisfies the Monge property, i.e.,*

$$\widetilde{c}_{i,j} + \widetilde{c}_{i+1,j+1} \le \widetilde{c}_{i+1,j} + \widetilde{c}_{i,j+1}$$

for each $i \in [n-1], j \in [n-1]$.

When the matrix $\widetilde{C}$ satisfies the Monge property, the DP requires $O(n)$ operations and block-cost queries, as summarized below.

COROLLARY 1. *Suppose that Assumption 1 holds and that each $c_{i,j}$ and $\overline{x}^*_{i,j}$ can be queried in $O(1)$ time. Then, the (Path) problem can be solved using $O(n)$ arithmetic and comparison operations and $O(n)$ queries, excluding the preprocessing time.*

Hence, under Assumption 1, the overall complexity is $O(n)$ if $O(n)$ preprocessing supports constant-time queries for both block costs and block minimizers. Importantly, during the pre-processing, it is not necessary to compute all $c_{i,j}$ values explicitly. Instead, it suffices to preprocess the data such that each block cost and minimizer can be queried in $O(1)$ time on demand because the SMAWK algorithm only requires $O(n)$ such queries in total, while recovery uses at most n block-minimizer queries. The $O(n)$ pre-computation time holds for certain special cases, which will be discussed in the next subsection.

REMARK 1. The classical lot-sizing problem with backlog is a special case of our Path formulation and satisfies Assumption 1. Aggarwal and Park (1993) accelerated the naive $O(n^3)$ DP to $O(n \log n)$ via a divide-and-conquer method that exploits the Monge property, but their approach differs fundamentally from ours. Their DP recursion is formed by selecting the preceding ordering and zero-inventory epochs, without precomputing $x^*_{i,j}$'s or $c_{i,j}$'s, while our DP recursion is computed by selecting the last index at which an increase is made and needs to do pre-computing. Hence, the ideas behind the acceleration are also different. While our method incurs a higher overall complexity of $O(n^2 \log n)$ (comprising $O(n)$ for the DP recursion and $O(n^2 \log n)$ for precomputing arc costs as established in the second proposition of Proposition 2 in Section 2.3), it is conceptually simpler and easier to implement. Moreover, unlike their method, our framework extends naturally to broader settings, including stochastic lot-sizing.

2.3. Computation of $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s

In the previous subsections, we have discussed the time complexity of the DP algorithm for solving the Path problem and its acceleration under the assumption with isotonic minima of f_i's. However, the overall time complexity of the DP algorithm also depends on the computation of $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s. This subsection addresses this issue.

General Convex Search: Recall that for each $i < j \in [n]$, the value $\overline{x}^*_{i,j}$ is the minimum minimizer of $\sum_{\ell=i}^{j} f_\ell(x)$. Thus, computing $\overline{x}^*_{i,j}$ requires solving a convex minimization problem, which can be approximated by convex interval search using function-value queries. Without loss of generality, we assume that each function $f_i(x)$ has a bounded minimum within the interval $[\ell, u]$ and define $U = u - \ell$. Let $\epsilon \in (0, U)$ be the desired accuracy in the decision variable. Ternary search uses exact function values and, for the trisection points $a < b$ of the current interval $[L, R]$, retains $[L, b]$ if $\sum_{\ell=i}^{j} f_\ell(a) \le \sum_{\ell=i}^{j} f_\ell(b)$, and $[a, R]$ otherwise. This preserves the smallest minimizer and yields an interval of width at most ϵ containing it in $O(\log(U/\epsilon))$ steps. Each step evaluates a block sum at a constant number of points, using $O(n)$ node-cost queries and operations. Over all blocks, this requires $O(n^3 \log(U/\epsilon))$ queries and operations; evaluating the costs at the resulting approximate minimizers adds $O(n^3)$ operations. Logarithmic factors are understood to be at least one.

This is a numerical preprocessing bound: ϵ controls location accuracy, not the final DP objective error or exact ordering comparisons. The exact DP results require exact block minimizers and costs. Ahuja and Orlin (2001) used the separability structure to accelerate those computations, but their method can only be implemented under the framework of the PAV algorithm and cannot be applied to our setting. Although we may not be able to improve this complexity, we show that efficient implementations exist for the motivating examples mentioned at the beginning of the paper.

Efficient Computation for Motivating Examples: To efficiently compute $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s for the motivating examples, we first introduce a lemma that allows us to develop efficient computation when the functions f_i are quadratic or piecewise linear convex with only a single kinking point.

LEMMA 1. *To compute $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s defined in (3) for all pairs (i,j) such that $1 \le i \le j \le n$, we have*

(i) If $f_i(x) = p_i x^2 - 2q_i x + r_i$ with $p_i > 0$ for any $i \in [n]$, then the computation can be done in $O(n^2)$ time; and

(ii) If $f_i(x) = p_i(r_i - x)^+ + q_i(x - r_i)^+$ with $p_i, q_i > 0$ for any $i \in [n]$, then the computation can be done in $O(n^2 \log n)$ time.

According to Lemma 1, combined with the results in Proposition 1, we conclude that the overall DP algorithm can be implemented in $O(n^2 \log n)$ time when the functions f_i's are either quadratic or piecewise linear convex with a single kinking point.

REMARK 2. When the functions $\{f_i\}_{i\in[n]}$ have isotonic minima, leveraging the Monge property in Theorem 2 and the SMAWK algorithm, the DP algorithm can be implemented in $O(n)$ time, provided that preprocessing allows each $\overline{x}^*_{i,j}$ and $c_{i,j}$ to be queried in $O(1)$ time. Importantly, this approach eliminates the need to explicitly compute all $\overline{x}^*_{i,j}$ and $c_{i,j}$ values beforehand. For the case where f_i's are quadratic functions, as shown in the proof of Lemma 1, an $O(n)$ preprocessing time suffices to ensure that each $\overline{x}^*_{i,j}$ and $c_{i,j}$ can be queried in $O(1)$ time. Consequently, the overall computational complexity for this case is only $O(n)$.

We now show the connection between the two cases in Lemma 1 and the three motivating examples introduced in Section 1.1.

- For the generalized reduced isotonic optimization problem (GIR), the weighted absolute-error loss (i.e., $f_i(\widehat{y}_i) = w_i|\widehat{y}_i - y_i|$ for each $i \in [n]$) and the weighted squared-error loss (i.e., $f_i(\widehat{y}_i) = w_i(\widehat{y}_i - y_i)^2$ for each $i \in [n]$) are commonly used in the literature. These losses correspond directly to cases (ii) and (i), respectively, of Lemma 1.
- For the stochastic lot-sizing problem (SLS-O), consider the standard scenario tree model (Guan and Miller 2008) with S scenarios, where each scenario $s \in [S]$ occurs with probability p_s, has demand

$d_t(s)$ at time t, and cumulative demand $D_t(s)$ from time 1 to t. The cost function can be expressed as $f_t(X_t) = \sum_{s\in[S]} p_s(h_t(X_t - D_t(s))^+ + b_t(D_t(s) - X_t)^+)$. To ensure that each $f_t(X_t)$ can be queried in $O(1)$ time for any $X_t \in \mathbb{R}+$, we assume that $S = O(1)$ and that all $D_t(s)$ values are precomputed. Since each $f_t(X_t)$ is the summation of S piecewise linear convex functions, each with a single kinking point, we can modify the algorithm for Case (ii) in Lemma 1 by inserting S nodes into the augmented binary search tree at each step. This allows us to compute all $\overline{x}^*_{i,j}$ and $c_{i,j}$ values in $O(Sn^2 \log(Sn)) = O(n^2 \log n)$ since $S = O(1)$.

- For the joint inventory control and pricing problem with costly price markdown (CPM), when the demand functions are linear in price, the revenue function $R(\widetilde{d})$ is quadratic, which corresponds to Case (i) of Lemma 1. For (CPM), project each quadratic block minimizer from Lemma 1(i) onto $[0, D(0)]$ and evaluate its block cost at the projected point. This takes $O(1)$ additional time per block. These observations are summarized in the following proposition.

PROPOSITION 2. *To compute $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s defined by (3) for all (i,j) such that $1 \le i \le j \le n$, we have*

(GIR): for ℓ_1 and ℓ_2 generalized reduced isotonic regression, the computation requires $O(n^2 \log n)$ and $O(n^2)$ time, separately;

(SLS-O): for the stochastic lot sizing problem with $S = O(1)$ number of scenarios, the computation requires $O(n^2 \log n)$ time; and

(CPM): for the joint inventory control and pricing problem with costly price markdown, when the demand functions are linear in prices, the computation requires $O(n^2)$ time.

3. Solving the Arborescence Problem Part I: Two Special Cost Structures

This section extends the Path problem to the more general Arborescence problem. We present two special cases that can be solved in polynomial time. As in Section 2, we state the results in this section for $x_0 = -\infty$.

To begin with, we develop an optimality condition for the Arborescence problem, similar to that in Theorem 1. Recall that we define $\mathcal{T}(i)$ as the sub-arborescence of $\mathcal{T}$ rooted at the node i.

PROPOSITION 3. *There exists an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the Arborescence problem with the fewest strict increases and $y^*_i = \mathbb{1}\{x^*_i > x^*_{p(i)}\}$ for all $i \in [n]$. Let $M \subseteq [n]$ denote the support of $\boldsymbol{y}^*$. Then the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ partitions $\mathcal{T}$ into several induced arborescences $\mathcal{T}^i = (N_{\mathcal{T}^i}, A_{\mathcal{T}^i})$ for all $i \in M$, where each $\mathcal{T}^i$ is rooted at node i and $N_{\mathcal{T}^i} = N_{\mathcal{T}(i)} \backslash (\cup_{j\in N_{\mathcal{T}(i)}\cap M, j\neq i} N_{\mathcal{T}(j)})$, and $A_{\mathcal{T}^i}$ are arcs of $A_{\mathcal{T}}$ restricted on $N_{\mathcal{T}^i}$. Let $\overline{x}^*_{\mathcal{T}^i} = \min\{x^* | x^* \in \arg\min \sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(x)\}$ and $c_{\mathcal{T}^i} = \sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(\overline{x}^*_{\mathcal{T}^i})$. Such an optimal solution can be chosen to satisfy:*

(a) $x^*_\ell = \overline{x}^*_{\mathcal{T}^i}$ for all $\ell \in N_{\mathcal{T}^i}$, and

(b) $\overline{x}^*_{\mathcal{T}^i} < \overline{x}^*_{\mathcal{T}^j}$ *for all* $i \in M$ *and* $j \in (N_{\mathcal{T}(i)} \cap M) \backslash \{i\}$.

The implication of Proposition 3 is similar to that of Theorem 1, indicating that the optimal solution for (Arborescence) partitions $\mathcal{T}$ into several small arborescences, with all nodes in any single one share the same x_i value, and x_i values for any two adjacent ones follows a strict inequality relation. In the case of the path graph, the DP algorithm defines its states based on all possible blocks $B(i,j)$ for $1 \le i \le j \le n$, resulting in $O(n^2)$ blocks/states. To compute each value-to-go function, we only need to compare the values of $O(n)$ possible preceding blocks. However, in the case of the arborescence graph, each block is no longer a set of consecutive indices, but rather an induced arborescence of $\mathcal{T}$, which can be exponentially many in the worst case. A direct extension that enumerates all connected constant-value blocks may therefore have exponentially many states. Furthermore, computing each value-to-go function requires comparing the values of all preceding induced arborescences, which are also exponentially many in the worst case. Thus, even though Proposition 3 shows a structural similarity between the Arborescence problem and the Path problem, a DP algorithm similar to (DP-original) may be infeasible due to the curse of dimensionality. Therefore, we focus on exploring special cases where computationally efficient algorithms can be developed.

3.1. Special Case I: Piecewise Linear Convex Cost Functions

This subsection studies the case where f_i's are piecewise linear convex. Specifically, we assume that the following condition holds throughout this subsection.

Assumption 2. *For any* $i \in [n]$, f_i *is piecewise linear convex with* m_i *number of kink points* $\{k^i_1, \ldots, k^i_{m_i}\}$, *where* $k^i_1 < \ldots < k^i_{m_i}$ *with* $m_i = O(1)$ *such that each* $f_i(x_i)$ *can be queried in* $O(1)$ *time for any* x_i.

Many application problems, including problem (SLS-C), satisfy Assumption 2. Additionally, piecewise linear convex functions often serve as effective approximations for general convex functions.

When Assumption 2 holds, we can develop an efficient DP algorithm whose running time is polynomial in the total number of kink points across all f_i's. Specifically, the complexity depends on $\sum_{i\in[n]} m_i$, which is of order $O(n)$.

DP Recursion: From Proposition 3, we know that an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ for the Arborescence problem can partition $\mathcal{T}$ into several induced arborescences $\mathcal{T}^i = (N_{\mathcal{T}^i}, A_{\mathcal{T}^i})$ for all $i \in M$, where $\overline{x}^*_{\mathcal{T}^i} < \overline{x}^*_{\mathcal{T}^j}$ for all $i \in M$ and $j \in (N_{\mathcal{T}(i)} \cap M) \backslash \{i\}$, and $x^*_\ell \equiv \overline{x}^*_{\mathcal{T}^i}$ for all $\ell \in N_{\mathcal{T}^i}$ (recall that $\overline{x}^*_{\mathcal{T}^i} = \min\{x^* | x^* \in \arg\min \sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(x)\}$). Since f_i's are assumed to be piecewise linear, we know that $\sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(x)$ is also piecewise linear convex with kink points $\bigcup_{\ell\in N_{\mathcal{T}^i}} \{k^\ell_1, \ldots, k^\ell_{m_\ell}\}$. Hence, $\overline{x}^*_{\mathcal{T}^i}$ can be one of the points from $\bigcup_{\ell\in N_{\mathcal{T}^i}} \{k^\ell_1, \ldots, k^\ell_{m_\ell}\}$. Taking all possible induced arborescences $\mathcal{T}^i$ containing

node r into consideration, we know that $x_r^* \in \mathcal{K} = \bigcup_{\ell\in[n]}\{k_1^\ell, \ldots, k_{m_\ell}^\ell\}$, which is of $\sum_{\ell\in[n]} m_\ell = O(n)$ many, i.e., only linear in n.

Thus, this motivates us to design an efficient DP algorithm to optimally solve the Arborescence problem when f_i's are piecewise linear convex functions. For each $i \in [n]$ and $x \in [\ell, u]$, let the value-to-go function $V(i,x)$ be defined as follows:

$$\begin{aligned} V(i,x) = \min_{(\boldsymbol{x},\boldsymbol{y})\in\mathbb{R}^{|\mathcal{T}(i)|}\times\{0,1\}^{|\mathcal{T}(i)|}} & \sum_{\ell\in N_{\mathcal{T}(i)}} K_\ell y_\ell + \sum_{\ell\in N_{\mathcal{T}(i)}} f_\ell(x_\ell) \\ \text{s.t.} \quad & (x_{p(\ell)} - x_\ell)(1-y_\ell) = 0, \quad \forall \ell \in N_{\mathcal{T}(i)}, \\ & x_{p(\ell)} \le x_\ell, \quad \forall \ell \in N_{\mathcal{T}(i)}, \\ & x_{p(i)} = x, \end{aligned} \tag{5}$$

which denotes the minimum cost of the isotonic optimization restricted to the sub-arborescence $\mathcal{T}(i)$, given that $x_{p(i)} = x$. Without loss of generality, we only need to consider the value of x to be k, where k belongs to the set $\{-\infty\} \cup \mathcal{K}$, since at optimality, $x_{p(i)}^*$ must take a value from $\{-\infty\} \cup \mathcal{K}$. The value $-\infty$ is used only in the root state $(1, -\infty)$, where only the increase option is available; no node cost is evaluated at $-\infty$. At a nonroot node i with $x_{p(i)} = k$, we have two choices: (i) keep x_i unchanged, setting $x_i = x_{p(i)} = k$, or (ii) increase the value of x_i to one element in $\mathcal{K}$ that is strictly greater than k. Recall that $\mathcal{C}(i)$ is defined as the set of all child nodes of i. In the first case, the value-to-go function is

$$V_\rightarrow(i,k) = f_i(k) + \sum_{j\in\mathcal{C}(i)} V(j,k).$$

In the second case, where x_i is strictly increased, the value-to-go function is

$$V_\uparrow(i,k) = \min_{k'\in\mathcal{K}:k'>k} \left\{ K_i + f_i(k') + \sum_{j\in\mathcal{C}(i)} V(j,k') \right\}.$$

Hence, the DP recursion can be recast as:

$$V(i,k) = \begin{cases} V_\uparrow(1,-\infty), & i = 1, k = -\infty, \\ \min\{V_\rightarrow(i,k), V_\uparrow(i,k)\}, & \forall i \in [n]\backslash\{1\}, k \in \mathcal{K}. \end{cases} \tag{DP-PL}$$

Note that $V(1,-\infty)$ equals the optimal value of the Arborescence problem. We next compute the overall complexity of (DP-PL). For each state (i,k), we need to first compute $V_\rightarrow(i,k)$ using $O(|\mathcal{C}(i)|)$ operations. Then we should compute $V_\uparrow(i,k)$, which needs us to compare $O(\sum_{i\in[n]} m_i)$ possible decisions, and evaluating each decision requires $O(|\mathcal{C}(i)|)$ operations. Therefore, the total computation across all (i,k) pairs is $\sum_{k'\in\mathcal{K}}\sum_{i\in[n]} O(\sum_{i\in[n]} m_i|\mathcal{C}(i)|) = O(n(\sum_{i\in[n]} m_i)^2)$. It is at most $O(n^3)$ under the assumption that $m_i = O(1)$ for all $i \in [n]$. We can further accelerate the DP algorithm to run in $O(n^2)$ time by adopting a similar approach as in Algorithm 1.

The Efficient Implementation: To accelerate the DP algorithm, note that for each $i \in [n]$, in the DP recursion (DP-PL), the computation of $V_\uparrow(i,k)$ for different $k \in \mathcal{K}$ can be efficiently implemented. We first sort the kink points in $\mathcal{K}$ once in descending order. For each $i \in [n]$, by scanning through the sorted values and maintaining a running minimum over the scanned values, we can also calculate all $V(i,k)$'s together and save computation. Hence, the total time complexity is $O((n+\log(\sum_{i\in[n]} m_i))\sum_{i\in[n]} m_i)$. The detailed implementation can be found in Algorithm 2 in Appendix E.2. The result is summarized in the following proposition.

PROPOSITION 4. *Under Assumption 2, the Arborescence problem can be solved in* $O((n+\log(\sum_{i\in[n]} m_i))\sum_{i\in[n]} m_i)$ *time and* $O(n\sum_{i\in[n]} m_i)$ *auxiliary space by efficiently implementing* (DP-PL). *Both bounds are* $O(n^2)$ *if* $m_i = O(1)$ *for all* $i \in [n]$.

REMARK 3. Note that the problem (SLS-C) is a special case of the (Arborescence) problem, where each function f_i has only one kink point. For $X_0 = 0$, use the candidate set $(\mathcal{K} \cup \{0\}) \cap [0,\infty)$ and return $V(1,0)$, allowing the root to remain at zero without paying K_1. According to Proposition 4, the problem (SLS-C) can be efficiently solved using Algorithm 2 with a time complexity of $O(n^2)$. This result matches the best-known computational complexity in Guan (2011), but with a simpler algorithm and a more straightforward analysis.

REMARK 4. Proposition 4 also implies that for the Path problem, when the f_i's are piecewise linear with a constant number of kink points, the $O(n^2 \log n)$ complexity of Algorithm 1 can be further improved to $O(n^2)$ by adopting a more efficient alternative Algorithm 2.

3.2. Special Case II: General Convex Cost Functions with Dominating Fixed Costs in Parent Nodes

In this subsection, we study another special case of the Arborescence problem, where the fixed cost at each parent node is greater than or equal to the sum of all fixed costs in its child nodes. That is, we assume that

ASSUMPTION 3. *For each node* $i \in [n]$, *we have* $K_i \geq \sum_{j\in\mathcal{C}(i)} K_j$.

In (SLS-C), the condition that $K_i \geq K_j$ for any $i \in [n]$ and $j \in \mathcal{C}(i)$ is a generalization of the non-increasing fixed costs assumption in the path case, which is frequently made in inventory problems (Chen and Simchi-Levi 2004a,b). For scenario-tree-based models, we express the fixed ordering cost at each node i as $K_i = p_i k_i$, where k_i denotes the fixed ordering cost and p_i is the probability of reaching node i. Since $p_i = \sum_{j\in\mathcal{C}(i)} p_j$ for each nonleaf node i, Assumption 3 is equivalent to $k_i \geq \sum_{j\in\mathcal{C}(i)} (p_j/p_i) k_j$ whenever $p_i > 0$. To ensure that this condition holds for arbitrary scenario probabilities and branching structures, it is sufficient to impose non-increasing fixed ordering costs

along each path, i.e., $k_i \geq k_j$ for all $j \in \mathcal{C}(i)$. In particular, constant fixed ordering costs satisfy Assumption 3 with equality at every nonleaf node.

This subsection shows that under Assumption 3, the Arborescence problem can be solved in polynomial time by characterizing the value-to-go functions as piecewise convex functions, each consisting of a few pieces. For simplicity, we assume that $f_i(x_i) \to \infty$ as $|x_i| \to \infty$ throughout this subsection. This assumption ensures that the optimal solution remains bounded.

DP Formulation: Different from the previous special case, our value-to-go functions have continuous states; i.e., the state variable x in the function $V(i,x)$ in (5) is continuous. Therefore, the DP recursion can be expressed as:

$$\begin{aligned} V(i,x) &= \min_{x' \geq x} \left\{ K_i \mathbb{1}\{x' > x\} + U(i,x') \right\}, \\ U(i,x) &= f_i(x) + \sum_{j \in \mathcal{C}(i)} V(j,x) \end{aligned} \tag{6}$$

For a prescribed initial value x_0, the optimal value is $V(1,x_0)$, which becomes $V(1,-\infty)$ under our convention. For finite x_0, any interval $[\ell, u]$ used to restrict the computation should also contain x_0. Since the value-to-go function has a continuous state x, its computation is generally difficult. We will show that $V(i,x)$ has a well-structured characterization using the concept called K-convexity, under the assumption that $K_i \geq \sum_{j \in \mathcal{C}(i)} K_j$ for any $i \in [n]$. More specifically, we show that $V(i,x)$ is a piecewise convex function in x with at most $O(|N_{\mathcal{T}(i)}|)$ convex pieces, and that its representation can be computed efficiently.

The notion of K-convexity dates back to Scarf (1960), where it was used to demonstrate that the well-known (s,S) policy is optimal for a classical stochastic inventory problem with fixed costs. Formally, K-convexity is defined as follows:

Definition 2. A real-valued function f is called K-convex for $K \geq 0$ if, for any $x_0 \leq x_1$ and $\lambda \in [0,1]$,

$$f((1-\lambda)x_0 + \lambda x_1) \leq (1-\lambda) f(x_0) + \lambda f(x_1) + \lambda K.$$

Characterization of $V(i,x)$'s and the Optimal Policy: The following theorem characterizes the value-to-go function $V(i,x)$ as a piecewise convex function with at most $|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|$ number of convex pieces (recall that $\mathcal{L}(i)$ is the leaf node set of the subtree $\mathcal{T}(i)$), which is of order $O(|N_{\mathcal{T}(i)}|)$, by leveraging its K_i-convexity property, for each $i \in [n]$. Intuitively, the piecewise convexity arises because, at each node, the cost-to-go function is defined as the minimum of two piecewise convex functions: one corresponding to keeping the value unchanged, and the other to increasing it and incurring a fixed cost. Each option contributes a distinct convex segment to the overall function. As we propagate upward through the tree, the DP recursion creates more and more new convex pieces of value-to-go functions. The K_i-convexity ensures each $V(i,x)$ introduces at most one more convex

piece compared with $U(i,x)$. Consequently, the total number of convex pieces grows at most linearly with the number of nodes in the subtree.

THEOREM 3. *Under Assumption 3, for each $i \in [n]$, the value-to-go function $V(i,x)$ defined in (5) satisfies the following four properties:*

(a) $V(i,x)$ is K_i-convex;

(b) there exists a pair of real numbers (s_i, S_i) satisfying

$$S_i \in \arg\min U(i,x), \tag{7a}$$

$$s_i \in \left\{ x | x \le S_i, U(i,x) = U(i,S_i) + K_i \right\}, \tag{7b}$$

such that an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the Arborescence problem is

$$x_i^* = \begin{cases} S_i, & x_{p(i)}^* \le s_i, \\ x_{p(i)}^*, & o.w. \end{cases}, \quad y_i^* = \begin{cases} 1, & x_{p(i)}^* \le s_i, \\ 0, & o.w; \end{cases}$$

(c) $V(i,x)$ can be expressed as

$$V(i,x) = \begin{cases} K_i + U(i,S_i), & x \le s_i, \\ U(i,x), & o.w.; \end{cases}$$

(d) $V(i,x)$ is piecewise convex with at most $|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|$ number of convex pieces.

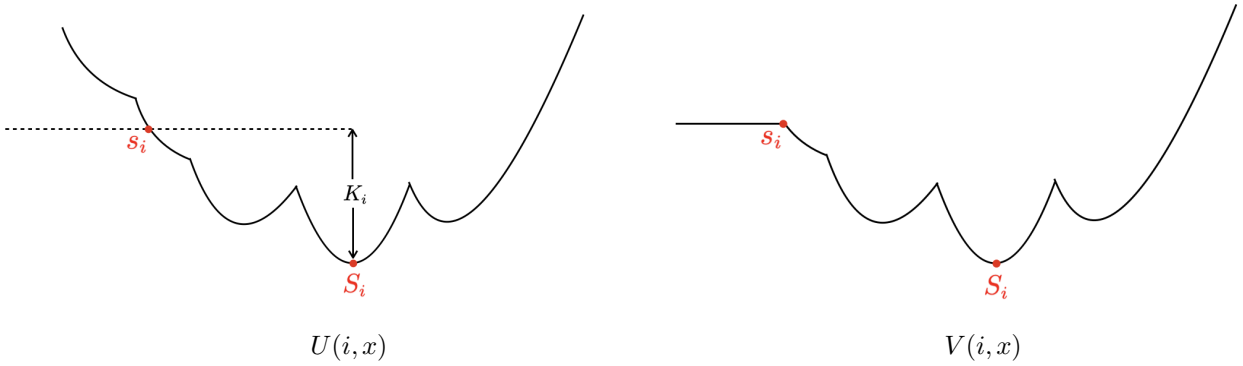


Figure 3 The changing dynamics from $U(i,x)$ to $V(i,x)$ when $K_i \ge \sum_{j \in \mathcal{C}(i)} K_j$

From Theorem 3(b), we know that computing an optimal solution to the Arborescence problem relies on determining the corresponding (s_i, S_i) thresholds. The resulting optimal policy resembles the classical (s_t, S_t) policy in stochastic inventory control (Scarf 1960): at each node i, if the value inherited from its parent is at or below s_i, set $x_i = S_i$; otherwise, leave it unchanged. We refer to this as the (s_i, S_i) policy. While such a policy is known to be optimal, computing the exact values of (s_i, S_i)'s is difficult in the traditional inventory setting due to the non-convexity of the value-to-go

functions $V(i,x)$. However, for our setting, leveraging Theorem 3(d), which shows that each $V(i,x)$ is a piecewise convex function with at most $O(|N_{\mathcal{T}(i)}|)$ pieces, we can efficiently obtain the expression of each $V(i,x)$. Consequently, the corresponding (s_i, S_i) values can also be computed efficiently.

REMARK 5. A related approach is proposed by Bhathena et al. (2025) for quadratic optimization with variable-activation indicators and a positive-definite Hessian whose sparsity graph is a tree. Their DP algorithm bounds the number of pieces in the value-to-go functions by exploiting Fenchel conjugation and the lower-envelope structure of strongly convex quadratic functions, without relying on K-convexity or dominating fixed costs. In contrast, our framework considers general convex node costs, isotonic constraints, and fixed costs associated with strict increases along edges. Under Assumption 3, the K_i-convexity property implies that $V(i,\cdot)$ can be obtained from $U(i,\cdot)$ by replacing its left tail with a constant piece, introducing at most one additional convex piece. Thus, both approaches yield a linear bound on the number of convex pieces in terms of subtree size, but rely on fundamentally different structural properties and DP transformations.

Algorithm and Complexity: We present an algorithm for obtaining the explicit functional representation of the value-to-go functions $V(i,x)$ and efficiently computing the optimal thresholds (s_i, S_i). According to Theorem 3(c), each $V(i,x)$ is a piecewise convex function, which allows for a compact representation through a structured list. Specifically, each entry of this list corresponds to one convex piece of $V(i,x)$ and includes relevant information about its functional form. Formally, we represent each function $V(i,x)$ through the following list structure:

$$\text{List}_i = [(-\infty, b_1^i, L_1^i, \mathcal{J}_1^i), (b_1^i, b_2^i, L_2^i, \mathcal{J}_2^i), \ldots, (b_{R_i-1}^i, \infty, L_{R_i}^i, \mathcal{J}_{R_i}^i)], \tag{8}$$

where the points $b_1^i \le b_2^i \le \ldots \le b_{R_i-1}^i$ represent the breakpoints that separate the convex segments of $V(i,x)$. We define the intervals $(-\infty, b_1^i], (b_1^i, b_2^i], \ldots, (b_{R_i-1}^i, \infty)$ to correspond directly to these convex segments. For notational convenience, we denote $b_0^i = -\infty$ and $b_{R_i}^i = \infty$. From Theorem 3(d), it follows that the number of convex pieces satisfies $R_i \le |N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|$. Additionally, each convex piece $r \in [R_i]$ is characterized by a scalar $L_r^i \in \mathbb{R}$ and an associated subset $\mathcal{J}_r^i \subseteq [n]$ which together completely describe the functional form of $V(i,x)$. Specifically, the value-to-go function $V(i,x)$ can be represented as follows:

$$V(i,x) = \begin{cases} L_1^i + \sum_{j\in\mathcal{J}_1^i} f_j(x), & x \in (-\infty, b_1^i], \\ L_2^i + \sum_{j\in\mathcal{J}_2^i} f_j(x), & x \in (b_1^i, b_2^i], \\ \cdots & \\ L_{R_i}^i + \sum_{j\in\mathcal{J}_{R_i}^i} f_j(x), & x \in (b_{R_i-1}^i, \infty). \end{cases} \tag{9}$$

We next introduce a proposition addressing three important aspects: (i) the validity of the representation provided in (9), (ii) an efficient method for constructing these piecewise convex representations

via DP recursion, and (iii) the computational complexity of obtaining the lists. A byproduct of the proof is that it directly provides an efficient algorithm for solving the Arborescence problem under the assumption $K_i \geq \sum_{j\in\mathcal{C}(i)} K_j$ for all $i \in N_{\mathcal{T}}$. Under the grid and exact-evaluation assumptions of Proposition 5, the thresholds (s_i, S_i) are recovered exactly, allowing reconstruction of an optimal solution. The DP constructs the list representation (8), restricted to $[\ell, u]$, for each $V(i,x)$ in $O(|N_{\mathcal{T}(i)}|^2 \log(U/\epsilon))$ time, yielding an overall complexity of $O(n^3 \log(U/\epsilon))$.

PROPOSITION 5. *Suppose that $K_i \geq \sum_{j\in\mathcal{C}(i)} K_j$ for all $i \in [n]$. Further assume that the thresholds S_i and s_i, defined in (7a) and (7b), respectively, can be chosen from a known uniform grid with spacing $\epsilon > 0$, and that all function evaluations and comparisons can be performed exactly. Then, the following statements hold:*

(a) There exists a list representation (8) for each $V(i,x)$, whose functional form can be recovered from the list representation according to (9).

(b) The optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ for the Arborescence problem can be obtained in $O(n^3 \log(U/\epsilon))$ time and $O(n^2)$ auxiliary space.

Discretization Error: Let $[\ell, u]$ contain a minimizer of every node cost, so that restricting decisions to this interval preserves optimality. Let L_i be a Lipschitz constant of f_i on $[\ell, u]$; such a constant exists because f_i is finite and convex on $\mathbb{R}$. Consider a common grid $\mathcal{G}_\epsilon \subseteq [\ell, u]$ containing both endpoints, with consecutive grid points at most ϵ apart. Let V_ϵ be the value functions obtained by solving the DP recursion (6) exactly with all decisions restricted to the same grid $\mathcal{G}_\epsilon$. For $x \in [\ell, u]$, let $q_\epsilon(x)$ be the smallest grid point at least x. This rounding changes each decision by at most ϵ, preserves order and equality, and therefore cannot introduce additional fixed charges. Define

$$e_i = \sup_{x\in[\ell,u]} \{V_\epsilon(i, q_\epsilon(x)) - V(i,x)\}.$$

To bound e_i, let $z \geq x$ be an optimal decision in the continuous DP at node i. Then $q_\epsilon(z)$ is feasible for the grid subproblem with inherited value $q_\epsilon(x)$, and

$$V_\epsilon(i, q_\epsilon(x)) \leq K_i \mathbb{1}\{q_\epsilon(z) > q_\epsilon(x)\} + f_i(q_\epsilon(z)) + \sum_{j\in\mathcal{C}(i)} V_\epsilon(j, q_\epsilon(z)) \leq V(i,x) + L_i\epsilon + \sum_{j\in\mathcal{C}(i)} e_j.$$

Thus each DP update adds at most $L_i\epsilon$ to the errors from its children. Induction gives

$$e_i \leq L_i\epsilon + \sum_{j\in\mathcal{C}(i)} e_j \leq \epsilon \sum_{k\in N_{\mathcal{T}(i)}} L_k.$$

Let $C_\epsilon = V_\epsilon(1, -\infty)$ be the actual objective value of the solution recovered from the grid DP. Applying the same comparison at the initial value $-\infty$, which is left unchanged, and using feasibility of the recovered solution yields

$$0 \leq C_\epsilon - \mathrm{OPT} \leq \epsilon \sum_{i=1}^{n} L_i.$$

In particular, if $L_i \leq L$ for all i, the optimality gap is at most $nL\epsilon$. Hence discretization errors accumulate additively, with each node contributing at most $L_i\epsilon$.

4. Solving the Arborescence Problem Part II: Special Arborescence Structures

This section establishes polynomial bounds on the number of convex pieces for arborescences of logarithmic height or bounded Horton–Strahler number. A bounded Horton–Strahler number means that the tree can be eliminated in a constant number of rounds by removing one root-to-leaf path from each remaining subtree in every round; a formal definition is given below. The computational implications are specified in Section 4.1. We first describe the general technique and briefly treat the logarithmic-height case, then analyze arborescences with bounded Horton–Strahler number. As in Section 2, we state the results in this section for $x_0 = -\infty$.

4.1. General Techniques

We first describe the general technique used in this section. Our main strategy is to demonstrate that the value-to-go function $V(i,x)$ specified by (5) is a piecewise convex function for each node $i \in N_{\mathcal{T}}$. A polynomial bound on the number of pieces implies that the algorithm requires only polynomially many scalar convex minimizations, level searches, and comparisons. A level search finds where a convex piece reaches a given value, thereby locating a boundary of a new flat piece. An exact polynomial-time implementation further requires that these scalar computations can be performed exactly in polynomial time under the chosen representation of the cost functions. Each piece is stored as a constant plus a sum of node costs, as in (9). Algorithm 3 in Appendix E.3 provides pseudocode for this algorithm. Let C_i denote the number of pieces of $V(i,\cdot)$. Storing all lists with explicit index sets requires $O(\sum_{i\in[n]} |N_{\mathcal{T}(i)}|C_i)$ stored indices and scalar quantities. Thus, the polynomial piece bounds derived below also imply polynomial space complexity, provided that all stored numerical quantities admit exact representations of polynomial bit length.

Lemma 2. *The value-to-go function $V(i,x)$ is piecewise convex for any $i \in N_{\mathcal{T}}$, regardless of the arborescence structure.*

The piecewise convexity of $V(i,x)$ comes from an inductive structure. The function $U(i,x) = f_i(x) + \sum_{j\in\mathcal{C}(i)} V(j,x)$ is piecewise convex whenever all child value functions are piecewise convex, since f_i is convex. The fixed-cost update (6) then constructs $V(i,x)$ from $U(i,x)$ by introducing flat segments while preserving piecewise convexity. Repeating this from the leaves up to the root proves the claim by induction. We implement the general arborescence DP (6) using the same list representation and update procedures similar to those developed for the dominating fixed costs case in Section 3.2. The computation proceeds from the leaves to the root. At each node i, we merge the child lists and add f_i to construct $U(i,\cdot)$. We then minimize each convex piece over the closure of its interval and scan the resulting minimizers from right to left, using scalar level searches to identify the new flat pieces of $V(i,\cdot)$. For each interval in the resulting list, we record a decision rule that either

retains the value inherited from the parent or increases it to a specified value. If both alternatives attain the same total cost, we choose to retain the inherited value. After selecting an optimal decision at the root, we recover the remaining decisions by traversing the arborescence from the root to the leaves. At each nonroot node i, we locate the interval in List_i containing the parent's decision $x^*_{p(i)}$. If the decision recorded for this interval is to retain the parent's value, we set $x^*_i = x^*_{p(i)}$; otherwise, we set x^*_i to the larger value recorded for that interval. Having shown that each value-to-go function $V(i,x)$ is piecewise convex, our next goal is to bound the number of convex pieces for each $V(i,x)$ by poly(n). In Section 3.2, we showed that under the assumption $K_i \geq \sum_{j\in\mathcal{C}(i)} K_j$ for each $i \in N_{\mathcal{T}}$, the number of convex pieces for each $V(i,x)$ is bounded by $|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)| = O(n)$, where we recall that $\mathcal{L}(i)$ is the set of leaf nodes in $\mathcal{T}(i)$. In this section, we extend our analysis and demonstrate that, even without this assumption, the number of convex pieces for $V(i,x)$ remains bounded by poly(n) for several classes of arborescence graphs.

We now introduce the procedure for upper-bounding the number of convex pieces of $V(i,x)$ according to its DP recursion (6), assuming that each child node's value-to-go function $V(j,x)$, for $j \in \mathcal{C}(i)$, has at most C_j convex pieces:

- **Summing piecewise convex functions:** We first analyze the number of convex pieces in $U(i,x)$. Given that each $V(j,x)$ has at most C_j convex pieces, the number of breakpoints separating different convex segments in each $V(j,x)$ is $C_j - 1$. Since the set of breakpoints for $\sum_{j\in\mathcal{C}(i)} V(j,x)$ is the union of breakpoints from each $V(j,x)$ for all $j \in \mathcal{C}(i)$, the total number of breakpoints in $\sum_{j\in\mathcal{C}(i)} V(j,x)$ is at most $\sum_{j\in\mathcal{C}(i)}(C_j - 1)$. Consequently, $\sum_{j\in\mathcal{C}(i)} V(j,x)$ has at most $\sum_{j\in\mathcal{C}(i)}(C_j - 1) + 1 \leq \sum_{j\in\mathcal{C}(i)} C_j$ convex pieces. Additionally, since $f_i(x)$ is convex, the function $U(i,x) = f_i(x) + \sum_{j\in\mathcal{C}(i)} V(j,x)$ is also piecewise convex, having at most $\sum_{j\in\mathcal{C}(i)} C_j$ convex pieces.
- **Cutting out new convex pieces:** Next, we analyze the number of convex pieces in $V(i,x)$. From the proof of Lemma 2, we observe that $V(i,x)$ is obtained from $U(i,x)$ by introducing new flat segments. These flat segments correspond to intervals in which increasing x is preferable to keeping it unchanged, leading to the addition of new convex pieces. To bound the total number of convex pieces of $V(i,x)$, we must restrict the number of these newly introduced flat segments. Globally, the number of new flat segments is at most the number of convex pieces, so the number of convex pieces can at most double.

As an immediate consequence, arborescences of logarithmic height have polynomially many convex pieces in their value functions. Indeed, let C_i be the number of convex pieces of $V(i,\cdot)$ and $h(i) = \max_{j\in\mathcal{L}(i)} |\mathcal{P}(i,j)|$. The preceding bounds give $C_i \leq 2$ when i is a leaf node and $C_i \leq 2\sum_{j\in\mathcal{C}(i)} C_j$ otherwise. Induction then yields $C_i \leq 2^{h(i)}|\mathcal{L}(i)|$, which is polynomial in n when the tree height is $O(\log n)$. To handle arborescences with bounded Horton–Strahler number even when their height is large, we next refine the count of newly introduced pieces. We call a function $f : [a,b] \to \mathbb{R}$ *weakly*

unimodal if there exists $x^* \in [a,b]$ such that f is non-increasing on $[a,x^*]$ and non-decreasing on $[x^*,b]$; x^* is called a *mode* of f.

DEFINITION 3. Suppose $g:\mathbb{R}\to\mathbb{R}$ is a piecewise convex function with convex pieces $(-\infty,c_1]$, $(c_1,c_2],\ldots,(c_{L-1},\infty)$. Define the quasi-convex pieces of g as $(-\infty,\widetilde{c}_1]$, $(\widetilde{c}_1,\widetilde{c}_2],\ldots,(\widetilde{c}_{R-1},\infty)$, which should satisfy the following requirements:

(a) $\{\widetilde{c}_1,\widetilde{c}_2,\ldots,\widetilde{c}_{R-1}\}\subseteq\{c_1,c_2,\ldots,c_{L-1}\}$;

(b) g is weakly unimodal in each quasi-convex piece; and

(c) g is no longer weakly unimodal in the union of any two consecutive quasi-convex pieces.

From Definition 3(a), we know that for any piecewise convex function g, each quasi-convex piece is a union of several consecutive convex segments. Definition 3(b) implies that within each quasi-convex piece, the function g is non-increasing to the left and non-decreasing to the right of some local minimum (mode). Moreover, Definition 3(c) ensures minimality in the construction of quasi-convex pieces: any combination of two consecutive quasi-convex pieces cannot form a larger quasi-convex piece. With these properties, we now present the following lemma.

LEMMA 3. *Suppose $g:\mathbb{R}\to\mathbb{R}$ is continuous and piecewise convex, with L convex pieces and $R\le L$ quasi-convex pieces. Assume that g attains a minimum at a finite point on the closure of each quasi-convex piece. Then, for any $K\in\mathbb{R}_+$, the function $h:\mathbb{R}\to\mathbb{R}$ defined by*

$$h(x)=\min_{x'\ge x}\left\{K\mathbb{1}\{x'>x\}+g(x')\right\}$$

is piecewise convex with at most $L+R$ number of convex pieces.

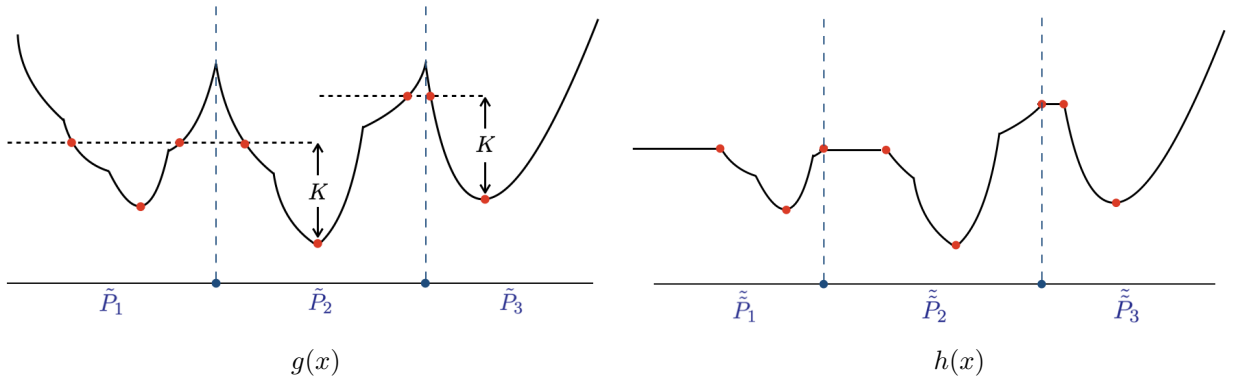


Figure 4 Illustration of Lemma 3, where $\widetilde{P}_1,\widetilde{P}_2,\widetilde{P}_3$ are quasi-convex pieces of $g(x)$ and $\widetilde{\widetilde{P}}_1,\widetilde{\widetilde{P}}_2,\widetilde{\widetilde{P}}_3$ are quasi-convex pieces of $h(x)$

These conditions hold for $g=U(i,\cdot)$: the DP update preserves continuity, and each convex piece is a constant plus a sum of node costs. Since every node cost has a finite minimizer, each such sum attains

a minimum on every nonempty closed interval. Readers can refer to Figure 4 for a visualization of how newly flat pieces of $h(x)$ are introduced from $g(x)$. Lemma 3 shows that the update from g to h introduces at most R additional flat pieces, where R is the number of quasi-convex pieces of g. With the definitions and lemmas introduced above, we now bound the number of convex pieces of each $V(i,x)$ by poly(n) for several special classes of arborescence structures.

4.2. Arborescences with Bounded Horton–Strahler Number

In this subsection, we establish polynomial bounds on the number of convex pieces in the DP value functions for arborescences with bounded Horton–Strahler number. Intuitively, the Horton–Strahler number is the minimum number of rounds needed to remove all nodes for the arborescence by deleting one root-to-leaf path from each remaining arborescence in every round, which we will formally define later. To obtain these bounds, we refine the count of newly introduced pieces by analyzing all local minima of $U(i,\cdot)$ together.

LEMMA 4. *For any $i \in N_{\mathcal{T}}$ and any convex piece P_ℓ^i with $\ell \in [L_i]$ in the DP representation of $V(i,\cdot)$, there are a constant $K_\ell^i \in \mathbb{R}$ and a set $\mathcal{D}_\ell^i \subseteq N_{\mathcal{T}(i)}$ such that*

$$V(i,x) = K_\ell^i + \sum_{j \in \mathcal{N}_\ell^i} f_j(x), \qquad \forall x \in P_\ell^i,$$

where $\mathcal{N}_\ell^i = \bigcup_{k \in \mathcal{D}_\ell^i} \mathcal{P}(i,k)$, which might be an empty set. The same representation holds for each convex piece in the DP representation of $U(i,\cdot)$, with $i \in \mathcal{N}_\ell^i$.

The representation in Lemma 4 alone does not ensure that a local minimum at a breakpoint minimizes $\sum_{j\in\mathcal{N}} f_j(x)$ over all $x \in \mathbb{R}$. The following lemma shows that every local minimizer of $U(i,\cdot)$ globally minimizes $\sum_{j\in\mathcal{N}^i} f_j(x)$ for some set $\mathcal{N}^i$ of nodes forming a subtree rooted at i.

LEMMA 5. *For any $i \in [n]$ and any local minimizer m of $U(i,\cdot)$, there exists a nonempty set $\mathcal{N}^i = \bigcup_{k\in\mathcal{D}^i} \mathcal{P}(i,k)$, where $\mathcal{D}^i \subseteq N_{\mathcal{T}(i)}$, such that $m \in \arg\min_{x\in\mathbb{R}} \sum_{j\in\mathcal{N}^i} f_j(x)$.*

We use the following spine bound to count these minima without enumerating all rooted subtrees.

LEMMA 6. *Let $\mathcal{P} := \mathcal{P}(1,j)$ for some $j \in \mathcal{L}$, and re-index its nodes from the root to the leaf as $1,2,\ldots,|\mathcal{P}|$. For each $\ell \in (\bigcup_{k\in\mathcal{P}} \mathcal{C}(k)) \setminus \mathcal{P}$, let m_ℓ denote the number of convex pieces of $V(\ell,\cdot)$. Then, for every $i \in \mathcal{P}$, the number of convex pieces of $V(i,\cdot)$ is at most*

$$\left(1 + \frac{n(n+1)}{2}\right)\left(1 + \sum_{\ell \in (\bigcup_{k\in\mathcal{P}} \mathcal{C}(k))\setminus\mathcal{P}} m_\ell\right).$$

We call the path $\mathcal{P}$ the spine, its nodes spine nodes, and their children outside $\mathcal{P}$ off-spine children. The above lemma bounds the number of convex pieces at each spine node by one plus the total number of pieces in the off-spine child value functions, multiplied by $1+n(n+1)/2$. Such a factor arises as follows. First assume that the original node cost functions f_k for $k \in \mathcal{P}$ are strictly convex and coercive. Before evaluating $V(i,x)$, group states by the most recent spine node at which a flat piece replaced their values as the DP moves toward the root. Each resulting region may contain several disjoint intervals. Points that have not been cut form one additional region. Here, a cut includes only points where the flat piece strictly lowers the function value. There are at most $|\mathcal{P}|-i+1$ such regions. Within a region associated with node j, $U(i,x)$ consists of the flat value introduced at j plus the costs added along spine nodes $j-1,\dots,i$, including their off-spine value functions. Although the flat value may differ between intervals, this added sum is the same throughout the region. Its number of convex pieces is at most $\max\{1, \sum_{\ell \in (\cup_{k\in\mathcal{P}} \mathcal{C}(k))\setminus\mathcal{P}} m_\ell\}$. Every local minimum of $U(i,\cdot)$ in the region must also locally minimize this sum; otherwise, changing the common value of the corresponding spine decisions and reoptimizing the attached subproblems would contradict local minimality. Strict convexity ensures that each convex piece accounts for at most one such minimum. Thus, as we sweep along the spine, each region contributes at most this many newly introduced flat pieces per level. Summing over the spine counts at most $|\mathcal{P}|(|\mathcal{P}|+1)/2$ pairs of a node and a region. Combining this count with Lemma 3 and the inherited off-spine breakpoints yields the factor $1+|\mathcal{P}|(|\mathcal{P}|+1)/2 \le 1+n(n+1)/2$. For general convex costs, we add εx^2, with $\varepsilon > 0$, to each original node cost f_k on the spine and let $\varepsilon \downarrow 0$; the uniform bound on the number of convex pieces is preserved in the limit. This result is crucial for bounding the number of convex pieces in arborescences with bounded Horton–Strahler number, which we define below.

Before proceeding, we note that Lemma 6 extends to additional useful settings. For each $i \in \mathcal{P}$, one can regard $f_i(x) + \sum_{\ell \in \mathcal{C}(i)\setminus\mathcal{P}} V(\ell, x)$ as a single piecewise-convex function. The same counting argument applies to the Path problem with piecewise-convex stage costs. Indeed, every consecutive sum of stage costs has at most $1+\sum_{j=1}^{n}(m_j-1)$ convex pieces, since f_j contributes at most m_j-1 breakpoints. Adding εx^2 to each f_j, with $\varepsilon > 0$, makes each convex piece strictly convex and each stage cost coercive. The argument in Lemma 6 then introduces at most $(n-i+1)(1+\sum_{j=1}^{n}(m_j-1))$ new flat pieces at stage i. Adding these counts and the inherited breakpoints gives the stated bound. As $\varepsilon \downarrow 0$, finite attainment gives pointwise convergence, as in Lemma 6. Convexity on each surviving interval follows by passing the convexity inequality to the limit, preserving the bound.

COROLLARY 2. *Consider the Path problem with continuous piecewise convex costs f_i, each having m_i convex pieces. Assume that all DP subproblems have finite, attained minima. Then, for every $i \in [n]$, the value function $V(i,\cdot)$ has the number of convex pieces at most*

$$\left(1+\frac{n(n+1)}{2}\right)\left(1+\sum_{j=1}^{n}(m_j-1)\right).$$

We measure the branching structure of an arborescence by its Horton–Strahler number (Biedl 2022). Intuitively, it is the minimum number of rounds needed to remove all nodes by deleting one root-to-leaf path from each remaining tree in each round.

DEFINITION 4. If i is a leaf node, set $\mathrm{hs}(i)=1$. For a nonleaf node i, let $a_i=\max_{j\in\mathcal{C}(i)}\mathrm{hs}(j)$. Set $\mathrm{hs}(i)=a_i$ if exactly one child attains this maximum, and $\mathrm{hs}(i)=a_i+1$ otherwise. The Horton–Strahler number of $\mathcal{T}$ is $\mathrm{hs}(1)$.

If $\mathrm{hs}(1)=k$, following a child of maximum Horton–Strahler number at each step gives a root-to-leaf path whose removal leaves only subtrees with number at most $k-1$. For example, a directed path has Horton–Strahler number 1. A star with at least three leaves has number 2 regardless of its root, while a caterpillar (a tree in which every node lies on or is adjacent to a central path) rooted at an endpoint of its central path has number at most 2. These three examples are illustrated in Figure 5.

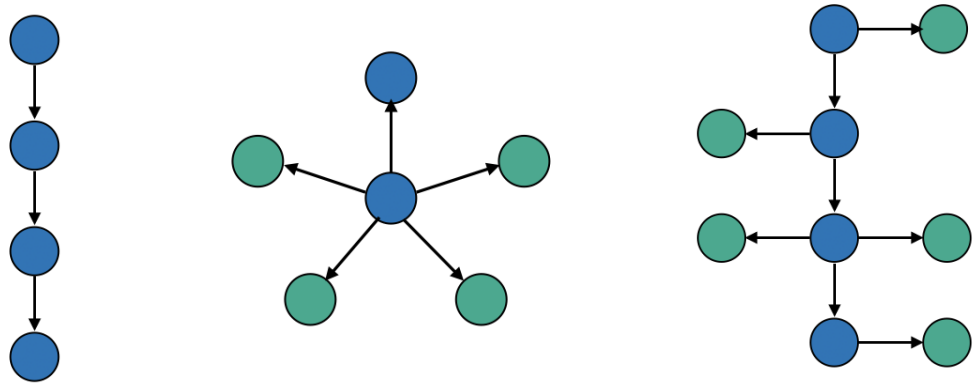


Figure 5 Examples of arborescences with Horton–Strahler number at most two. Blue marks a selected path; green marks the remaining nodes.

Applying Lemma 6 recursively to this path decomposition gives the following result.

THEOREM 4. *If $\mathcal{T}$ has Horton–Strahler number at most $k\in\mathbb{Z}_+$, then, for every $i\in[n]$, the number of convex pieces of $V(i,\cdot)$ is at most $(1+n(n+1)/2)^k=O(n^{2k})$.*

The following corollary follows directly from Theorem 4.

COROLLARY 3. *If $\mathcal{T}$ has bounded Horton–Strahler number, then the number of convex pieces of $V(i,x)$ for each $i \in N_{\mathcal{T}}$ is* poly(n).

5. General Arborescences: Tight Bounds on Convex Pieces

It is currently unknown whether the general Arborescence problem is NP-hard. In this section, we study the number of convex pieces in the value functions generated by the parametric DP developed in the preceding sections. We establish tight quasi-polynomial bounds of $n^{\Theta(\log n)}$ on this number in the worst case, demonstrating the superpolynomial size of these explicit representations even for convex, continuously differentiable piecewise-quadratic node costs. As in Section 4, we take $x_0 = -\infty$ and count only the n cost-bearing nodes.

We first observe that an arborescence of Horton–Strahler number k contains at least $2^k - 1$ nodes. This follows by induction on the number of nodes, rather than on k: each child subtree has fewer nodes, even when its Horton–Strahler number is also k. The bound is immediate for a leaf. At a nonleaf node of number k, if a unique child attains the largest number, that child also has number k, and its subtree already contains at least $2^k - 1$ nodes by induction. Otherwise, at least two children have number $k-1$, so their subtrees and the parent contain at least $1 + 2(2^{k-1} - 1) = 2^k - 1$ nodes. Consequently, $\mathrm{hs}(1) \le \lfloor \log_2(n+1) \rfloor$. Combining this observation with Theorem 4 gives the following upper bound on the number of convex pieces of the value functions for general arborescences.

COROLLARY 4. *For an arborescence with n nodes, each value function $V(i,\cdot)$ for $i \in [n]$ admits a representation with at most $(1 + n(n+1)/2)^{\lfloor \log_2(n+1) \rfloor} = n^{O(\log n)}$ convex pieces.*

We next establish a matching lower bound by constructing a family of instances whose root value functions have quasi-polynomially many convex pieces. The idea is to make the number of local minima grow much faster than the tree size. Starting from a two-node tree, each step combines two copies of the current tree with an added path. Suitably chosen costs create many new minima between existing ones, simultaneously across the root function $U(1,\cdot)$. The tree size quadruples, while the number of selected minima is multiplied by at least the previous tree size. Repeating this construction yields quasi-polynomially many minima. The final DP update turns these minima into distinct boundaries between convex pieces of $V(1,\cdot)$.

PROPOSITION 6. *For every integer $r \ge 0$, there exists an instance on an arborescence with at most two children per node and $n_r = 2 \cdot 4^r$ nodes such that the root value function $V(1,\cdot)$ requires at least $2^{r^2+1} + 1 = n_r^{\Omega(\log n_r)}$ convex interval pieces.*

Combining Corollary 4 and Proposition 6 gives the following tight bound.

THEOREM 5. *Over instances with at most n nodes, the worst-case minimum number of convex interval pieces needed to represent a DP value function $V(i,\cdot)$ is $n^{\Theta(\log n)}$.*

6. Numerical Results

Experimental Setup: To evaluate the computational performance of proposed DP algorithms, we evaluate three classes of problem instances: (i) quadratic cost functions with path structures, (ii) quadratic cost functions with general arborescence structures, and (iii) piecewise linear convex cost functions with general arborescence structures. We use the big-M formulation solved by Gurobi as the baseline. The standard perspective reformulation is not a natural alternative, even for quadratic costs: an inactive indicator imposes $x_i = x_{p(i)}$ rather than $x_i = 0$, and $f_i(x_i)$ is incurred regardless of the indicator. Specifically, for each $i \in [n]$, we linearized the bilinear constraint $(x_i - x_{p(i)})(1 - y_i) = 0$ using the big-M coefficient M_i as $x_i - x_{p(i)} \le M_i y_i$, where M_i is selected as $\max_{i\in[n]} x^*_{i,i} - \min_{i\in[n]} x^*_{i,i}$, i.e., the range of the node-cost minimizers. We set the computational time limit for Gurobi as 30 minutes. For each class of instances, we implement the algorithm with the best theoretical time complexity developed in this paper and compare it against Gurobi. All the experiments are conducted on a MacBook Pro with an Apple M2 Chip and 8 GB RAM. The Gurobi solver we use is of version 12.0.0 build v12.0.0rc1. In both the DP and Gurobi implementations, the quadratic experiments use $x_0 = -\infty$, so the root charge K_1 is always incurred. The piecewise-linear SLS-C experiments use $X_0 = 0$, so K_1 is incurred only when $X_1 > 0$. The detailed experimental setups are described below:

- *Quadratic & path*: We compare two algorithms in this paper with the baseline: the DP without acceleration given by (DP-original) (with time complexity $O(n^3)$) here, which we denote by DP_0; and the DP with acceleration given by Algorithm 1 (with time complexity $O(n^2 \log n)$, the best algorithm under this setting in the paper), which we denote by DP_1. We test for $n \in \{500, 1000, 1500, 2000, 2500, 3000\}$, and for any particular n, we generate 10 random instances as follows: $K_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 60)$; $f_i(x) = a_i x^2 - 2b_i x$, in which a_i's form a decreasingly ordered sequence of n random numbers generated from $\text{Unif}(0, 20)$, and $b_i = \widehat{b}_i + \epsilon_i$ where $\widehat{b}_i$'s form an increasingly ordered sequence of n random numbers generated from $\text{Unif}(0, 20)$ and $\epsilon_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(-5, 0)$. We generate a_i, b_i in such a way because the minimum of $f_i(x)$ is attained at b_i/a_i, which is isotonic concerning i if $\epsilon_i = 0$. ϵ_i are added to prevent b_i/a_i from being isotonic, but make them "almost" isotonic. For Gurobi, the average running time and the average gap are reported; for DP_0 and DP_1, the average running time is reported, both in Table 1.

- *Quadratic & arborescence*: We compare the DP algorithm given by Section 3.2 and Section 4 with the baseline. It is implemented by tracking the functional forms of piecewise convex value-to-go functions. Although Section 5 shows quasi-polynomial growth in the worst case, we also examine the number of convex pieces on randomly generated arborescences. We test for $n \in \{500, 1000, 1500, 2000, 2500, 3000\}$, and for any particular n, we generate 10 random instances as follows: the arborescence is generated by uniformly assigning node i as a child of one of $1, \ldots, i-1$;

$K_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 60)$; $f_i(x) = a_i x^2 - 2b_i x$, in which a_i's form a decreasingly ordered sequence of n random numbers generated from $\text{Unif}(0, 20)$, and b_i's form an increasingly ordered sequence of n random numbers generated from $\text{Unif}(0, 20)$ (here we do not need to add noises on b_i to make b_i/a_i "almost" isotonic). For Gurobi, the average running time and the average gap are reported; for DP, the average running time and the average maximum number of convex pieces among all value-to-go functions are reported, both in Table 2.

REMARK 6. Ordered or nearly ordered node-cost minimizers do not eliminate the fixed-cost tradeoff: choosing the individual minimizers may incur fixed charges, whereas assigning a common value to adjacent nodes avoids fixed charges on the corresponding edges but may increase node costs by moving decisions away from their individual minimizers. Moreover, for an increase $0 < \delta < M_i$, the continuous relaxation of the big-M formulation permits a fixed-cost contribution of only $K_i\delta/M_i$, rather than the full fixed cost K_i. This can weaken the lower bound used to certify optimality. Tables 1–2 report the running times and remaining optimality gaps for these instance families.

- *Piece-wise Linear & arborescence*: We compare Algorithm 2, whose $O(n^2)$ time complexity follows from Proposition 4, with the big-M baseline. It is the algorithm with the best complexity designed under this setting for both the path graph and the arborescence graph. To conduct a unified experiment, we only test for the arborescence case given the page limit. We use synthetic instances of the piecewise-linear lot-sizing formulation given by (SLS-C). We test for $n \in \{500, 1000, 1500, 2000, 2500, 3000\}$, and for any particular n, we generate 10 random instances as follows: $K_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 60)$; $h_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 10)$; $b_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 10)$; $d_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(5, 20)$. For Gurobi, the average running time and the average gap are reported; for the DP algorithm, the average running time is reported, both in Table 3.

Table 1 Quadratic & Path

	Gurobi		DP_0	DP_1
n	time (s)	gap (%)	time (s)	time (s)
500	1269.24	0.02	10.99	0.67
1000	1800.00	0.08	87.53	2.68
1500	1800.00	0.08	301.63	7.17
2000	1800.00	0.08	695.95	10.62
2500	1800.00	0.08	1389.40	16.90
3000	1800.00	0.06	1800.00+	24.13

Implications: The numerical results support the following observations.

Table 2 Quadratic & Arborescence

	Gurobi		DP	
n	time (s)	gap (%)	time (s)	Max. number of convex pieces
500	1800.00	5.20	0.19	164.50
1000	1800.00	4.66	0.54	310.90
1500	1800.00	4.93	1.37	465.80
2000	1800.00	5.10	2.02	623.40
2500	1800.00	5.89	3.01	773.10
3000	1800.00	6.04	4.07	921.20

Table 3 Piecewise Linear & Arborescence

	Gurobi		DP
n	time (s)	gap (%)	time (s)
500	156.88	0.00	1.04
1000	1800.00	0.53	4.27
1500	1800.00	0.66	9.70
2000	1800.00	1.16	17.15
2500	1800.00	1.22	27.80
3000	1800.00	1.24	40.64

- *Quadratic & path*: For $n \geq 1000$, Gurobi's reported average running time is 1800 seconds, with a small positive average optimality gap. DP_0 does not finish within the 30-minute limit at $n = 3000$, whereas DP_1 completes all reported sizes. When comparing the two DP algorithms, DP_1 consistently outperforms DP_0, with the advantage becoming increasingly significant as the instance size n grows– achieving up to nearly a hundredfold reduction in runtime. The observed speedup is consistent with the improved complexity bound of Algorithm 1.
- *Quadratic & arborescence*: Gurobi's reported average running time is 1800 seconds at every tested size, with positive average optimality gaps. In contrast, the DP algorithm averages under five seconds across all tested sizes. Part of this efficiency stems from the quadratic cost structure of our problem setup, which allows for the explicit analytical computation of breakpoints rather than relying on binary search. The remarkable computational advantage is also attributed to the limited growth of convex segments in the value-to-go functions. Although the worst-case number of convex pieces grows quasi-polynomially, Table 2 shows approximately linear growth in the average maximum piece count for this random instance family. The small representations and explicit breakpoint calculations help explain the observed running times.

- *Piecewise linear & arborescence*: For $n \geq 1000$, Gurobi's reported average running time is 1800 seconds, and its average optimality gap increases with the problem size. In contrast, the proposed DP algorithm has average running times ranging from 1.04 to 40.64 seconds. The observed growth is roughly quadratic in n, consistent with Proposition 4.

Appendix F presents two additional numerical studies. The first examines how the number of convex pieces varies with the Horton–Strahler number k. The second evaluates the algorithms on open-loop and closed-loop stochastic lot-sizing instances constructed from historical sales data, corresponding to path and arborescence structures, respectively.

7. Conclusions and Future Direction

This paper presented a general framework for isotonic optimization problems incorporating fixed costs. By generalizing conventional isotonic optimization to include general convex cost functions and enforcing isotonic constraints, we addressed a broader class of practical optimization problems. For path structures, we developed an efficient dynamic programming (DP) algorithm and demonstrated additional computational improvements by leveraging inherent structural properties. For arborescence graph structures, we illustrated that despite increased complexity, the problem remains solvable in polynomial time under specific conditions. These conditions include scenarios with piecewise linear convex cost functions, cases with dominant fixed costs in parent nodes, and particular graph structures such as arborescences of logarithmic height or bounded Horton–Strahler number. For general arborescences, we established tight worst-case quasi-polynomial bounds on the number of convex pieces required to represent the parametric DP value-to-go functions. On the tested instance families, the accelerated path DP and the tree DP implementations have lower average running times than the big-M baselines. The experiments also illustrate how value-function piece counts vary with tree structure. Future research could explore whether isotonic optimization with fixed costs on general arborescences is polynomially solvable. Extending the framework to directed acyclic graphs, particularly those whose underlying undirected graphs have bounded treewidth, would allow modeling a wider range of applications. Multiple parents and shared descendants prevent a direct extension of our decomposition into independent child subproblems. A possible approach is to apply DP over a tree decomposition, using the decisions at the boundary nodes connecting each subproblem to the rest of the graph as the state. Developing tractable representations of the resulting multivariate value functions remains an open challenge, even when the number of boundary nodes is bounded by a constant independent of the problem size.

References

Aggarwal A, Klawe M, Moran S, Shor P, Wilber R (1986) Geometric applications of a matrix searching algorithm. *Proceedings of the second annual symposium on Computational geometry*, 285–292.

Aggarwal A, Park JK (1993) Improved algorithms for economic lot size problems. *Operations research* 41(3):549–571.

Ahuja RK, Hochbaum DS, Orlin JB (2003) Solving the convex cost integer dual network flow problem. *Management Science* 49(7):950–964.

Ahuja RK, Orlin JB (2001) A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Operations Research* 49(5):784–789.

Bacchetti P (1989) Additive isotonic models. *Journal of the American Statistical Association* 84(405):289–294.

Barlow R, Ubhaya V (1971) Isotonic approximation. *Optimizing Methods in Statistics*, 77–86 (Elsevier).

Barlow RE, Brunk HD (1972) The isotonic regression problem and its dual. *Journal of the American Statistical Association* 67(337):140–147.

Best MJ, Chakravarti N (1990) Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming* 47(1):425–439.

Best MJ, Chakravarti N, Ubhaya VA (2000) Minimizing separable convex functions subject to simple chain constraints. *SIAM Journal on Optimization* 10(3):658–672.

Bhathena A, Fattahi S, Gómez A, Küçükyavuz S (2025) A parametric approach for solving convex quadratic optimization with indicators over trees. *Mathematical Programming* 1–46.

Biedl T (2022) Horton-strahler number, rooted pathwidth and upward drawings of trees. *Information Processing Letters* 175:106230.

Chakravarti N (1989) Isotonic median regression: a linear programming approach. *Mathematics of operations research* 14(2):303–308.

Chen X, Hu P (2012) Joint pricing and inventory management with deterministic demand and costly price adjustment. *Operations Research Letters* 40(5):385–389.

Chen X, Li X, Su Y (2024) An active-set-based recursive approach for solving convex isotonic regression with generalized order restrictions. *Asia-Pacific Journal of Operational Research* 41(3):2350025.

Chen X, Simchi-Levi D (2004a) Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finite horizon case. *Operations research* 52(6):887–896.

Chen X, Simchi-Levi D (2004b) Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The infinite horizon case. *Mathematics of operations Research* 29(3):698–723.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2022) *Introduction to algorithms* (MIT press).

Crowston WB, Wagner M, Williams JF (1973) Economic lot size determination in multi-stage assembly systems. *Management Science* 19(5):517–527.

Dykstra RL (1981) An isotonic regression algorithm. *Journal of Statistical Planning and Inference* 5(4):355–363.

Galil Z, Park K (1990) A linear-time algorithm for concave one-dimensional dynamic programming. *Information Processing Letters* 33(6):309–311, URL http://dx.doi.org/10.1016/0020-0190(90)90215-J.

Guan Y (2011) Stochastic lot-sizing with backlogging: computational complexity analysis. *Journal of Global Optimization* 49(4):651–678.

Guan Y, Miller AJ (2008) Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research* 56(5):1172–1183.

Haiminen N, Gionis A, Laasonen K (2008) Algorithms for unimodal segmentation with applications to unimodality detection. *Knowledge and information systems* 14:39–57.

Hardwick J, Stout QF (2014) Optimal reduced isotonic regression. arXiv:1412.2844.

Hochbaum DS (2001) An efficient algorithm for image segmentation, markov random fields and related problems. *Journal of the ACM (JACM)* 48(4):686–701.

Hochbaum DS (2004) 50th anniversary article: Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today. *Management Science* 50(6):709–723.

Huangfu Q, Hall JAJ (2018) Parallelizing the dual revised simplex method. *Mathematical Programming Computation* 10(1):119–142, URL http://dx.doi.org/10.1007/s12532-017-0130-5.

Kaufman Y, Tamir A (1993) Locating service centers with precedence constraints. *Discrete Applied Mathematics* 47(3):251–261.

Levi R, Pál M, Roundy RO, Shmoys DB (2007) Approximation algorithms for stochastic inventory control models. *Mathematics of Operations Research* 32(2):284–302.

Love SF (1972) A facilities in series inventory model with nested schedules. *Management Science* 18(5-part-1):327–338.

Maxwell WL, Muckstadt JA (1985) Establishing consistent and realistic reorder intervals in production-distribution systems. *Operations Research* 33(6):1316–1341.

Niculescu-Mizil A, Caruana R (2005) Predicting good probabilities with supervised learning. *Proceedings of the 22nd international conference on Machine learning*, 625–632.

Pardalos PM, Xue G (1999) Algorithms for a class of isotonic regression problems. *Algorithmica* 23:211–222.

Robertson T, Wright F (1973) Multiple isotonic median regression. *The Annals of Statistics* 1(3):422–432.

Robertson T, Wright FT (1975) Consistency in generalized isotonic regression. *The Annals of Statistics* 3(2):350–362, URL http://dx.doi.org/10.1214/aos/1176343061.

Salanti G, Ulm K (2003) A nonparametric changepoint model for stratifying continuous variables under order restrictions and binary outcome. *Statistical methods in medical research* 12(4):351–367.

Scarf H (1960) The optimality of (S, s) policies in the dynamic inventory problem. Arrow KJ, Karlin S, Suppes P, eds., *Mathematical Methods in the Social Sciences, 1959*, 196–202 (Stanford, CA: Stanford University Press).

Schell MJ, Singh B (1997) The reduced monotonic regression method. *Journal of the American Statistical Association* 92(437):128–135.

Schwarz LB, Schrage L (1975) Optimal and system myopic policies for multi-echelon production/inventory assembly systems. *Management Science* 21(11):1285–1294.

Simchi-Levi D, Chen X, Bramel J (2014) The logic of logistics: Theory, algorithms, and applications for logistics management. *Springer Series in Operations Research and Financial Engineering*, 1–447 (Springer).

Stout QF (2008) Unimodal regression via prefix isotonic regression. *Computational Statistics & Data Analysis* 53(2):289–297.

Stout QF (2013) Isotonic regression via partitioning. *Algorithmica* 66:93–112.

Stout QF (2014) An algorithm for L_∞ approximation by step functions. arXiv:1412.2379.

Stout QF (2015) L_∞ isotonic regression for linear, multidimensional, and tree orders. arXiv:1507.02226.

Stout QF (2018) Weighted L_∞ isotonic regression. *Journal of Computer and System Sciences* 91:69–81.

Stout QF (2019) Fastest known isotonic regression algorithms .

Stout QF (2021a) L_0 isotonic regression with secondary objectives. arXiv:2106.00279.

Stout QF (2021b) L_p isotonic regression algorithms using an L_0 approach. arXiv:2107.00251.

Stout QF (2023) Best L_p isotonic regressions, $p \in \{0, 1, \infty\}$. arXiv:2306.00269.

Tamir A (1993) The least element property of center location on tree networks with applications to distance and precedence constrained problems. *Mathematical programming* 62:475–496.

Tan J (2014) Sales Transactions Weekly. UCI Machine Learning Repository, URL http://dx.doi.org/10.24432/C5XS4Q, dataset.

Tempelmeier H (2013) Stochastic lot sizing problems. *Handbook of stochastic models and analysis of manufacturing system operations*, 313–344 (Springer).

Tibshirani RJ, Hoefling H, Tibshirani R (2011) Nearly-isotonic regression. *Technometrics* 53(1):54–61.

Vargas V (2009) An optimal solution for the stochastic version of the wagner–whitin dynamic lot-size model. *European Journal of Operational Research* 198(2):447–451.

Wang X, Ying J, Cardoso JVdM, Palomar DP (2022) Efficient algorithms for general isotone optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8575–8583.

Yu Z, Chen X, Li X (2023) A dynamic programming approach for generalized nearly isotonic optimization. *Mathematical Programming Computation* 15(1):195–225.

Online Appendix: Isotonic Optimization with Fixed Costs

Appendix A: Proofs of Technical Results in Section 2

A.1. Proof of Theorem 1

THEOREM 1. *Suppose* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *is an optimal solution for the Path problem chosen with* $y_i^* = \mathbb{1}\{x_i^* > x_{i-1}^*\}$ *for all* $i \in [n]$*, such that* $y_i^* = 1$ *if* $i \in \{i_0, i_1, i_2, \ldots, i_R\} \subseteq [n]$ *where* $1 = i_0 < i_1 < i_2 < \ldots < i_R < i_{R+1} = n+1$ *and* $y_i^* = 0$ *otherwise. Then*

(a) $x_i^* = \overline{x}^*_{i_{r-1}, i_r - 1}$ *for all* $i \in B(i_{r-1}, i_r - 1)$*; and*

(b) The strict inequality $\overline{x}^*_{i_{r-1}, i_r - 1} < \overline{x}^*_{i_r, i_{r+1}-1}$ *must hold for any* $r \in [R]$.

Proof. Indexes $i_1, \ldots, i_R$ partition $[n]$ into blocks $B(1, i_1 - 1), B(i_1, i_2 - 1) \ldots, B(i_R, n)$ where $B(i, j) = \{i, \ldots, j\}$. Since x_i's in each block share the same value, we let $x_i^* = \widetilde{x}_{i_{r-1}, i_r - 1}$ for any $i \in B(i_{r-1}, i_r - 1)$. Hence, the cost corresponding to block $B(i_{r-1}, i_r - 1)$ equals $K_{i_{r-1}} + \sum_{\ell \in B(i_{r-1}, i_r - 1)} f_\ell(\widetilde{x}_{i_{r-1}, i_r - 1})$. If we can prove that $\widetilde{x}_{i_{r-1}, i_r - 1} = \overline{x}^*_{i_{r-1}, i_r - 1}$, the proof is done. Assume by contradiction that for some $r \in [R+1]$, $\widetilde{x}_{i_{r-1}, i_r - 1} \neq \overline{x}^*_{i_{r-1}, i_r - 1}$. Two cases need to be discussed.

Case 1: $\boldsymbol{\widetilde{x}_{i_{r-1}, i_r - 1} > \overline{x}^*_{i_{r-1}, i_r - 1}}$. We decrease $\widetilde{x}_{i_{r-1}, i_r - 1}$ until one of the following two different cases happens: if $r = 1$ or $r > 1$ and $\widetilde{x}_{i_{r-2}, i_{r-1} - 1} \leq \overline{x}^*_{i_{r-1}, i_r - 1}$, then we decrease $\widetilde{x}_{i_{r-1}, i_r - 1}$ to be $\overline{x}^*_{i_{r-1}, i_r - 1}$ without violating the isotonic constraint between block $B(i_{r-2}, i_{r-1} - 1)$ and $B(i_{r-1}, i_r - 1)$. Since $\overline{x}^*_{i_{r-1}, i_r - 1}$ minimizes $\sum_{\ell \in B(i_{r-1}, i_r - 1)} f_\ell(x)$, we know that the cost corresponding to block $B(i_{r-1}, i_r - 1)$ decreases or at least holds the same, contradicting the optimality of $\boldsymbol{x}^*$; if $r > 1$ and $\widetilde{x}_{i_{r-2}, i_{r-1} - 1} > \overline{x}^*_{i_{r-1}, i_r - 1}$, then we decrease $\widetilde{x}_{i_{r-1}, i_r - 1}$ to be $\widetilde{x}_{i_{r-2}, i_{r-1} - 1}$. Therefore, block $B(i_{r-2}, i_{r-1} - 1)$ and $B(i_{r-1}, i_r - 1)$ are merged together to be a single block $B(i_{r-2}, i_r - 1)$. From the convexity of $\sum_{\ell = i_{r-1}}^{i_r - 1} f_\ell(x)$, we know that it is nondecreasing in $[\overline{x}^*_{i_{r-1}, i_r - 1}, \infty)$, and by moving its evaluated point from $\widetilde{x}_{i_{r-1}, i_r - 1}$ towards its minimizer $\overline{x}^*_{i_{r-1}, i_r - 1}$ (and make it to be $\widetilde{x}_{i_{r-2}, i_{r-1} - 1}$), its value decreases or at least holds the same. Besides, by merging block $B(i_{r-2}, i_{r-1} - 1)$ and $B(i_{r-1}, i_r - 1)$, the fixed cost $K_{i_{r-1}}$ is saved. Hence, the overall cost decreases or holds the same, contradicting the optimality of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$.

Case 2: $\boldsymbol{\widetilde{x}_{i_{r-1}, i_r - 1} < \overline{x}^*_{i_{r-1}, i_r - 1}}$. We increase $\widetilde{x}_{i_{r-1}, i_r - 1}$ until one of the following two different cases happens: if $r = R+1$ or $r < R+1$ and $\widetilde{x}_{i_r, i_{r+1} - 1} \geq \overline{x}^*_{i_{r-1}, i_r - 1}$, then we increase $\widetilde{x}_{i_{r-1}, i_r - 1}$ to be $\overline{x}^*_{i_{r-1}, i_r - 1}$ without violating the isotonic constraint between block $B(i_{r-1}, i_r - 1)$ and $B(i_r, i_{r+1} - 1)$. Since $\overline{x}^*_{i_{r-1}, i_r - 1}$ minimizes $\sum_{\ell = i_{r-1}}^{i_r - 1} f_\ell(x)$, we know that the cost corresponding to block $B(i_{r-1}, i_r - 1)$ decreases or at least holds the same, contradicting the optimality of $\boldsymbol{x}^*$; if $r < R+1$ and $\widetilde{x}_{i_r, i_{r+1} - 1} < \overline{x}^*_{i_{r-1}, i_r - 1}$, then we increase $\widetilde{x}_{i_{r-1}, i_r - 1}$ to be $\widetilde{x}_{i_r, i_{r+1} - 1}$. Therefore, block $B(i_{r-1}, i_r - 1)$ and $B(i_r, i_{r+1} - 1)$ are merged together to be a single block $B(i_{r-1}, i_{r+1} - 1)$. From the convexity of $\sum_{\ell = i_{r-1}}^{i_r - 1} f_\ell(x)$, we

know that it is nonincreasing in $(-\infty, \overline{x}^*_{i_{r-1},i_r-1}]$, and by moving its evaluated point from $\widetilde{x}_{i_{r-1},i_r-1}$ towards its minimizer $\overline{x}^*_{i_{r-1},i_r-1}$ (and make it to be $\widetilde{x}_{i_r,i_{r+1}-1}$), its value decreases or at least holds the same. Besides, by merging block $B(i_{r-1}, i_r-1)$ and $B(i_r, i_{r+1}-1)$, the fixed cost K_{i_r} is saved. Hence, the overall cost decreases or holds the same, contradicting the optimality of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. □

A.2. Proof of Proposition 1

PROPOSITION 1. *Suppose that $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s are precomputed. Then, the DP recursion* (DP-original) *can be efficiently implemented using Algorithm 1 in $O(n^2 \log n)$ time.*

Proof. We first show the correctness of the algorithm. Line 2 to line 4 computes $V(j,k)$ for the case when $1 = j < k \le n+1$, according to the recursion (DP-original). In the for loop from line 5 to line 16, for each $j = 2, \ldots, n$, we first sort those $\overline{x}^*_{1,j-1}, \ldots, \overline{x}^*_{j-1,j-1}, \overline{x}^*_{j,j}, \ldots, \overline{x}^*_{j,n}$ and then compute the value-to-go functions $V(j, j+1), \ldots, V(j, n+1)$ progressively according to the sorted list, such that we do not need to go over the list over and over again to determine each of $V(j, j+1), \ldots, V(j, n+1)$ and some computation is saved. The key is that we use $V_{\min}(\ell)$ to record the minimum $V(i,j)$ from the beginning up to the current position in the sorted list, which is kept updated in the loop. Hence, at the ℓ-th position of the sorted list, if $\pi(\ell) < j$, then $V_{\min}(\ell)$ is updated to be $\min\{V(\pi(\ell), j), V_{\min}(\ell-1)\}$; else if $\pi(\ell) \ge j$, from (DP-original), we update $V_{\min}(\ell) = V_{\min}(\ell-1)$ and $V(j, \pi(\ell)+1) = c_{j,\pi(\ell)} + V_{\min}(\ell)$. This computes $V(j,k)$ for the case when $2 \le j < k \le n+1$. Hence, we prove the correctness of Algorithm 1.

Next, we derive the time complexity of Algorithm 1. The for loop from line 2 to line 4 requires $O(n)$ time; the for loop from line 5 to line 16 has $O(n)$ iterations, and in each iteration, line 6 can be implemented in $O(n)$ time, the sorting in line 7 takes $O(n \log n)$ time, and the for loop from line 9 to line 15 takes $O(n)$ time. Hence, the overall dynamic programming can be implemented in $O(n^2 \log n)$ time. □

A.3. Proof of Theorem 2

We begin with a technical lemma.

LEMMA EC.1. *Suppose that three convex functions $g_1(y), g_2(y), g_3(y)$ satisfy $-\infty < \overline{y}^*_1 \le \overline{y}^*_2 \le \overline{y}^*_3 < \infty$, where $\overline{y}^*_i = \min\{y^* | y^* \in \arg\min g_i(y)\}$ for $i \in [3]$. Then the following inequality must hold:*

$$\min_y \Big\{ g_1(y) + g_2(y) + g_3(y) \Big\} + \min_y \Big\{ g_2(y) \Big\} \ge \min_y \Big\{ g_1(y) + g_2(y) \Big\} + \min_y \Big\{ g_2(y) + g_3(y) \Big\}.$$

Proof. Other than $\overline{y}_1^*, \overline{y}_2^*, \overline{y}_3^*$ defined in the statement of the lemma, let us define some additional minima:

$$\overline{y}_{12}^* = \min\left\{ y^* \middle| y^* \in \arg\min\left\{ g_1(y) + g_2(y) \right\} \right\},$$

$$\overline{y}_{23}^* = \min\left\{ y^* \middle| y^* \in \arg\min\left\{ g_2(y) + g_3(y) \right\} \right\},$$

$$\overline{y}_{123}^* = \min\left\{ y^* \middle| y^* \in \arg\min\left\{ g_1(y) + g_2(y) + g_3(y) \right\} \right\}.$$

It can be easily seen that $\overline{y}_1^* \le \overline{y}_{12}^* \le \overline{y}_2^* \le \overline{y}_{23}^* \le \overline{y}_3^*$ and $\overline{y}_{12}^* \le \overline{y}_{123}^* \le \overline{y}_{23}^*$ due to the convexity of $g_1(y), g_2(y), g_3(y)$ and the assumption $-\infty < \overline{y}_1^* \le \overline{y}_2^* \le \overline{y}_3^* < \infty$. Two cases are remaining to be discussed: $\overline{y}_{123}^* \le \overline{y}_2^*$ and $\overline{y}_{123}^* > \overline{y}_2^*$.

Case 1: $\overline{y}_{123}^* \le \overline{y}_2^*$. In this case, we know that

$$\begin{aligned}
&\min_y \left\{ g_1(y) + g_2(y) + g_3(y) \right\} + \min_y \left\{ g_2(y) \right\} - \min_y \left\{ g_1(y) + g_2(y) \right\} - \min_y \left\{ g_2(y) + g_3(y) \right\} \\
=& g_1(\overline{y}_{123}^*) + g_2(\overline{y}_{123}^*) + g_3(\overline{y}_{123}^*) + g_2(\overline{y}_2^*) - g_1(\overline{y}_{12}^*) - g_2(\overline{y}_{12}^*) - g_2(\overline{y}_{23}^*) - g_3(\overline{y}_{23}^*) \\
\ge& g_1(\overline{y}_{123}^*) + g_2(\overline{y}_{123}^*) + g_3(\overline{y}_{123}^*) + g_2(\overline{y}_2^*) - g_1(\overline{y}_{123}^*) - g_2(\overline{y}_{123}^*) - g_2(\overline{y}_2^*) - g_3(\overline{y}_2^*) \\
=& g_3(\overline{y}_{123}^*) - g_3(\overline{y}_2^*) \\
\ge& 0,
\end{aligned}$$

where the first inequality holds because $g_1(\overline{y}_{12}^*) + g_2(\overline{y}_{12}^*) = \min_y\{g_1(y) + g_2(y)\} \le g_1(\overline{y}_{123}^*) + g_2(\overline{y}_{123}^*)$, and $g_2(\overline{y}_{23}^*) + g_3(\overline{y}_{23}^*) = \min_y\{g_2(y) + g_3(y)\} \le g_2(\overline{y}_2^*) + g_3(\overline{y}_2^*)$, and the second inequality follows from the convexity of $g_3(y)$ and $\overline{y}_{123}^* \le \overline{y}_2^* \le \overline{y}_3^*$. Therefore, the inequality in the lemma holds.

Case 2: $\overline{y}_{123}^* > \overline{y}_2^*$. In this case, we have that

$$\begin{aligned}
&\min_y \left\{ g_1(y) + g_2(y) + g_3(y) \right\} + \min_y \left\{ g_2(y) \right\} - \min_y \left\{ g_1(y) + g_2(y) \right\} - \min_y \left\{ g_2(y) + g_3(y) \right\} \\
=& g_1(\overline{y}_{123}^*) + g_2(\overline{y}_{123}^*) + g_3(\overline{y}_{123}^*) + g_2(\overline{y}_2^*) - g_1(\overline{y}_{12}^*) - g_2(\overline{y}_{12}^*) - g_2(\overline{y}_{23}^*) - g_3(\overline{y}_{23}^*) \\
=& \underbrace{\left[g_1(\overline{y}_2^*) + g_2(\overline{y}_2^*) - g_1(\overline{y}_{12}^*) - g_2(\overline{y}_{12}^*) \right]}_{\ge 0} + \underbrace{\left[g_2(\overline{y}_{123}^*) + g_3(\overline{y}_{123}^*) - g_2(\overline{y}_{23}^*) - g_3(\overline{y}_{23}^*) \right]}_{\ge 0} + \underbrace{\left[g_1(\overline{y}_{123}^*) - g_1(\overline{y}_2^*) \right]}_{\ge 0} \\
\ge& 0,
\end{aligned}$$

where the second equality holds by adding $g_1(\overline{y}_2^*)$ and subtracting $g_1(\overline{y}_2^*)$ simultaneously from the last step, and after grouping, we know that the expression in the first curly bracket is nonnegative because $g_1(\overline{y}_{12}^*) + g_2(\overline{y}_{12}^*) = \min_y\{g_1(y) + g_2(y)\} \le g_1(\overline{y}_2^*) + g_2(\overline{y}_2^*)$, the expression in the second curly bracket is nonnegative because $g_2(\overline{y}_{23}^*) + g_3(\overline{y}_{23}^*) = \min_y\{g_2(y) + g_3(y)\} \le g_2(\overline{y}_{123}^*) + g_3(\overline{y}_{123}^*)$, and the expression in the third curly bracket is nonnegative because of the convexity of $g_1(y)$ and the fact that $\overline{y}_{123}^* > \overline{y}_2^* \ge \overline{y}_1^*$. Therefore, the inequality in the lemma holds. □

Based on Lemma EC.1, we can prove Theorem 2.

THEOREM 2. *Under Assumption 1, matrix* $\widetilde{C} = (\widetilde{c}_{i,j})_{i\in[n],j\in[n]}$ *defined by* (4) *satisfies the Monge property, i.e.,*

$$\widetilde{c}_{i,j} + \widetilde{c}_{i+1,j+1} \le \widetilde{c}_{i+1,j} + \widetilde{c}_{i,j+1}$$

for each $i \in [n-1], j \in [n-1]$.

Proof. If $i \ge j$, then $\widetilde{c}_{i+1,j} = \infty$ and the inequality of the Monge condition is automatically satisfied as its RHS is ∞. Hence, we only consider the case when $i \le j-1$. In this case,

$$\begin{aligned}
&\widetilde{c}_{i,j} + \widetilde{c}_{i+1,j+1} \le \widetilde{c}_{i+1,j} + \widetilde{c}_{i,j+1} \\
\iff\quad & K_i + \min_x \left\{\sum_{\ell=i}^{j} f_\ell(x)\right\} + K_{i+1} + \min_x \left\{\sum_{\ell=i+1}^{j+1} f_\ell(x)\right\} \le K_{i+1} + \min_x \left\{\sum_{\ell=i+1}^{j} f_l(x)\right\} + K_i + \min_x \left\{\sum_{\ell=i}^{j+1} f_\ell(x)\right\} \\
\iff\quad & \min_x \left\{\sum_{\ell=i}^{j} f_\ell(x)\right\} + \min_x \left\{\sum_{\ell=i+1}^{j+1} f_\ell(x)\right\} \le \min_x \left\{\sum_{\ell=i+1}^{j} f_\ell(x)\right\} + \min_x \left\{\sum_{\ell=i}^{j+1} f_\ell(x)\right\} \\
\iff\quad & \min_x \left\{f_i(x) + \sum_{\ell=i+1}^{j} f_\ell(x)\right\} + \min_x \left\{\sum_{\ell=i+1}^{j} f_\ell(x) + f_{j+1}(x)\right\} \\
& \le \min_x \left\{\sum_{\ell=i+1}^{j} f_\ell(x)\right\} + \min_x \left\{f_i(x) + \sum_{\ell=i+1}^{j} f_\ell(x) + f_{j+1}(x)\right\}
\end{aligned}$$

By viewing $g_1(y) = f_i(y), g_2(y) = \sum_{\ell=i+1}^{j} f_\ell(y), g_3(y) = f_{j+1}(y)$ and applying the result in Lemma EC.1 (the conditions for $g_1(y), g_2(y), g_3(y)$ in Lemma EC.1 are satisfied), we know that the inequality in the last line holds. This concludes the proof. □

A.4. Proof of Lemma 1

LEMMA 1. *To compute* $\overline{x}^*_{i,j}$*'s and* $c_{i,j}$*'s defined in* (3) *for all pairs* (i,j) *such that* $1 \le i \le j \le n$*, we have*

(i) If $f_i(x) = p_i x^2 - 2q_i x + r_i$ *with* $p_i > 0$ *for any* $i \in [n]$*, then the computation can be done in* $O(n^2)$ *time; and*

(ii) If $f_i(x) = p_i(r_i - x)^+ + q_i(x - r_i)^+$ *with* $p_i, q_i > 0$ *for any* $i \in [n]$*, then the computation can be done in* $O(n^2 \log n)$ *time.*

Proof. (i) Since $\overline{x}^*_{i,j} = \arg\min \sum_{\ell=i}^{j}(p_l x^2 - 2q_\ell x + r_\ell)$, it can be easily seen that $\overline{x}^*_{i,j} = \frac{\sum_{\ell=i}^{j} q_\ell}{\sum_{\ell=i}^{j} p_\ell}$. Hence, $c_{i,j} = K_i + \sum_{\ell=i}^{j}(p_\ell(\frac{\sum_{\ell=i}^{j} q_\ell}{\sum_{\ell=i}^{j} p_\ell})^2 - 2q_\ell(\frac{\sum_{\ell=i}^{j} q_\ell}{\sum_{\ell=i}^{j} p_\ell}) + r_\ell) = K_i - \frac{(\sum_{\ell=i}^{j} q_\ell)^2}{\sum_{\ell=i}^{j} p_\ell} + \sum_{\ell=i}^{j} r_\ell$. Let us pre-calculate $\sum_{\ell=1}^{i} p_\ell$, $\sum_{\ell=1}^{i} q_\ell$ and $\sum_{\ell=1}^{i} r_\ell$ for all $i \in [n]$ using $O(n)$ time and store them. Hence, each $\overline{x}^*_{i,j} = \frac{\sum_{\ell=1}^{j} q_\ell - \sum_{\ell=1}^{i-1} q_\ell}{\sum_{\ell=1}^{j} p_\ell - \sum_{\ell=1}^{i-1} p_\ell}$ and $c_{i,j} = K_i - \frac{(\sum_{\ell=1}^{j} q_\ell - \sum_{\ell=1}^{i-1} q_\ell)^2}{\sum_{\ell=1}^{j} p_\ell - \sum_{\ell=1}^{i-1} p_\ell} + \sum_{\ell=1}^{j} r_\ell - \sum_{\ell=1}^{i-1} r_\ell$ can be calculated from those stored prefix sums in $O(1)$ time. Then calculating all $\overline{x}^*_{i,j}$'s and $c_{i,j}$'s requires $O(n^2)$ time.

(ii) Since $\sum_{\ell=i}^{j}(p_\ell(r_\ell - x)^+ + q_\ell(x - r_\ell)^+)$ is a piecewise linear function with kinking points $r_i, \ldots, r_j$, then its smallest minimizer $\overline{x}^*_{i,j}$ should also be one of those kinking points. Let ℓ^*_{ij} be an index in

$\{i,\dots,j\}$ such that $\overline{x}^*_{i,j} = r_{\ell^*_{ij}}$. We require that the left derivative at $r_{\ell^*_{ij}}$ is strictly negative and the right derivative is non-negative, i.e., the standard first-order conditions for minimizing a convex function. Thus, $r_{\ell^*_{ij}}$ is the smallest knot value at which the right derivative becomes nonnegative. Equal knot values are ordered by their original indices. Hence, we know that ℓ^*_{ij} should satisfy

$$\begin{aligned} &\sum_{\ell=i:r_\ell>r_{\ell^*_{ij}}}^{j} p_\ell \le \sum_{\ell=i:r_\ell\le r_{\ell^*_{ij}}}^{j} q_\ell \quad \text{and} \quad \sum_{\ell=i:r_\ell\ge r_{\ell^*_{ij}}}^{j} p_\ell > \sum_{\ell=i:r_\ell<r_{\ell^*_{ij}}}^{j} q_\ell, \\ \iff &\sum_{\ell=i:r_\ell>r_{\ell^*_{ij}}}^{j} (p_\ell+q_\ell) \le \sum_{\ell=i}^{j} q_\ell \quad \text{and} \quad \sum_{\ell=i:r_\ell\ge r_{\ell^*_{ij}}}^{j} (p_\ell+q_\ell) > \sum_{\ell=i}^{j} q_\ell. \end{aligned} \tag{EC.1}$$

Let us compute $\sum_{\ell\in[i]} q_\ell$ for all $i\in[n]$ in $O(n)$ time and store them, such that we can obtain $\sum_{\ell=i}^{j} q_\ell = \sum_{\ell\in[j]} q_\ell - \sum_{\ell\in[i-1]} q_\ell$ in $O(1)$ time from those stored prefix sums. After that, let us maintain an appropriate data structure such that we can calculate $\overline{x}^*_{i,i}, \overline{x}^*_{i,i+1}, \dots, \overline{x}^*_{i,n}$ in a single round efficiently. This requires us to maintain a special data structure, such as an augmented balanced binary search tree (e.g., an augmented red-black tree), which is used to compute dynamic order statistics efficiently, as mentioned by Stout (2008). Each node represents an $i\in[n]$ with its key r_i, and the augmented position is the sum of $p_\ell+q_\ell$ over all ℓ's in the subtree rooted at the current node i. To compute $\overline{x}^*_{i,i}, \overline{x}^*_{i,i+1}, \dots, \overline{x}^*_{i,n}$ in round $i\in[n]$, we insert node i till node n one by one, and maintain an augmented balanced binary search tree along the way. Inserting a new node into the tree, rebalancing it, and updating the augmented positions takes $O(\log n)$ time (Stout 2008, Cormen et al. 2022). After doing so for each node j where $j\in\{i,\dots,n\}$, we find ℓ^*_{ij} satisfying the condition (EC.1) in the following way:

(a) Initializing $S=0$ and starting from the root node.

(b) For the current node k, we check the augmented position of its right child (i.e., the sum of $p_\ell+q_\ell$ in its right subtree), and compare its value R plus S with $\sum_{\ell=i}^{j} q_\ell$; if $S+R>\sum_{\ell=i}^{j} q_\ell$, then we move to the right child; else if $S+R\le\sum_{\ell=i}^{j} q_\ell$ and $S+R+(p_k+q_k)>\sum_{\ell=i}^{j} q_\ell$, we output $\ell^*_{ij}=k$ and stop; else if $S+R+(p_k+q_k)\le\sum_{\ell=i}^{j} q_\ell$, we update $S=S+R+(p_k+q_k)$ and move to the left child.

(c) Recursively doing step (b) until we output ℓ^*_{ij}.

The validity follows directly from the logic of the above procedure. It requires $O(\log n)$ steps, each implemented using $O(1)$ time. Hence, calculating all $\overline{x}^*_{i,i}, \overline{x}^*_{i,i+1}, \dots, \overline{x}^*_{i,n}$ in the same loop requires $O(n\log n)$ time, and computing all $\overline{x}^*_{i,j}$'s requires us to implement the above loop for all $i\in[n]$, which takes $O(n^2\log n)$ time.

To compute $c_{i,j}$'s based on those $\overline{x}^*_{i,j}$'s, note that each $\overline{x}^*_{i,j}$ is just $r_{\ell^*_{ij}}$ for some $\ell^*_{ij}\in\{i,\dots,j\}$. Hence, we calculate $\sum_{\ell\in[i]}(p_\ell(r_\ell-r_k)^+ + q_\ell(r_k-r_\ell)^+)$ for each $i\in[n]$ and $k\in[n]$ in $O(n^2)$ time and store them, and compute $c_{i,j} = K_i + \sum_{\ell=i}^{j}(p_\ell(r_\ell-r_{\ell^*_{ij}})^+ + q_\ell(r_{\ell^*_{ij}}-r_\ell)^+) = K_i + (\sum_{\ell\in[j]}(p_\ell(r_\ell -$

$r_{\ell^*_{ij}})^+ + q_\ell(r_{\ell^*_{ij}} - r_\ell)^+)) - (\sum_{\ell\in[i-1]}(p_\ell(r_\ell - r_{\ell^*_{ij}})^+ + q_\ell(r_{\ell^*_{ij}} - r_\ell)^+))$ from those prefixed sums in $O(1)$ time. Hence, the computation of all $c_{i,j}$'s needs additional $O(n^2)$ time.

Consequently, the overall computation complexity is $O(n^2 \log n)$. □

Appendix B: Proofs of Technical Results in Section 3

B.1. Proof of Proposition 3

PROPOSITION 3. *There exists an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the Arborescence problem with the fewest strict increases and $y^*_i = \mathbb{1}\{x^*_i > x^*_{p(i)}\}$ for all $i \in [n]$. Let $M \subseteq [n]$ denote the support of $\boldsymbol{y}^*$. Then the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ partitions $\mathcal{T}$ into several induced arborescences $\mathcal{T}^i = (N_{\mathcal{T}^i}, A_{\mathcal{T}^i})$ for all $i \in M$, where each $\mathcal{T}^i$ is rooted at node i and $N_{\mathcal{T}^i} = N_{\mathcal{T}(i)} \backslash (\cup_{j\in N_{\mathcal{T}(i)}\cap M, j\neq i} N_{\mathcal{T}(j)})$, and $A_{\mathcal{T}^i}$ are arcs of $A_{\mathcal{T}}$ restricted on $N_{\mathcal{T}^i}$. Let $\overline{x}^*_{\mathcal{T}^i} = \min\{x^* | x^* \in \arg\min \sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(x)\}$ and $c_{\mathcal{T}^i} = \sum_{\ell\in N_{\mathcal{T}^i}} f_\ell(\overline{x}^*_{\mathcal{T}^i})$. Such an optimal solution can be chosen to satisfy:*

(a) $x^*_\ell = \overline{x}^*_{\mathcal{T}^i}$ for all $\ell \in N_{\mathcal{T}^i}$, and

(b) $\overline{x}^*_{\mathcal{T}^i} < \overline{x}^*_{\mathcal{T}^j}$ for all $i \in M$ and $j \in (N_{\mathcal{T}(i)} \cap M)\backslash\{i\}$.

Proof. The proof follows a similar idea to the proof for Theorem 1. Hence, we will only prove it by sketch. Assume by contradiction that $x^*_\ell \equiv \widetilde{x}_{\mathcal{T}^i}$ for all $\ell \in N_{\mathcal{T}^i}$ and $\widetilde{x}_{\mathcal{T}^i} \neq \overline{x}^*_{\mathcal{T}^i}$. Two cases need to be discussed. One is that $\widetilde{x}_{\mathcal{T}^i} > \overline{x}^*_{\mathcal{T}^i}$, for which case we can decrease $\widetilde{x}_{\mathcal{T}^i}$ until $\widetilde{x}_{\mathcal{T}^i} = \overline{x}^*_{\mathcal{T}^i}$ or arborescence $\mathcal{T}^i$ is merged with its precedent arborescence $\mathcal{T}^j$ where $i \in N_{\mathcal{T}(j)} \cap M$, contradicting the optimality of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. The other is that $\widetilde{x}_{\mathcal{T}^i} < \overline{x}^*_{\mathcal{T}^i}$, in which case we can either increase $\widetilde{x}_{\mathcal{T}^i}$ until $\widetilde{x}_{\mathcal{T}^i} = \overline{x}^*_{\mathcal{T}^i}$ or the arborescence $\mathcal{T}^i$ is merged with one of its subsequent arborescences $\mathcal{T}^j$ for all $j \in N_{\mathcal{T}(i)} \cap M$, which contradicts the optimality of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. This concludes the proof. □

B.2. Proof of Proposition 4

PROPOSITION 4. *Under Assumption 2, the Arborescence problem can be solved in $O((n + \log(\sum_{i\in[n]} m_i))\sum_{i\in[n]} m_i)$ time and $O(n\sum_{i\in[n]} m_i)$ auxiliary space by efficiently implementing (DP-PL). Both bounds are $O(n^2)$ if $m_i = O(1)$ for all $i \in [n]$.*

Proof. We first sort all the kink points in $\mathcal{K}$ using $O(\sum_{i\in[n]} m_i \log(\sum_{i\in[n]} m_i))$ time. Then, for each $i \in [n]$, we do backpropagation following the same idea of Algorithm 1 to calculate $V(i,k)$'s for all $k \in \mathcal{K}$ in the same round. With the presorted kink points in $\mathcal{K}$, for each $i \in [n]$, the backpropagation step to compute $V(i,k)$'s for all $k \in \mathcal{K}$ only needs $O(\sum_{i\in[n]} m_i)$ time, and for all $i \in [n]$ it takes $O(n\sum_{i\in[n]} m_i)$. Besides, recall that the evaluation part also takes $O(n\sum_{i\in[n]} m_i)$ time. Putting them together, we conclude that (DP-PL) can be solved in $O((n+\log(\sum_{i\in[n]} m_i))\sum_{i\in[n]} m_i)$ total time. It is of the order $O(n^2)$ if we assume that $m_i \equiv O(1)$ for all $i \in [n]$. For space, there are $O(n\sum_{i\in[n]} m_i)$ DP states, each storing a constant number of values and an index of an optimal decision. These indices allow an optimal solution to be recovered in $O(n)$ time. The sorted kink points and traversal stack require $O(\sum_{i\in[n]} m_i + n)$ additional space. Hence the total auxiliary space is $O(n\sum_{i\in[n]} m_i)$, which is $O(n^2)$ when $m_i = O(1)$ for all $i \in [n]$. □

B.3. Proof of Theorem 3

Below, we summarize several key properties of K-convex functions from the literature, which are useful to prove Theorem 3.

LEMMA EC.2 **(Lemma 9.3.2 and Proposition 9.3.3 of Simchi-Levi et al. (2014))**. *The K-convex functions have the following properties:*

(a) A real-valued convex function is also 0-convex and hence K convex for all $K \geq 0$. A K_1-convex function is also K_2-convex function for $K_1 \leq K_2$.

(b) If $f_1(x)$ and $f_2(x)$ are K_1-convex and K_2-convex, then for $\alpha, \beta \geq 0$, $\alpha f_1(x) + \beta f_2(x)$ is $(\alpha K_1 + \beta K_2)$-convex.

(c) If $f(x)$ is a K-convex function, then function

$$\phi(x) = \min_{x' \geq x} \left\{ Q\mathbb{1}\{x' > x\} + f(x') \right\}$$

is $\max\{K, Q\}$-convex.

(d) Assume that f is a continuous K-convex function and $f(x) \to \infty$ as $|x| \to \infty$. Let S be a minimum point of f and s be any element of the set

$$\left\{ x | x \leq S, f(x) = f(S) + K \right\}.$$

Then the following results hold:

(i) $f(S) + K = f(s) \leq f(x)$, for all $x \leq s$.

(ii) $f(x)$ is a non-increasing function on $(-\infty, s)$.

(iii) $f(x) \leq f(x') + K$ for all x, x' with $s \leq x \leq x'$.

THEOREM 3. *Under Assumption 3, for each $i \in [n]$, the value-to-go function $V(i, x)$ defined in (5) satisfies the following four properties:*

(a) $V(i, x)$ is K_i-convex;

(b) there exists a pair of real numbers (s_i, S_i) satisfying

$$S_i \in \arg\min U(i, x), \tag{7a}$$

$$s_i \in \left\{ x | x \leq S_i, U(i, x) = U(i, S_i) + K_i \right\}, \tag{7b}$$

such that an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the Arborescence problem is

$$x_i^* = \begin{cases} S_i, & x_{p(i)}^* \leq s_i, \\ x_{p(i)}^*, & o.w. \end{cases}, \quad y_i^* = \begin{cases} 1, & x_{p(i)}^* \leq s_i, \\ 0, & o.w; \end{cases}$$

(c) $V(i,x)$ can be expressed as

$$V(i,x)=\begin{cases}K_i+U(i,S_i), & x\le s_i,\\ U(i,x), & o.w.;\end{cases}$$

(d) $V(i,x)$ is piecewise convex with at most $|N_{\mathcal{T}(i)}|+|\mathcal{L}(i)|$ number of convex pieces.

Proof. We prove this by induction. First consider all $i\in\mathcal{L}$, and in this case $U(i,x)=f_i(x)$. For each $i\in\mathcal{L}$, $V(i,x)=\min_{x'\ge x}\{K_i\mathbb{1}\{x'>x\}+f_i(x')\}$. Because $f_i(x)$ is convex whence 0-convex by Lemma EC.2(a), from Lemma EC.2(c) we know that $V(i,x)$ is K_i-convex and (a) holds. Besides, since $f_i(x)\to\infty$ as $|x|\to\infty$, with the definition of (s_i,S_i) we know that when $x^*_{p(i)}\le s_i$, we should increase x_i to let $x^*_i=S_i$ and incur a cost $K_i+f_i(S_i)=f_i(s_i)\le f_i(x^*_{p(i)})$, otherwise we keep $x^*_i=x^*_{p(i)}$ unchanged and incur a cost $f_i(x^*_{p(i)})$. Hence, we can express $V(i,x)$ as

$$V(i,x)=\begin{cases}K_i+f_i(S_i), & x\le s_i,\\ f_i(x), & \text{o.w.}\end{cases}$$

It verifies (b). It can be easily seen that $V(i,x)$ is a piecewise convex function with two pieces $(-\infty,s_i]$ and (s_i,∞). Since i is a leaf node, $|N_{\mathcal{T}(i)}|=|\mathcal{L}(i)|=1$, we know that $V(i,x)$ is a piecewise convex function with $|N_{\mathcal{T}(i)}|+|\mathcal{L}(i)|$ pieces. Hence, (c) is justified, and the statement in the theorem holds for all leaf nodes.

For $i\in[n]\backslash\mathcal{L}$, suppose the statement in the theorem holds for $V(j,x)$ for all $j\in\mathcal{C}(i)$. Again, recall that $V(i,x)=\min_{x'\ge x}\{K_i\mathbb{1}\{x'>x\}+U(i,x')\}$. Since each $V(j,x)$ is K_j-convex for each $j\in\mathcal{C}(i)$ and $f_i(x)$ is convex whence 0-convex, by Lemma EC.2(b) we know that $U(i,x)=f_i(x)+\sum_{j\in\mathcal{C}(i)}V(j,x)$ is $\sum_{j\in\mathcal{C}(i)}K_j$-convex. Since $K_i\ge\sum_{j\in\mathcal{C}(i)}K_j$ according to our assumption, by Lemma EC.2(c) we know that $V(i,x)$ is K_i-convex, which means (a) holds. Because $U(i,x)$ is $\sum_{j\in\mathcal{C}(i)}K_j$-convex whence K_i-convex by our assumption $K_i\ge\sum_{j\in\mathcal{C}(i)}K_j$ and Lemma EC.2(a), and goes to ∞ as $|x|\to\infty$, from the definition of (s_i,S_i) and Lemma EC.2(d) we know that

(i) $U(i,S_i)+K_i=U(i,s_i)\le U(i,x)$, for all $x\le s_i$.

(ii) $U(i,x)$ is a non-increasing function on $(-\infty,s_i)$.

(iii) $U(i,x)\le U(i,x')+K_i$ for all x,x' with $s_i\le x\le x'$.

Hence, we will make an increase and let $x^*_i=S_i$ if $x^*_{p(i)}\le s_i$, and hold it still to be $x^*_{p(i)}$ otherwise. Therefore, $V(i,x)$ can be expressed as

$$V(i,x)=\begin{cases}K_i+U(i,S_i), & x\le s_i,\\ U(i,x), & \text{o.w.}\end{cases}$$

It verifies (b) and (c). Besides, for each $j\in\mathcal{C}(i)$, $V(j,x)$ is a piecewise convex function with $|N_{\mathcal{T}(j)}|+|\mathcal{L}(j)|$ pieces by induction hypothesis, which means that $\sum_{j\in\mathcal{C}(i)}V(j,x)$ is a piecewise convex function with at most $\sum_{j\in\mathcal{C}(i)}|N_{\mathcal{T}(j)}|+\sum_{j\in\mathcal{C}(i)}|\mathcal{L}(j)|=|N_{\mathcal{T}(i)}\backslash\{i\}|+|\mathcal{L}(i)|=|N_{\mathcal{T}(i)}|+|\mathcal{L}(i)|-1$ number

of pieces, the same for $U(i,x) = f_i(x) + \sum_{j\in\mathcal{C}(i)} V(j,x)$ by simply adding another convex function $f_i(x)$. From the expression of $V(i,x)$, we know that it introduces at most one convex piece $(-\infty, s_i]$ compared with $U(i,x)$. Hence, we conclude that $V(i,x)$ is a piecewise convex function with at most $|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|$ pieces and the proof of part (d) is completed. □

B.4. Proof of Proposition 5

PROPOSITION 5. *Suppose that $K_i \geq \sum_{j\in\mathcal{C}(i)} K_j$ for all $i \in [n]$. Further assume that the thresholds S_i and s_i, defined in (7a) and (7b), respectively, can be chosen from a known uniform grid with spacing $\epsilon > 0$, and that all function evaluations and comparisons can be performed exactly. Then, the following statements hold:*

(a) There exists a list representation (8) for each $V(i,x)$, whose functional form can be recovered from the list representation according to (9).

(b) The optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ for the Arborescence problem can be obtained in $O(n^3 \log(U/\epsilon))$ time and $O(n^2)$ auxiliary space.

Proof. We provide a step-by-step algorithm to show the validity of the list representation, and the algorithm's computational complexity will be derived. We first round ℓ down and u up to the nearest points of the known grid, retaining ℓ and u for the resulting endpoints and setting $U = u - \ell$. All searches use the known grid indices and exact function values and comparisons, ensuring exact threshold recovery. Solution reconstruction requires only the lists restricted to $[\ell, u]$; thresholds below ℓ may be omitted. The initial root value is computed separately as $V(1,-\infty) = K_1 + \min_{z\in[\ell,u]} U(1,z)$.

Step A. Computation for Leaf Nodes: We first show that the list representation (9) is valid for each $i \in \mathcal{L}$ and show how to obtain the list as well as (s_i, S_i) and compute its complexity. Note that in this case, $U(i,x) = f_i(x)$. Hence, $S_i \in \arg\min\{f_i(x)\}$. The minimizer S_i is recovered exactly by minimizing f_i over the grid points in $[\ell, u]$, using $O(\log(U/\epsilon))$ adjacent-grid comparisons. If $s_i \in [\ell, u]$, a grid-index search for $f_i(s_i) = f_i(S_i) + K_i$ similarly recovers s_i exactly in $O(\log(U/\epsilon))$ time. Note that there can be a case where $s_i \leq \ell$. If so, s_i does not need to be computed since it never helps us compute the optimal solution, as $x^*_{p(i)} \in [\ell, u]$ indicated by Theorem 3 and $x^*_{p(i)} \geq s_i$ always holds, which means we do not need to make the increase at this node according to Theorem 3(b). Hence, by Theorem 3(b), we know that

$$V(i,x) = \begin{cases} K_i + f_i(S_i), & x \leq s_i, \\ f_i(x), & \text{o.w.} \end{cases}$$

It can be easily seen that the representation list for $V(i,x)$ is

$$\text{List}_i = [(-\infty, s_i, K_i + f_i(S_i), \emptyset), (s_i, \infty, 0, \{i\})],$$

and the overall complexity of obtaining this list as well as (s_i, S_i) is $O(\log(U/\epsilon))$. Hence, for all $i \in \mathcal{L}$, this step takes $O(|\mathcal{L}| \log(U/\epsilon)) = O(n \log(U/\epsilon))$ time.

Step B. Computation for Non-leaf Nodes: For each $i \in [n]\backslash\mathcal{L}$, now assume that List_j and (s_j, S_j) have already been computed for all $j \in \mathcal{C}(i)$. We show that the list representation (9) also exists for $V(i,x)$, discuss how to compute List_i as well as (s_i, S_i), and how to compute its complexity. We further divide the process into several sub-procedures:

Step B.1. Get List Representation of $U(i,x)$: We first obtain the list representation of $U(i,x) = f_i(x) + \sum_{j\in\mathcal{C}(i)} V(j,x)$ from List_j for each $j \in \mathcal{C}(i)$. Since $f_i(x)$ is convex and $V(j,x)$ is piecewise convex for each $j \in \mathcal{C}(i)$, $U(i,x)$ is also piecewise convex, whose breakpoints can be obtained by taking the distinct union of breakpoints of $V(j,x)$'s for all $j \in \mathcal{C}(i)$ and sort them increasingly, i.e.,

$$[\widetilde{b}^i_1, \widetilde{b}^i_2, \ldots, \widetilde{b}^i_{\widetilde{R}_i}] = \text{sort}\left(\bigcup_{j\in\mathcal{C}(i)} [b^j_1, b^j_2, \ldots, b^j_{R_j-1}]\right),$$

where $\text{sort}(\cdot)$ is the operator of returning an increasingly sorted input list, and $\widetilde{b}^i_1, \widetilde{b}^i_2, \ldots, \widetilde{b}^i_{\widetilde{R}_i-1}$ are $\widetilde{R}_i - 1$ many increasingly ordered distinct breakpoints of $U(i,x)$. Since each $[b^j_1, b^j_2, \ldots, b^j_{R_j}]$ is an increasingly sorted list, to get the merged increasingly sorted list $[\widetilde{b}^i_1, \widetilde{b}^i_2, \ldots, \widetilde{b}^i_{\widetilde{R}_i-1}]$, the best-known complexity is $O(\log(|\mathcal{C}(i)|)\sum_{i\in\mathcal{C}(i)} R_j)$ achieved by using heap (see, for example, Cormen et al. (2022)). By Theorem 3(c) we know that $\sum_{i\in\mathcal{C}(i)} R_j \le \sum_{i\in\mathcal{C}(i)}(|N_{\mathcal{T}(j)}| + |\mathcal{L}(j)|) \le |N_{\mathcal{T}(i)}| + |\mathcal{L}(i)| - 1$, hence we know that the complexity can be represented as $O(\log(|\mathcal{C}(i)|)(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|))$.

Besides $\widetilde{b}^i_1, \widetilde{b}^i_2, \ldots, \widetilde{b}^i_{\widetilde{R}_i-1}$, we should also get $\widetilde{L}^i_1, \widetilde{L}^i_2, \ldots, \widetilde{L}^i_{\widetilde{R}_i}$ and $\widetilde{\mathcal{J}}^i_1, \widetilde{\mathcal{J}}^i_2, \ldots, \widetilde{\mathcal{J}}^i_{\widetilde{R}_i}$ in order to get the full list representation of $U(i,x)$ as

$$\widetilde{\text{List}}_i = \left[(-\infty, \widetilde{b}^i_1, \widetilde{L}^i_1, \widetilde{\mathcal{J}}^i_1), (\widetilde{b}^i_1, \widetilde{b}^i_2, \widetilde{L}^i_2, \widetilde{\mathcal{J}}^i_2), \ldots, (\widetilde{b}^i_{\widetilde{R}_i-1}, \infty, \widetilde{L}^i_{\widetilde{R}_i}, \widetilde{\mathcal{J}}^i_{\widetilde{R}_i})\right].$$

Denote $\widetilde{b}^i_0 = -\infty$ and $\widetilde{b}^i_{\widetilde{R}_i} = \infty$. For each $r \in [\widetilde{R}_i]$ and $j \in \mathcal{C}(i)$, let $r_j(r) \in [R_j]$ be the index such that $(\widetilde{b}^i_{r-1}, \widetilde{b}^i_r) \in (b^j_{r_j-1}, b^j_{r_j})$. Hence, we can calculate $\widetilde{L}^i_r$ and $\mathcal{J}^i_r$ through

$$\widetilde{L}^i_r = \sum_{j\in\mathcal{C}(i)} L^j_{r_j(r)}, \quad \widetilde{\mathcal{J}}^i_r = \{i\} \cup \left(\bigcup_{j\in\mathcal{C}(i)} \mathcal{J}^j_{r_j(r)}\right), \quad \text{for each } r \in [\widetilde{R}_i].$$

Those summations and unions operations for all $r \in [\widetilde{R}_i]$ can be done in $O(\widetilde{R}_i \sum_{j\in\mathcal{C}(i)} |\mathcal{J}^j_{r_j(r)}|)$ time (note that by maintaining a pointer of each list $[\widetilde{b}^j_1, \widetilde{b}^j_2, \ldots, \widetilde{b}^j_{\widetilde{R}_j-1}]$ for $j \in \mathcal{C}(i)$, we can decide $r_j(1), r_j(2), \ldots, r_j(\widetilde{R}_i)$ sequentially by moving the pointer gradually to the right in the list and never returning back, which requires time $O(\widetilde{R}_i|\mathcal{C}(i)|)$ for all $j \in \mathcal{C}(i)$ and does not harm this complexity). Since $\widetilde{R}_i \le \sum_{j\in\mathcal{C}(i)} R_j \le |N_{\mathcal{T}(i)}| + |\mathcal{L}(i)| - 1$, and $\sum_{j\in\mathcal{C}(i)} |\mathcal{J}^j_{r_j(r)}| \le \sum_{j\in\mathcal{C}(i)} |N_{\mathcal{T}(j)}| = |N_{\mathcal{T}(i)}| - 1$, we can re-express this complexity as $O(|N_{\mathcal{T}(i)}|(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|))$.

In conclusion, the overall complexity of getting the list representation $\widetilde{\text{List}}_i$ of $U(i,x)$ is $O(|N_{\mathcal{T}(i)}|(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|))$. For all $i \in [n]\backslash\mathcal{L}$, this step takes $O(\sum_{i\in[n]\backslash\mathcal{L}} |N_{\mathcal{T}(i)}|(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|)) = O(n^3)$ time.

Step B.2. Obtain (s_i, S_i): Now given the list representation $\widetilde{\text{List}}_i$ obtained from the last step for $U(i,x)$, we want to compute (s_i, S_i). According to the definition, $S_i \in \arg\min U(i,x)$. To compute S_i, minimize $U(i,x)$ over the grid points in each convex piece intersected with $[\ell, u]$, and choose a grid point with the smallest value among these candidates. On r-th piece, $U(i,x) = \widetilde{L}^i_r + \sum_{j\in\widetilde{\mathcal{J}}^i_r} f_j(x)$, whose evaluation for any particular x in the piece takes $O(|\widetilde{\mathcal{J}}^i_r|)$ time. Note that the binary search only needs to be implemented in r-th piece intersecting with $[\ell, u]$ (the range that any possible x^*_i must locate in), which is denoted as $\mathcal{B}^i_r$ with interval length B^i_r. Skip a piece only if $\mathcal{B}^i_r$ contains no grid point. If it contains exactly one grid point, evaluate $U(i,x)$ at that point directly; otherwise, apply binary search over its grid points. This takes $O(|\widetilde{\mathcal{J}}^i_r|\log(2+B^i_r/\epsilon))$ time per retained piece. Hence, the overall complexity to find S_i is $O(\sum_{r\in[\widetilde{R}_i]}|\widetilde{\mathcal{J}}^i_r|\log(2+B^i_r/\epsilon))$ time. Using the fact that $|\widetilde{\mathcal{J}}^i_r| \le |N_{\mathcal{T}(i)}|$ and $B^i_r \le U$ for any $r \in \widetilde{R}_i$, the complexity can be further simplified as $O(\widetilde{R}_i|N_{\mathcal{T}(i)}|\log(U/\epsilon)) = O((|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|)|N_{\mathcal{T}(i)}|\log(U/\epsilon))$.

If $U(i,\ell) \le U(i,S_i) + K_i$, then $V(i,x) = U(i,x)$ on $[\ell, u]$. In this case, the list for $U(i,\cdot)$ on this interval will not change and the next Step B.3 can be skipped. Otherwise, to compute s_i, we find a convex piece in $\widetilde{\text{List}}_i$ (say, the r_{s_i}-th piece $(\widetilde{b}^i_{r_{s_i}-1}, \widetilde{b}^i_{r_{s_i}}]$) whose intersection with $[\ell, S_i]$ is nonempty and satisfies

$$U\left(i, \max\{\ell, \widetilde{b}^i_{r_{s_i}-1}\}\right) \ge U(i,S_i) + K_i, \qquad U\left(i, \min\{S_i, \widetilde{b}^i_{r_{s_i}}\}\right) \le U(i,S_i) + K_i.$$

For the selected piece, let $A = \max\{\ell, \tilde{b}^i_{r_{s_i}-1}\}$ and $B = \min\{S_i, \tilde{b}^i_{r_{s_i}}\}$. By Lemma EC.2(d) and Theorem 3(b), the closed interval $[A, B]$ contains a valid s_i. Search its grid points, including both endpoints, to recover s_i exactly. Each evaluation of $U(i,\cdot)$ on this piece costs $O(|\widetilde{\mathcal{J}}^i_{r_{s_i}}|)$ operations. Including the search for the piece, the complexity is $O(|N_{\mathcal{T}(i)}|(\widetilde{R}_i + \log(U/\epsilon))) = O(|N_{\mathcal{T}(i)}|(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)| + \log(U/\epsilon)))$, where we use $\widetilde{R}_i \le |N_{\mathcal{T}(i)}| + |\mathcal{L}(i)| - 1$, with logarithmic factors to be at least one. Thus, this cost is absorbed by the bound for computing (s_i, S_i).

In conclusion, the time for obtaining (s_i, S_i) from $\widetilde{\text{List}}_i$ is $O((|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|)|N_{\mathcal{T}(i)}|\log(U/\epsilon))$. Therefore, for all $i \in [n]\backslash\mathcal{L}$, this step takes $O(\sum_{i\in[n]\backslash\mathcal{L}}(|N_{\mathcal{T}(i)}| + |\mathcal{L}(i)|)|N_{\mathcal{T}(i)}|\log(U/\epsilon)) = O(n^3\log(U/\epsilon))$ time.

Step B.3. Get List Representation of $V(i,x)$: With $\widetilde{\text{List}}_i$ and (s_i, S_i) pre-computed, we are now ready to get the list representation List_i of $V(i,x)$. Recall that $V(i,x)$ can be expressed as

$$V(i,x) = \begin{cases} K_i + U(i,S_i), & x \le s_i, \\ U(i,x), & \text{o.w.} \end{cases}$$

From this, we can see that List_i can be obtained from $\widetilde{\text{List}}_i$ by executing the following operations: we first delete the first r_{s_i} tuples in $\widetilde{\text{List}}_i$. After that, we append two tuples $(-\infty, s_i, K_i + U(i,S_i), \emptyset)$ and $(s_i, \widetilde{b}_{r_{s_i}}, \widetilde{L}_{r_{s_i}}, \widetilde{\mathcal{I}}_{r_{s_i}})$ on the left side of the list. The second tuple is omitted if $s_i = \widetilde{b}_{r_{s_i}}$. It only consumes $O(1)$ time. Hence, for all $i \in [n]\backslash\mathcal{L}$, this step consumes $O(n)$ time.

Step C. Recover the Optimal Solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ Using Theorem 3(b), we can easily recover the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ in $O(n)$ time given (s_i, S_i)'s for any $i \in [n]$ have already been obtained.

For the space complexity analysis, we count each stored index, scalar, or pointer as one storage unit. Storing a list with explicit index sets requires $O(R_i + \sum_{r=1}^{R_i} |\mathcal{J}_r^i|) = O(|N_{\mathcal{T}(i)}|^2)$ storage units, since $R_i \le 2|N_{\mathcal{T}(i)}|$. We process nodes in postorder, retain the thresholds (s_i, S_i), and discard the child lists after constructing the parent list for $U(i,\cdot)$. Between merges, the retained lists correspond to pairwise disjoint subtrees, so their combined storage is $O(n^2)$. Constructing the parent list at node i requires an additional $O(|N_{\mathcal{T}(i)}|^2)$ storage units, while the thresholds and merge pointers require $O(n)$ units in total. Grid searches store indices rather than the entire grid. Thus, the maximum auxiliary space is $O(n^2)$, independently of the branching factor. An optimal solution can be recovered from the stored thresholds without recomputing the value functions.

Putting all the analysis above together, we can conclude that there exists a list representation (8) for each $V(i,x)$, whose functional form can be recovered from the list representation according to (9), and the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of Arborescence problem can be obtained in $O(n^3 \log(U/\epsilon))$ time and $O(n^2)$ auxiliary space. □

Appendix C: Proofs of Technical Results in Section 4

C.1. Proof of Lemma 2

LEMMA 2. *The value-to-go function $V(i,x)$ is piecewise convex for any $i \in N_{\mathcal{T}}$, regardless of the arborescence structure.*

Proof. The proof is conducted by induction. We first check the statement for each $i \in \mathcal{L}$. From Theorem 3 and the fact that $U(i,x) = f_i(x)$, we know that

$$V(i,x) = \begin{cases} K_i + f_i(S_i), & x \le s_i, \\ f_i(x), & \text{o.w.} \end{cases}$$

where S_i and s_i are defined by (7a) and (7b). Therefore, $V(i,x)$ is piecewise convex with two convex pieces.

Next, for any $i \in N_{\mathcal{T}} \backslash \mathcal{L}$, assume $V(j,x)$ is piecewise convex for each $j \in \mathcal{C}(i)$. Recall that

$$V(i,x) = \min_{x' \ge x} \left\{ K_i \mathbb{1}\{x' > x\} + U(i,x') \right\}.$$

Since the summation of piecewise convex functions is still piecewise convex, we know that $U(i,x) = f_i(x) + \sum_{j \in \mathcal{C}(i)} V(j,x)$ is also a piecewise convex function. For convenience, let us express its convex pieces to be $P_1 = (-\infty, c_1], P_2 = (c_1, c_2], \ldots, P_L = (c_{L-1}, \infty)$, and the minimum at each convex piece to be

$$\widehat{x}_\ell = \min \left\{ x \,|\, x \in \arg\min_{y \in \mathrm{cl}(P_\ell)} U(i,y) \right\}, \quad \forall \ell \in [L],$$

where $\mathrm{cl}(\cdot)$ denotes the closure of a set. Now define a map $S_i(x):(-\infty,\widehat{x}_L)\to\{\widehat{x}_1,\dots\widehat{x}_L\}$ as

$$S_i(x)=\min\ \underset{y\in\{\widehat{x}_{\ell(x)},\dots,\widehat{x}_L\}}{\arg\min}\ U(i,y),$$

where $\ell(x)=\min\{\ell\in[L]\,|\,\widehat{x}_\ell>x\}$. $S_i(x)$ represents the optimal minimum point of $U(i,x)$ in all convex pieces with its value greater than x. Hence, $V(i,x)$ can be re-expressed as

$$V(i,x)=\begin{cases}\min\{U(i,x),\,K_i+U(i,S_i(x))\} & x\in(-\infty,\widehat{x}_L)\\ U(i,x) & x\in[\widehat{x}_L,\infty)\end{cases}.$$

It is because when $x\in(-\infty,\widehat{x}_L)$, due to the piecewise convexity of $K_i+U(i,x)$, as long as making an increase is a better decision compared with holding still at i, the increased level should be the best minimum point of $U(i,x)$ in all convex pieces with its value greater than x, i.e., $S_i(x)$; when $x\in[\widehat{x}_L,\infty)$, $U(i,x)$ is non-decreasing, which means that the optimal decision is to make no increase. Besides, observe that $S_i(x)$ is a non-decreasing right continuous step function with at most L steps, and we can define its discontinuous points as $q_1,q_2,\dots,q_{\tau-1}$ (note that $\{q_1,\dots,q_{\tau-1}\}\subseteq\{\widehat{x}_1,\dots,\widehat{x}_L\}$), such that $S_i(x)$ holds to be the same value in $Q_1=(-\infty,q_1),Q_2=[q_1,q_2),\dots,Q_\tau=[q_{\tau-1},\widehat{x}_L)$, with $2\le\tau\le L$. If $\tau=1$, take $Q_1=(-\infty,\widehat{x}_L)$. Let the value of $S_i(x)$ evaluated in $Q_1,\dots,Q_\tau$ to be $S_i^1,\dots,S_i^\tau$, then $V(i,x)$ can be further evaluated as

$$V(i,x)=\begin{cases}\min\{U(i,x),\,K_i+U(i,S_i^1)\} & x\in Q_1,\\ \min\{U(i,x),\,K_i+U(i,S_i^2)\} & x\in Q_2,\\ \dots & \\ \min\{U(i,x),\,K_i+U(i,S_i^\tau)\} & x\in Q_\tau,\\ U(i,x) & x\in[\widehat{x}_L,\infty)\end{cases}.$$

It just means that in each Q_t for $t\in[\tau]$, the functional form of $V(i,x)$ is obtained from $U(i,x)$ by changing the evaluation at the region where $U(i,x)\ge K_i+U(i,S_i^t)$ to be $K_i+U(i,S_i^t)$ and holding the evaluation the same as $U(i,x)$ in the remaining region of Q_t. Geometrically speaking, it just means that several flat pieces evaluated as $K_i+U(i,S_i^t)$ might be introduced. Hence, $V(i,x)$ is still piecewise convex in Q_t. In conclusion, $V(i,x)$ is a piecewise convex function. □

C.2. Proof of Lemma 3

LEMMA 3. *Suppose $g:\mathbb{R}\to\mathbb{R}$ is continuous and piecewise convex, with L convex pieces and $R\le L$ quasi-convex pieces. Assume that g attains a minimum at a finite point on the closure of each quasi-convex piece. Then, for any $K\in\mathbb{R}_+$, the function $h:\mathbb{R}\to\mathbb{R}$ defined by*

$$h(x)=\min_{x'\ge x}\left\{K\mathbb{1}\{x'>x\}+g(x')\right\}$$

is piecewise convex with at most $L+R$ number of convex pieces.

Proof. For g, let its convex pieces be $P_1=(-\infty,c_1], P_2=(c_1,c_2],\dots,P_L=(c_{L-1},\infty)$, and quasi-convex pieces be $\widetilde{P}_1=(-\infty,\widetilde{c}_1], \widetilde{P}_2=(\widetilde{c}_1,\widetilde{c}_2],\dots,\widetilde{P}_R=(\widetilde{c}_{R-1},\infty)$. Choose a finite minimizer of g on the closure of each quasi-convex piece

$$\widetilde{x}_r \in \arg\min_{y\in \mathrm{cl}(\widetilde{P}_r)} g(y), \quad \forall r\in[R],$$

where $\mathrm{cl}(\cdot)$ denotes the closure of a set. Following the similar idea as in the proof of Lemma 2, but now for quasi-convex pieces instead of convex pieces, define a map $\widetilde{S}(x):(-\infty,\widetilde{x}_R)\to\{\widetilde{x}_1,\dots\widetilde{x}_R\}$ as

$$\widetilde{S}(x)=\min \arg\min_{y\in\{\widetilde{x}_{r(x)},\dots,\widetilde{x}_R\}} g(y),$$

where $r(x)=\min\{r\in[R]\,|\,\widetilde{x}_r>x\}$. $\widetilde{S}(x)$ represents the optimal minimum point in all quasi-convex pieces with its value greater than x. By nature of the weak unimodality of $g(y)$ in each quasi-convex piece, with the same reason as mentioned in the proof of Lemma 2, $h(x)$ can be re-expressed as

$$h(x)=\begin{cases}\min\{g(x)\,,\,K+g(\widetilde{S}(x))\} & x\in(-\infty,\widetilde{x}_R)\\ g(x) & x\in[\widetilde{x}_R,\infty)\end{cases}.$$

Now let us partition $(-\infty,\widetilde{x}_R)$ by pieces $\widetilde{Q}_1=(-\infty,\widetilde{x}_1), \widetilde{Q}_2=[\widetilde{x}_1,\widetilde{x}_2),\dots,\widetilde{Q}_R=[\widetilde{x}_{R-1},\widetilde{x}_R)$. Note that $\widetilde{S}(x)$ is constantly evaluated in each $\widetilde{Q}_r$ for $r\in[R]$ (actually, $\widetilde{S}(x)$ may even be constantly evaluated in several consecutive $\widetilde{Q}_r$). Define the evaluation of $\widetilde{S}(x)$ in $\widetilde{Q}_1,\dots,\widetilde{Q}_R$ to be $\widetilde{S}^1,\dots,\widetilde{S}^R$. Then, $h(x)$ can be further expressed as:

$$h(x)=\begin{cases}\min\{g(x),K+g(\widetilde{S}^1)\}, & x\in\widetilde{Q}_1,\\ \min\{g(x),K+g(\widetilde{S}^2)\}, & x\in\widetilde{Q}_2,\\ \dots & \\ \min\{g(x),K+g(\widetilde{S}^R)\}, & x\in\widetilde{Q}_R,\\ g(x) & x\in[\widetilde{x}_R,\infty)\end{cases}.$$

The function g is non-increasing on $\widetilde{Q}_1$. For each $r=2,\dots,R$, it is non-decreasing on $[\widetilde{x}_{r-1},\widetilde{c}_{r-1})$ and non-increasing on $[\widetilde{c}_{r-1},\widetilde{x}_r)$, and hence quasi-concave on $\widetilde{Q}_r$. Hence, compared with $g(x)$, $\min\{g(x),K+g(\widetilde{S}^r)\}$ can only introduce at most one more convex piece in the $\widetilde{Q}_r$, whose range is $\{x\in\widetilde{Q}_r\,|\,g(x)\ge K+g(\widetilde{S}^r)\}$ and the evaluation of $h(x)$ on it is constantly $K+g(\widetilde{S}^r)$. In conclusion, compared with $g(x)$, $h(x)$ has at most R additional convex pieces (actually flat pieces), which means that $h(x)$ is piecewise convex with at most $L+R$ number of convex pieces. □

C.3. Proof of Lemma 4

LEMMA 4. *For any $i\in N_{\mathcal{T}}$ and any convex piece P_ℓ^i with $\ell\in[L_i]$ in the DP representation of $V(i,\cdot)$, there are a constant $K_\ell^i\in\mathbb{R}$ and a set $\mathcal{D}_\ell^i\subseteq N_{\mathcal{T}(i)}$ such that*

$$V(i,x)=K_\ell^i+\sum_{j\in\mathcal{N}_\ell^i} f_j(x), \qquad \forall x\in P_\ell^i,$$

where $\mathcal{N}_\ell^i = \bigcup_{k\in\mathcal{D}_\ell^i} \mathcal{P}(i,k)$*, which might be an empty set. The same representation holds for each convex piece in the DP representation of* $U(i,\cdot)$*, with* $i \in \mathcal{N}_\ell^i$*.*

Proof. We use induction. At a leaf node i, $U(i,x) = f_i(x)$, while each piece of $V(i,\cdot)$ is either a restriction of f_i or a constant. Suppose the claim holds for all children of i. On each interval formed by the child DP breakpoints, write $V(j,x) = K^j + \sum_{k\in\mathcal{N}^j} f_k(x)$, where $\mathcal{N}^j = \bigcup_{k\in\mathcal{D}^j} \mathcal{P}(j,k)$. Then

$$U(i,x) = \sum_{j\in\mathcal{C}(i)} K^j + \sum_{k\in\{i\}\cup\bigcup_{j\in\mathcal{C}(i)}\mathcal{N}^j} f_k(x).$$

The node set in this expression retains the required path structure. Indeed, for each child $j \in \mathcal{C}(i)$ and each $k \in \mathcal{D}^j$, the path from i to k consists of node i followed by the path from j to k, so $\mathcal{P}(i,k) = \{i\} \cup \mathcal{P}(j,k)$. Defining $\mathcal{D}^i = \{i\} \cup \bigcup_{j\in\mathcal{C}(i)} \mathcal{D}^j$, we therefore obtain $\bigcup_{k\in\mathcal{D}^i} \mathcal{P}(i,k) = \{i\} \cup \bigcup_{j\in\mathcal{C}(i)} \mathcal{N}^j$. Here, $\mathcal{P}(i,i) = \{i\}$, so the representation also covers the case in which all child node sets are empty. The update (6) to obtain $V(i,x)$ either retains a restriction of this expression or introduces a constant piece with an empty node set. This proves the lemma. □

C.4. Proof of Lemma 5

LEMMA 5. *For any* $i \in [n]$ *and any local minimizer* m *of* $U(i,\cdot)$*, there exists a nonempty set* $\mathcal{N}^i = \bigcup_{k\in\mathcal{D}^i} \mathcal{P}(i,k)$*, where* $\mathcal{D}^i \subseteq N_{\mathcal{T}(i)}$*, such that* $m \in \arg\min_{x\in\mathbb{R}} \sum_{j\in\mathcal{N}^i} f_j(x)$*.*

Proof. By Lemma 4, each convex piece of $U(i,\cdot)$ has the form $K + \sum_{j\in\mathcal{N}} f_j(x)$. Let m be a local minimizer. If m is not a breakpoint, the desired result follows directly from convexity. Otherwise, suppose m is a breakpoint, let x_j^* for $j \in N_{\mathcal{T}(i)}$ be optimal decisions attaining $U(i,m)$, with $x_i^* = m$. Set $\mathcal{N}^i = \{j \in N_{\mathcal{T}(i)} : x_j^* = m\}$. For any $j \in \mathcal{N}^i$ and $k \in \mathcal{P}(i,j)$, $m = x_i^* \le x_k^* \le x_j^* = m$. Hence, $\mathcal{N}^i$ is nonempty and $\mathcal{N}^i = \bigcup_{k\in\mathcal{D}^i} \mathcal{P}(i,k)$ with $\mathcal{D}^i = \mathcal{N}^i$. Let A collect the fixed costs and the node costs outside $\mathcal{N}^i$ in this solution. For sufficiently close t, replace the value m at every node in $\mathcal{N}^i$ by t, leaving the other decisions unchanged. Equalities inside $\mathcal{N}^i$ are preserved. On each edge leaving $\mathcal{N}^i$, the child value exceeds m, so the inequality remains strict for t sufficiently close to m. All other constraints are unchanged. Thus, the modified solution is feasible for $U(i,t)$ and has cost $A + \sum_{j\in\mathcal{N}^i} f_j(t)$. Together with local minimality, this gives

$$A + \sum_{j\in\mathcal{N}^i} f_j(m) = U(i,m) \le U(i,t) \le A + \sum_{j\in\mathcal{N}^i} f_j(t).$$

Therefore, m locally minimizes $\sum_{j\in\mathcal{N}^i} f_j(x)$ and, by convexity, is a global minimizer. □

C.5. Proof of Lemma 6

LEMMA 6. *Let $\mathcal{P} := \mathcal{P}(1,j)$ for some $j \in \mathcal{L}$, and re-index its nodes from the root to the leaf as $1,2,\ldots,|\mathcal{P}|$. For each $\ell \in (\bigcup_{k\in\mathcal{P}} \mathcal{C}(k)) \setminus \mathcal{P}$, let m_ℓ denote the number of convex pieces of $V(\ell,\cdot)$. Then, for every $i \in \mathcal{P}$, the number of convex pieces of $V(i,\cdot)$ is at most*

$$\left(1+\frac{n(n+1)}{2}\right)\left(1+\sum_{\ell\in(\bigcup_{k\in\mathcal{P}}\mathcal{C}(k))\setminus\mathcal{P}} m_\ell\right).$$

Proof. To prove the lemma, let us first re-express the dynamic programming recursion for $V(i,x)$ for each $i \in \mathcal{P}$ by grouping the directly-incurred cost at i (i.e., $f_i(x)$) and all value-to-go functions of the child nodes of i except $i+1$ (i.e., $\sum_{j\in\mathcal{C}(i)\setminus\{i+1\}} V(j,x)$) together:

$$\begin{aligned}
V(i,x) &= \min_{x'\geq x}\left\{K_i\mathbb{1}\{x'>x\} + U(i,x')\right\}\\
&= \min_{x'\geq x}\left\{K_i\mathbb{1}\{x'>x\} + f_i(x') + \sum_{j\in\mathcal{C}(i)} V(j,x')\right\}\\
&= \min_{x'\geq x}\left\{K_i\mathbb{1}\{x'>x\} + \left(f_i(x') + \sum_{j\in\mathcal{C}(i)\setminus\{i+1\}} V(j,x')\right) + V(i+1,x')\right\}\\
&= \min_{x'\geq x}\left\{K_i\mathbb{1}\{x'>x\} + \widetilde{f}_i(x') + V(i+1,x')\right\} \qquad \left(\because \widetilde{f}_i(x') := f_i(x') + \sum_{j\in\mathcal{C}(i)\setminus\{i+1\}} V(j,x')\right)
\end{aligned} \tag{EC.2}$$

where we view node $|\mathcal{P}|+1$ as the only child of node $|\mathcal{P}|$ and let $V(|\mathcal{P}|+1,x) \equiv 0$ for all x. It can be observed from (EC.2) that getting the functional representation of $V(i,x)$ for all $i \in \mathcal{P}$ is the same as doing isotonic optimization on only a path with nodes of $\mathcal{P}$, except that each convex cost function $f_i(x)$ should be substituted by a piecewise convex cost function $\widetilde{f}_i(x)$ which has at most $\max\{1, \sum_{j\in\mathcal{C}(i)\setminus\{i+1\}} m_j\}$ number of convex pieces. We bound all quasi-convex pieces of $U(i,x) = \widetilde{f}_i(x) + V(i+1,x)$ together. First suppose that the spine costs f_k for any $k \in \mathcal{P}$ are strictly convex and coercive, leaving the off-spine subproblems unchanged. A limiting argument at last removes this temporary restriction.

Step A: Bounding the number of quasi-convex pieces of $U(i,\cdot)$. Write $p = |\mathcal{P}|$, and let R_i denote the number of quasi-convex pieces of $U(i,\cdot)$. Since $U(i,\cdot)$ is continuous, coercive, and strictly convex on each convex piece, the minimality condition in Definition 3 ensures that its quasi-convex pieces have distinct local minima. We adapt the perturbation argument in the proof of Lemma 5. Let m be a local minimizer of $U(i,\cdot)$, and choose optimal spine decisions x_k^* for $k = i,\ldots,p$ attaining $U(i,m)$, with $x_i^* = m$. Let j be the first subsequent spine node with $x_j^* > m$, taking $j = p+1$ if no such node exists. Thus, $x_i^* = \cdots = x_{j-1}^* = m$. For sufficiently close t, replace these common values by t, reoptimize their attached off-spine subproblems, and keep the remaining spine decisions fixed.

The equalities along nodes $i,\dots,j-1$ are preserved; when $j\le p$, the next isotonic inequality remains strict because $t<x_j^*$. Hence, feasibility and all fixed-cost indicators along the spine are preserved. For a constant A independent of t, local minimality and the resulting feasible solution give

$$A+\sum_{k=i}^{j-1}\widetilde{f}_k(m)=U(i,m)\le U(i,t)\le A+\sum_{k=i}^{j-1}\widetilde{f}_k(t).$$

Therefore, m locally minimizes $\sum_{k=i}^{j-1}\widetilde{f}_k(x)$. Partitioning at the breakpoints of the off-spine value functions shows that this sum has at most $\max\{1,\sum_{k=i}^{j-1}\sum_{\ell\in\mathcal{C}(k)\setminus\{k+1\}}m_\ell\}$ convex pieces, each of which is strictly convex. By continuity, finite endpoints may be included in these pieces. Any local minimum of the sum is a global minimizer of its restriction to each piece containing that point, and strict convexity makes this minimizer unique. This applies equally to smooth minima, non-smooth minima, and minima at breakpoints. Consequently, each convex piece accounts for at most one local minimum. Summing over the possible values of j gives

$$R_i\le\sum_{j=i+1}^{p+1}\max\left\{1,\sum_{k=i}^{j-1}\sum_{\ell\in\mathcal{C}(k)\setminus\{k+1\}}m_\ell\right\}.$$

Step B: Bounding the number of convex pieces of $V(i,\cdot)$. Let C_i denote the number of convex pieces in the DP representation of $V(i,\cdot)$, with $C_{p+1}=1$, and set $b_i:=\sum_{\ell\in\mathcal{C}(i)\setminus\{i+1\}}(m_\ell-1)$. Combining the breakpoints inherited from $V(i+1,\cdot)$ and the off-spine children gives at most $C_{i+1}+b_i$ convex pieces for $U(i,\cdot)$. Lemma 3 therefore yields $C_i\le C_{i+1}+b_i+R_i$. Write $q=p-i+1$. Summing along the spine and using Step A, we obtain

$$\begin{aligned}
C_i&\le 1+\sum_{i'=i}^{p}b_{i'}+\sum_{i'=i}^{p}R_{i'}\\
&\le 1+\sum_{i'=i}^{p}b_{i'}+\sum_{i'=i}^{p}\sum_{j=i'+1}^{p+1}\max\left\{1,\sum_{k=i'}^{j-1}\sum_{\ell\in\mathcal{C}(k)\setminus\{k+1\}}m_\ell\right\}\\
&\le\left(1+\frac{q(q+1)}{2}\right)\left(1+\sum_{\ell\in\left(\bigcup_{k\in\mathcal{P}}\mathcal{C}(k)\right)\setminus\mathcal{P}}m_\ell\right)\\
&\le\left(1+\frac{n(n+1)}{2}\right)\left(1+\sum_{\ell\in\left(\bigcup_{k\in\mathcal{P}}\mathcal{C}(k)\right)\setminus\mathcal{P}}m_\ell\right).
\end{aligned}$$

The third inequality counts $q(q+1)/2$ pairs (i',j) and bounds each inner sum by the total off-spine piece count, and the last inequality holds because $q\le n$.

Finally, set $\varepsilon>0$, replace $f_k(x)$ by $f_k^\varepsilon(x):=f_k(x)+\varepsilon x^2$ for $k\in\mathcal{P}$, and let V^ε denote the resulting value functions. The perturbed node costs are strictly convex and coercive, while all off-spine value functions and their piece counts remain unchanged. Thus, the integer bound M on the number of

pieces established above is independent of ε. For each fixed state x, evaluating the perturbed objective at optimal decisions $z_j^*(x)$ of the original subproblem gives

$$0 \le V^\varepsilon(i,x) - V(i,x) \le \varepsilon \sum_{k \in \mathcal{P} \cap N_{\mathcal{T}(i)}} (z_k^*(x))^2 \to 0 \quad \text{as} \quad \varepsilon \downarrow 0$$

Along a sequence $\varepsilon \downarrow 0$, pass to a subsequence with a fixed number of pieces, convergent breakpoints in the extended real line, and fixed node-index sets from Lemma 4. This is possible because there are at most M pieces and finitely many possible index sets. On each limiting interval with nonempty interior, let $\mathcal{N}$ denote the corresponding fixed index set. Every fixed interior point x eventually satisfies $V^\varepsilon(i,x) = A^\varepsilon + \sum_{j \in \mathcal{N}} f_j(x) + \varepsilon |\mathcal{N} \cap \mathcal{P}| x^2$. Convergence at one interior point implies $A^\varepsilon \to A$ for some $A \in \mathbb{R}$, so $V(i,x) = A + \sum_{j \in \mathcal{N}} f_j(x)$ throughout the interval. This expression is convex, and continuity extends it to finite endpoints. Discarding collapsed intervals therefore yields at most M convex pieces, retaining the representation in Lemma 4. □

C.6. Proof of Theorem 4

Proof. Let C_i denote the number of convex pieces of $V(i,\cdot)$, and write $\mathcal{M}(t) = 1 + t(t+1)/2$ for nonnegative integers t. We prove the claim by induction on k. If $k = 1$, the arborescence is a directed path, and Lemma 6 gives the bound $\mathcal{M}(n)$. For $k \ge 2$, choose a root-to-leaf path $\mathcal{P}$ by always following a child of maximum Horton–Strahler number. For each $\ell \in (\bigcup_{v \in \mathcal{P}} \mathcal{C}(v)) \setminus \mathcal{P}$, the subtree $\mathcal{T}(\ell)$ has Horton–Strahler number at most $k-1$, so induction bounds C_ℓ by $\mathcal{M}(|N_{\mathcal{T}(\ell)}|)^{k-1}$. For positive integers a, b, it holds that $\mathcal{M}(a+b) - \mathcal{M}(a) - \mathcal{M}(b) = ab - 1 \ge 0$. Since the off-spine subtrees are disjoint and contain at most $n-1$ nodes in total, it follows that $1 + \sum_{\ell \in (\bigcup_{v \in \mathcal{P}} \mathcal{C}(v)) \setminus \mathcal{P}} \mathcal{M}(|N_{\mathcal{T}(\ell)}|) \le 1 + \mathcal{M}(n-1) \le \mathcal{M}(n)$, where the inequality also holds when there are no off-spine subtrees. Thus, for every $i \in \mathcal{P}$, Lemma 6 and $k - 1 \ge 1$ give

$$\begin{aligned} C_i &\le \mathcal{M}(n) \left(1 + \sum_{\ell \in (\bigcup_{v \in \mathcal{P}} \mathcal{C}(v)) \setminus \mathcal{P}} \mathcal{M}(|N_{\mathcal{T}(\ell)}|)^{k-1} \right) \\ &\le \mathcal{M}(n) \left(1 + \sum_{\ell \in (\bigcup_{v \in \mathcal{P}} \mathcal{C}(v)) \setminus \mathcal{P}} \mathcal{M}(|N_{\mathcal{T}(\ell)}|) \right)^{k-1} \le \mathcal{M}(n)^k. \end{aligned}$$

Nodes outside $\mathcal{P}$ satisfy the same bound by induction. This completes the proof. □

Appendix D: Proofs of Technical Results in Section 5

D.1. Proof of Proposition 6

PROPOSITION 6. *For every integer $r \ge 0$, there exists an instance on an arborescence with at most two children per node and $n_r = 2 \cdot 4^r$ nodes such that the root value function $V(1,\cdot)$ requires at least $2^{r^2+1} + 1 = n_r^{\Omega(\log n_r)}$ convex interval pieces.*

Proof. For a root function U before its fixed-cost update, write $\mathcal{D}_K U(x) = \min\{U(x), K + \min_{z \geq x} U(z)\}$. We recursively construct groups of selected local minimizers of U. Each group consists of $C \geq 2$ equally spaced points $c, c+4, \ldots, c+4(C-1)$ with a common value H. For every point b in the group, we require: (i) $U(x) = H + (x-b)^2$ for $|x-b| \leq 2$. Neighboring intervals $[b-2, b+2]$ meet at their endpoints, so their union within each group is a connected interval. (ii) $U(y) \geq H$ for every $y \geq b$. This bound applies to all points to the right, including gaps between groups, not just to other selected minima. Different groups are separated by gaps between their quadratic intervals, across which U remains continuous. The height H is constant within each group and nondecreasing across groups from left to right.

Step A. Starting instance. Take a root r with one leaf child ℓ, and set

$$K_\ell = 16, \qquad f_\ell(x) = (x-6)^2, \qquad K_r = 1, \qquad f_r(x) = \begin{cases} x^2 - 16, & x \leq 2, \\ 4x - 20, & x > 2. \end{cases}$$

Both node costs are convex, continuously differentiable, and coercive. Therefore,

$$V(\ell, x) = \begin{cases} 16, & x \leq 2, \\ (x-6)^2, & x > 2, \end{cases} \quad \Longrightarrow \quad U(x) = f_r(x) + V(\ell, x) = \min\{x^2, (x-4)^2\}.$$

The two requirements hold for one group with centers 0 and 4 and common height $H = 0$.

Step B. The Recursive Construction. Suppose the current tree has n nodes, and retain U for its root function before the fixed-cost update. Choose an integer $L \geq 2$ and set $a = 2/(3L)$. Construct a new tree from two copies of the current tree as follows; see Figure EC.1.

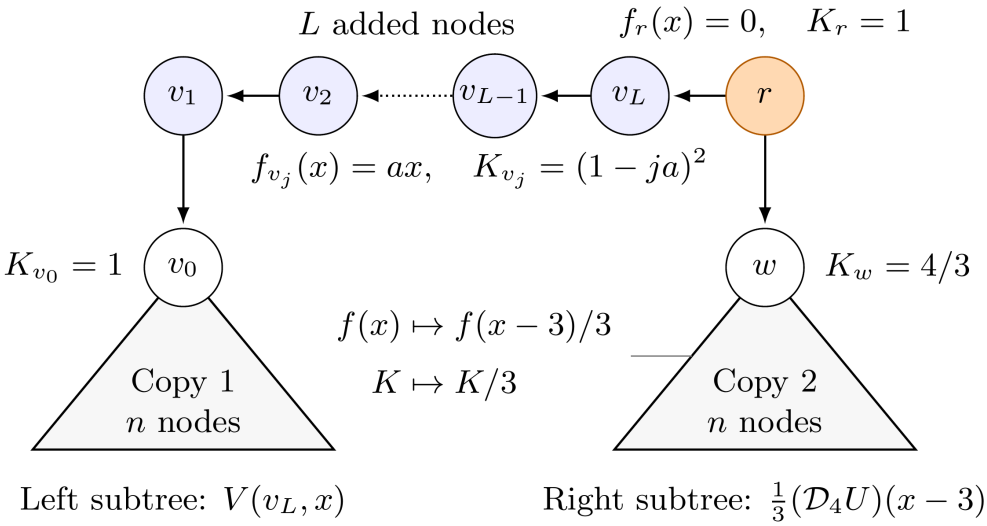


Figure EC.1 The recursive construction. Each triangle contains one entire n-node copy of the old tree, including its root. Only the first copy receives the added path. The two subtrees are joined by node r.

Step B.1. First Copy with an Added Path. Denote the first copy's root by v_0 and attach a path of L new nodes above it, numbered $v_1, \dots, v_L$ from bottom to top. Give each added node v_j the cost $f_{v_j}(x) = ax$, and set $K_{v_j} = (1 - ja)^2$ for $j = 0, \dots, L$. The affine costs of the new nodes are defined temporary; their coercive extensions are specified below. These fixed costs lie in $[1/9, 1]$, with $K_{v_0} = 1$ and $K_{v_L} = 1/9$. All other costs in this copy remain unchanged. The value function of the resulting subtree is $V(v_L, x)$, which can be expressed in terms of U. An optimal solution has at most one strict increase along the added path, including the transition into v_0. Indeed, if several increases occur, retain the last one, closest to v_0, and lower all preceding path values to x. This preserves the order constraints and all decisions in the old subtree, reduces the linear node costs, and removes the earlier fixed charges. The retained increase incurs the same fixed charge, regardless of its size. With no increase, the cost is $2x/3 + U(x)$. If the sole increase occurs at v_j, to a value $z > x$, then the old root and the bottom j added nodes take value z, while the remaining $L - j$ added nodes take value x. Here $j = 0$ means that only the old root increases. Consequently,

$$V(v_L, x) = \min\left\{\frac{2}{3}x + U(x),\ \min_{0 \le j \le L}\left\{\left(\frac{2}{3} - ja\right)x + K_{v_j} + \min_{z \ge x}\{jaz + U(z)\}\right\}\right\}.$$

Step B.2. Second Copy with Scaled and Shifted Costs. In the second copy, denoting its root by 2, and first set its root fixed cost to 4. Then replace every node cost $f(x)$ by $f(x-3)/3$ and divide every fixed cost by 3. Subtracting 3 from every decision preserves the order constraints and fixed-cost indicators, while all objective terms are divided by 3. Hence,

$$V(w, x) = \frac{1}{3}(\mathcal{D}_4 U)(x - 3).$$

Step B.3. New Root. Add a new root r with $f_r(x) = 0$ and $K_r = 1$, whose children are v_L and w. The new tree has $2n + L + 1$ nodes. The value function at r before the fixed-cost update is

$$U(r, x) = V(v_L, x) + \frac{1}{3}(\mathcal{D}_4 U)(x - 3).$$

Step C. New Minima and Their Preservation. We show that each pair of consecutive old centers produces $L + 1$ new selected minima. On the interval considered below, each fixed increase location in the first copy yields an affine optimized cost. Adding the second copy's quadratic contribution produces quadratics with distinct centers and a common minimum height, whose pointwise minimum creates the new minima. See Figure EC.2 for this process.

Step C.1. Creation of New Minima. Fix consecutive old centers $c, c+4$ in a group of height H, and write $x = c + t$. For $0 \le j \le L$ and $3/2 \le t \le 10/3$, we first establish

$$\min_{z \ge c+t}\{jaz + U(z)\} = H + jac + 4ja - \frac{(ja)^2}{4}. \tag{EC.3}$$

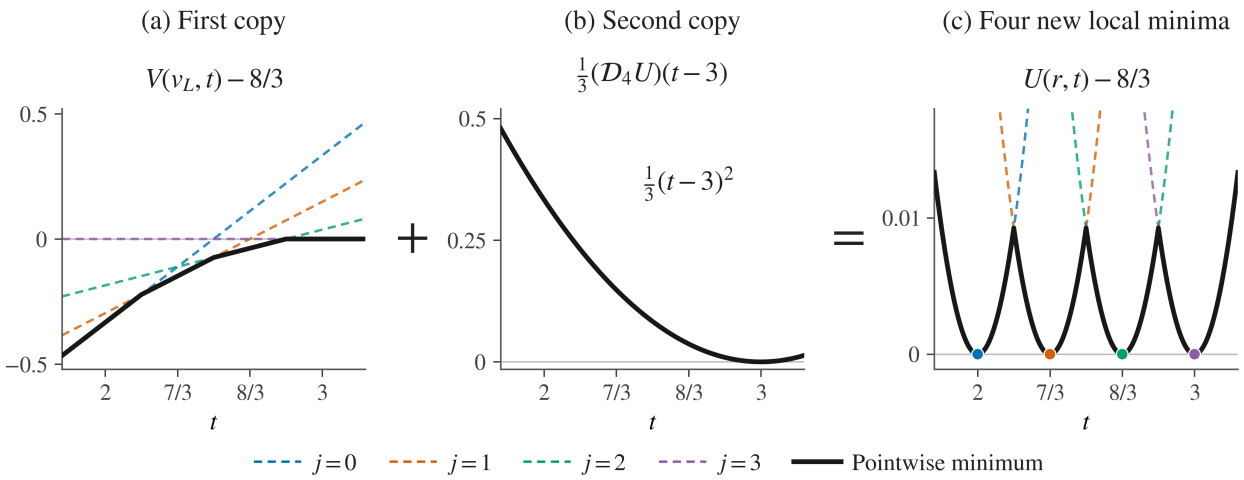


Figure EC.2 Creation of new local minima in Step C.1 with $L=3$ and $c=H=0$. Adding the second copy's quadratic cost to the first copy's affine alternatives produces quadratics with a common minimum height. Their pointwise minimum has four local minima. Matching colors indicate the same increase location j. Panels (a) and (c) subtract $H_{\text{new}}=8/3$; panel (c) uses a magnified vertical scale.

by examining the feasible values of z in three cases.

(i) For $c+t\le z\le c+2$, which is possible only when $t\le 2$, write $z=c+y$. Since $U(z)=H+y^2$, $y\ge t\ge 3/2$, and $ja\le 2/3$, then $jaz+U(z)$ exceeds the right-hand side of (EC.3) by

$$y^2+ja(y-4)+\frac{(ja)^2}{4}\ge\frac{9}{4}-\frac{5}{2}ja\ge\frac{7}{12}>0.$$

(ii) For $c+2\le z\le c+4$ with $z\ge c+t$, we have $U(z)=H+(z-c-4)^2$. Rearranging yields

$$jaz+U(z)=H+jac+4ja-\frac{(ja)^2}{4}+\left(z-c-4+\frac{ja}{2}\right)^2.$$

The minimum is attained at $z=c+4-ja/2$, which is feasible $4-ja/2\ge 4-1/3=11/3>t$.

(iii) For $z\ge c+4$, the old lower bound $U(z)\ge H$ gives

$$jaz+U(z)\ge H+jac+4ja\ge H+jac+4ja-\frac{(ja)^2}{4}.$$

These comparisons prove (EC.3). Substituting (EC.3) and $K_{v_j}=(1-ja)^2$ into Step B.1, the alternative with an increase at v_j has cost $H+2c/3+1+2ja+3(ja)^2/4+(2/3-ja)t$. The no-increase alternative is no cheaper. To see this, we consider two cases: for $3/2\le t\le 3$, we have $U(c+t)-H\ge 1$, whose cost $2(c+t)/3+U(c+t)$ is at least $H+2(c+t)/3+1$, the cost of the alternative $j=0$; for $3\le t\le 10/3$, the no-increase cost exceeds the cost $H+2c/3+8/3$ of the alternative $j=L$ by $(t-10/3)(t-4)\ge 0$. Consequently,

$$V(v_L,c+t)=H+\frac{2}{3}c+1+\min_{0\le j\le L}\left\{2ja+\frac{3}{4}(ja)^2+\left(\frac{2}{3}-ja\right)t\right\}.$$

We next determine the contribution of the second copy. For $3/2 \le t \le 10/3$, the point $c+t-3$ lies in the quadratic interval around c, so $U(c+t-3) = H + (t-3)^2 < H+4$. Besides, $U(y) \ge H$ for every $y \ge c-2$. Thus, paying the fixed cost 4 cannot improve this value, implying

$$(\mathcal{D}_4 U)(c+t-3) = H + (t-3)^2.$$

Adding this contribution as in Step B.3, substituting $ja = 2j/(3L)$, and rearranging gives

$$U(r, c+t) = H_{\text{new}} + \frac{1}{3} \min_{0 \le j \le L} \left(t - 2 - \frac{j}{L} \right)^2 \tag{EC.4}$$

for any $3/2 \le t \le 10/3$, where $H_{\text{new}} = (4H + 2c + 8)/3$. Hence, the pair $c, c+4$ produces $L+1$ selected centers $c + 2 + j/L$, $j = 0, \ldots, L$, all with height H_{new}. Within distance $1/(2L)$ of each center, its quadratic is no larger than the others. Each center therefore has a quadratic neighborhood of radius $1/(2L)$ and coefficient $1/3$. All these neighborhoods lie within the interval in (EC.4) because $L \ge 2$.

Step C.2. Preservation of Condition (ii). We verify condition (ii) for the new selected centers by showing that $U(r, c+t) \ge H_{\text{new}}$ for every $t \ge 2$.

(i) For $2 \le t \le 10/3$, the bound follows directly from (EC.4).

(ii) For $10/3 \le t \le 4$, the right-hand side of (EC.3) remains a lower bound because the feasible set $z \ge c+t$ shrinks as t increases. Since $(\mathcal{D}_4 U)(c+t-3) = H + (t-3)^2$ still holds, adding the second copy's contribution and rearranging the increase and no-increase alternatives gives

$$U(r, c+t) \ge H_{\text{new}} + \min \left\{ \frac{4}{3} \left(t - \frac{7}{2} \right)^2, \frac{1}{3} \min_{0 \le j \le L} \left(t - 2 - \frac{j}{L} \right)^2 \right\} \ge H_{\text{new}}.$$

(iii) For $t \ge 4$, every added-path decision is at least $c+t$, and the old subtree contributes at least H. Since fixed costs are nonnegative, the first copy with its added path costs at least $H + 2(c+t)/3$. The second copy contributes at least $H/3$, giving

$$U(r, c+t) \ge \frac{4}{3} H + \frac{2}{3}(c+t) \ge \frac{4H + 2c + 8}{3} = H_{\text{new}}.$$

The new groups also remain separated. For a pair $(c, c+4)$, the selected quadratic neighborhoods have union $[c+2-1/(2L),\ c+3+1/(2L)]$, whose length is at most $3/2$. For distinct old pairs $(c, c+4)$ and $(c', c'+4)$ with $c < c'$, we have $c' - c \ge 4$, so their corresponding new intervals are disjoint. Moreover, the new heights are nondecreasing from left to right, since $H_{\text{new}} = (4H + 2c + 8)/3$ increases with c and the old group heights H are nondecreasing. Thus, an old group of C centers supplies $C-1$ distinct new groups, each containing $L+1$ selected minima. The same two copies and added path produce all $(C-1)(L+1)$ minima simultaneously.

Step C.3. Restoration of Condition (i). The quadratic neighborhoods obtained in Step C.1 have coefficient $1/3$ and radius $1/(2L)$. We rescale the new instance to restore condition (i) and the spacing of four between consecutive centers. After the coercive modification described introduced later, which preserves the local formulas and lower bounds above, set

$$X = 4L(x-2), \qquad \widehat{U}(X) = 48L^2 U\left(r, 2 + \frac{X}{4L}\right).$$

The centers $c+2+j/L$ become $4Lc+4j$, restoring spacing four, and their quadratic neighborhoods have radius two. For $|X-(4Lc+4j)| \le 2$, the local formula becomes $\widehat{U}(X) = 48L^2 H_{\text{new}} + \big(X - (4Lc+4j)\big)^2$. Hence condition (i) holds at each new center, with common height $48L^2 H_{\text{new}}$ within each group. The increasing change of variable and positive cost scaling also preserve condition (ii), group separation, and height ordering. The instance therefore satisfies all requirements for the next recursive round. This normalization is implemented by applying the same change of variable to every node cost and multiplying all node costs and fixed costs by $48L^2$.

Step D. Admissible Costs and Counting. At each recursive round, before normalization in Step C.3, add $\operatorname{dist}(x,[\ell,u])^2$ to the cost of each new path node and the new root, where $[\ell,u]$ contains all selected old centers. For each state $x = c+t$ covered by (EC.4), an optimal solution used in Step C.1 assigns every new node either x or $c+4-ja/2$, both in $[c,c+4] \subseteq [\ell,u]$. The added penalties are nonnegative for every feasible solution and vanish on this optimal solution, so the optimal value remains unchanged. Thus, (EC.4) and the lower bounds in Step C.2 are preserved. Thereafter, all modified node costs are convex, continuously differentiable, and coercive, and remain so after shifting and positive scaling. All fixed costs are positive.

Let n_r and M_r denote the numbers of nodes and selected minima after round r, with $n_0 = M_0 = 2$. Choose $L = 2n_{r-1} - 1$ in round $r \ge 1$. Then $n_r = 2n_{r-1} + L + 1 = 4n_{r-1} = 2 \cdot 4^r$. Each old group of $C \ge 2$ centers produces $(C-1)(L+1) \ge (C/2)(L+1) \ge Cn_{r-1}$ selected minima. Summing over all old groups gives

$$M_r \ge n_{r-1} M_{r-1} \ge 2 \prod_{h=0}^{r-1} (2 \cdot 4^h) = 2^{r^2+1}.$$

Finally, relabel the normalized root as node 1 and set $K_1 = 1$. For each selected center b of height H, conditions (i) and (ii) imply $V(1,x) = H + \min\{(x-b)^2, 1\}$ for $b-2 \le x \le b$. Its derivative jumps downward from 0 to -2 at $b-1$, so this point must be a boundary between convex interval pieces. The M_r distinct such boundaries therefore force at least $M_r + 1 \ge 2^{r^2+1} + 1$ pieces. Since $n_r = 2 \cdot 4^r$, which implies $r = (\log_2 n_r - 1)/2$, this gives the claimed $n_r^{\Omega(\log n_r)}$ lower bound. $\square$

Appendix E: Supplementary Tables and Algorithms

E.1. Summary of Algorithms (Table EC.1)

Table EC.1 Summary of Algorithms

Graph structure	Case	Algorithm
Path	General convex costs	Algorithm 1
	Isotonic minima	Accelerated DP (Section 2.2)
Arborescence	Piecewise linear convex costs	Algorithm 2
	Dominating parent fixed costs	Algorithm 3
	Logarithmic height	Algorithm 3
	Bounded Horton–Strahler number	Algorithm 3
	General convex costs	Algorithm 3

E.2. DP Implementation for Piecewise Linear Costs (Algorithm 2)

Algorithm 2 Efficient Implementation of DP Recursion (DP-PL)

Input: f_i's and their corresponding kink points set $(\{k_1^i, \ldots, k_{m_i}^i\})$'s, for $i \in [n]$
$\overline{\mathcal{K}} \leftarrow$ the decreasingly sorted list of $\bigcup_{i\in[n]}\{k_1^i, \ldots, k_{m_i}^i\}$
procedure DFS(i)
  for $j \in \mathcal{C}(i)$ **do**
    DFS(j)
  end for
  $V_{\rightarrow}(i, \overline{\mathcal{K}}[0]) \leftarrow f_i(\overline{\mathcal{K}}[0]) + \sum_{j\in\mathcal{C}(i)} V(j, \overline{\mathcal{K}}[0])$ ▷ Indices start at 0
  $V_{\uparrow}(i, \overline{\mathcal{K}}[0]) \leftarrow \infty$
  $V(i, \overline{\mathcal{K}}[0]) \leftarrow \min\{V_{\uparrow}(i, \overline{\mathcal{K}}[0]), V_{\rightarrow}(i, \overline{\mathcal{K}}[0])\}$
  for $\ell \in [|\overline{\mathcal{K}}| - 1]$ **do**
    $V_{\rightarrow}(i, \overline{\mathcal{K}}[\ell])) \leftarrow f_i(\overline{\mathcal{K}}[\ell]) + \sum_{j\in\mathcal{C}(i)} V(j, \overline{\mathcal{K}}[\ell])$
    $V_{\uparrow}(i, \overline{\mathcal{K}}[\ell]) \leftarrow \min\{V_{\uparrow}(i, \overline{\mathcal{K}}[\ell-1]), K_i + f_i(\overline{\mathcal{K}}[\ell-1]) + \sum_{j\in\mathcal{C}(i)} V(j, \overline{\mathcal{K}}[\ell-1])\}$
    $V(i, \overline{\mathcal{K}}[\ell]) \leftarrow \min\{V_{\uparrow}(i, \overline{\mathcal{K}}[\ell]), V_{\rightarrow}(i, \overline{\mathcal{K}}[\ell])\}$
  end for
end procedure
Execute DFS(1)
$V(1, -\infty) \leftarrow K_1 + \min_{k\in\mathcal{K}} \left\{ f_1(k) + \sum_{j\in\mathcal{C}(1)} V(j, k) \right\}$
Output: the optimal decisions towards $V(1, -\infty)$

E.3. DP Implementation for General Convex Costs (Algorithm 3)

Algorithm 3 General Arborescence DP with Convex-Piece Lists

Input: Arborescence $\mathcal{T}$, convex costs f_i, fixed costs $K_i \geq 0$, and $x_0 = -\infty$.
Subprocedures: Algorithms 4 and 5.
for each node i in postorder **do**
  $\widetilde{\text{List}}_i \leftarrow \textsc{MergeLists}(i, \{\text{List}_j\}_{j\in\mathcal{C}(i)})$.
  $\text{List}_i \leftarrow [\,]$; $(\mu, z) \leftarrow (+\infty, +\infty)$. ▷ Best value and attaining point to the right
  for each entry $(a, b, L, \mathcal{J})$ of $\widetilde{\text{List}}_i$, from right to left **do**
    Let $g(x) = L + \sum_{j\in\mathcal{J}} f_j(x)$ and choose a finite $\widehat{x} \in \arg\min_{x\in\text{cl}((a,b))} g(x)$.
    Prepend the block $\textsc{Clip}(\widehat{x}, b, L, \mathcal{J}, K_i + \mu, z)$ to List_i. ▷ $\widehat{x}$ is infeasible when $x > \widehat{x}$.
    If $g(\widehat{x}) < \mu$, set $(\mu, z) \leftarrow (g(\widehat{x}), \widehat{x})$.
    Prepend the block $\textsc{Clip}(a, \widehat{x}, L, \mathcal{J}, K_i + \mu, z)$ to List_i. ▷ $\widehat{x}$ is now feasible when $x < \widehat{x}$.
  end for
  Merge adjacent entries with identical $(L, \mathcal{J})$, preserving the stored target for flat pieces.
end for
$\text{OPT} \leftarrow K_1 + \mu$.
Output: OPT and $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ recovered from $\{\text{List}_i\}_{i\in[n]}$ and the stored minimizing points.

Algorithm 4 MergeLists

procedure $\textsc{MergeLists}(i, \{\text{List}_j\}_{j\in\mathcal{C}(i)})$
  $B \leftarrow$ the sorted distinct child-list breakpoints, together with $-\infty$ and $+\infty$.
  for each consecutive pair $a < b$ in B **do**
    For each $j \in \mathcal{C}(i)$, let $(L_j, \mathcal{J}_j)$ be the coefficients of the child entry covering (a, b).
    Create the entry $(a, b, \sum_{j\in\mathcal{C}(i)} L_j, \{i\} \cup \bigcup_{j\in\mathcal{C}(i)} \mathcal{J}_j)$.
  end for
  return the created entries in increasing order of left endpoints.
end procedure

Algorithm 5 Clip

procedure $\textsc{Clip}(a, b, L, \mathcal{J}, c, z)$
  Let $g(x) = L + \sum_{j\in\mathcal{J}} f_j(x)$.
  Find $D = \{x \in (a, b) : g(x) \leq c\}$, taking $D = (a, b)$ if $c = +\infty$.
  If D has positive length, create $(\inf D, \sup D, L, \mathcal{J})$.
  For each nonempty interval component (s, t) of $(a, b) \setminus D$, create $(s, t, c, \emptyset)$ with target z.
  return the created entries in increasing order of left endpoints.
end procedure

Appendix F: Additional Numerical Experiments

F.1. Sensitivity to the Horton–Strahler Number

We fix $n = 2047$ and vary the Horton–Strahler number $k \in \{1, 3, 5, 7, 9, 11\}$. For each k, we start with a complete binary tree of $2^k - 1$ nodes and randomly distribute the remaining nodes among unary chains inserted along its edges or above its root. These insertions preserve k. Nodes are indexed with parents preceding their children. As in Table 2, we use $f_i(x) = a_i x^2 - 2b_i x$, with K_i sampled independently from $\mathrm{Unif}(0, 60)$ and a_i and b_i sampled independently from $\mathrm{Unif}(0, 20)$ before being sorted in decreasing and increasing order, respectively. We generate three paired replicates, using the same cost coefficients across all k within each replicate, and apply the same list-based DP implementation. Let C_i denote the number of convex pieces of $V(i, \cdot)$. Table EC.2 reports the average root and maximum piece counts, together with the leaf count and average tree height (measured in edges).

Table EC.2 Sensitivity to the Horton–Strahler Number ($n = 2047$)

k	Leaves	Mean height	Mean C_1	Mean $\max_{i\in[n]} C_i$
1	1	2046.00	96.00	98.00
3	4	896.33	104.00	106.00
5	16	346.67	129.67	131.33
7	64	128.67	217.33	219.67
9	256	47.33	425.33	426.00
11	1024	10.00	681.67	681.67

At the same tree size, the average maximum piece count increases from 98.00 to 681.67, approximately sevenfold, as k increases from 1 to 11. Thus, larger k is associated with substantially larger value-function representations in this family. In this tree family, increasing k also reduces the tree height and increases the number of leaves. The observed increase in piece counts reflects these changes in tree structure induced by increasing k. Since n is fixed, the experiment does not test the worst-case quasi-polynomial growth as n increases.

F.2. Lot-Sizing with Real Sales Data

We compare Algorithm 2 with the big-M formulation on lot-sizing instances derived from the UCI Sales Transactions Weekly dataset, which contains 52 weekly observations for each of 811 products (Tan 2014). To cover different sales volumes, we select ten products at equally spaced ranks of mean weekly sales. Treating sales as a demand proxy, each product's weekly demand takes the means of its lower and upper 26 observations with equal probability, independently across weeks. We use zero initial inventory and lead time, holding cost $h = 1$, backlog cost $b = 5$, and fixed ordering

cost $K = 2\bar{d}$, where $\bar{d}$ is mean weekly sales. These cost parameters are illustrative. We consider $T \in \{2, 4, 6, 8, 10, 12\}$. Open-loop orders are fixed initially, giving a path with T nodes; closed-loop orders depend on past demand, giving a binary history tree with $2^T - 1$ decision nodes. Orders precede the current week's demand in both models. Each node's inventory cost averages over its conditional demand distribution, and both inventory and fixed ordering costs are weighted by the node's realization probability. Both methods solve the same instances with $X_0 = 0$ and incur the root setup cost only when $X_1 > 0$. The big-M formulation uses continuous cumulative orders and the valid bound $M = Td_{\text{high}}$, where d_{high} is the higher weekly demand. The experiments are conducted on the same computer as in Section 6, using Python 3.9.12 and NumPy 1.26.4. For this experiment, the big-M formulation is solved by HiGHS (Huangfu and Hall 2018) through SciPy 1.13.1, with one thread, a 10-second solve limit, and relative gap tolerance 10^{-6}. After initialization, we measure one sequential run per method and instance. Running times include algorithm-specific construction, optimization, solution recovery, and routine checks, but exclude shared scenario generation. Table EC.3 reports means over the ten products, including time-limited runs. "Gap" is the mean relative MIP gap reported by the solver, including zero gaps for solved instances; "Solved" counts instances certified to the requested tolerance. Total running times include overhead outside the solve limit.

Table EC.3 DP and Big-M Running Times on Lot-Sizing Instances

	Open-loop ($n = T$)				Closed-loop ($n = 2^T - 1$)			
T	DP (s)	Big-M (s)	Gap (%)	Solved	DP (s)	Big-M (s)	Gap (%)	Solved
2	0.00047	0.00210	0.00	10/10	0.00056	0.00246	0.00	10/10
4	0.00061	0.00804	0.00	10/10	0.00173	0.06202	0.00	10/10
6	0.00079	0.02125	0.00	10/10	0.00617	2.80395	0.00	10/10
8	0.00098	0.03390	0.00	10/10	0.01786	7.03673	8.91	3/10
10	0.00124	0.08687	0.00	10/10	0.07310	7.26047	13.20	3/10
12	0.00150	0.17007	0.00	10/10	0.30727	9.21975	22.07	3/10

The DP has lower mean running times at every tested horizon in both models. For $T = 12$, it solves the closed-loop instances with 4,095 nodes in 0.307 seconds on average, whereas seven of ten big-M runs reach the time limit, with an overall mean gap of 22.07%. Whenever big-M certifies an optimum, its objective agrees with the DP value within relative tolerance 10^{-7}. As a secondary observation, closed-loop decisions reduce mean expected cost by 15.34% at $T = 12$ under these fitted demand models.