Solving Convex Quadratic Optimization with Indicators Over Structured Graphs

Aaresh Bhathena* Salar Fattahi† Andrés Gómez‡ Simge Küçükyavuz§

Abstract

This paper studies convex quadratic minimization problems in which each continuous variable is coupled with a binary indicator variable. We focus on the structured setting where the Hessian matrix of the quadratic term is positive definite and exhibits sparsity. We develop an exact parametric dynamic programming algorithm whose computational complexity depends explicitly on the treewidth of the Hessian's support graph, its volume growth, and an appropriate margin parameter. Under suitable structural conditions, the overall complexity scales linearly with the problem dimension. To demonstrate the practical impact of our approach, we introduce a novel framework for joint forecasting and outlier detection by extending exponential smoothing to time series with outliers. Computational experiments on both synthetic and real data sets show that our method significantly outperforms state-of-the-art solvers.

Keywords: Mixed-integer quadratic programming, dynamic programming, outlier detection, exponential smoothing.

1 Introduction

We consider the following mixed-integer quadratic program (MIQP), defined by a symmetric and positive definite matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and vectors $\boldsymbol{\lambda}, \boldsymbol{c} \in \mathbb{R}^n$:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{z}\in\{0,1\}^n} \quad \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top \boldsymbol{z} \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad i = 1, 2, \ldots, n. \tag{1b}$$

In this problem, the binary vector $\boldsymbol{z} \in \{0,1\}^n$ encodes the support of the continuous vector $\boldsymbol{x} \in \mathbb{R}^n$. Specifically the constraint $\boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0$ enforces that $\boldsymbol{x}_i = 0$ whenever $\boldsymbol{z}_i = 0$, and $\boldsymbol{z}_i = 1$ allows $\boldsymbol{x}_i \in \mathbb{R}$ to be unconstrained. The vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ acts as the component-wise regularization parameter that promotes sparsity in $\boldsymbol{x}$. We assume throughout that $\boldsymbol{\lambda}_i > 0$ for every $i = 1, \ldots, n$ as $\boldsymbol{\lambda}_i \leq 0$

*Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA. aareshfb@umich.edu

†Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA. fattahi@umich.edu

‡Daniel J. Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA, USA. gomezand@usc.edu

§Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. simge@northwestern.edu

implies that $z_i = 1$ at optimality. Without loss of generality, we also normalize the diagonal entries of $\boldsymbol{Q}$ to one by rescaling each variable $\boldsymbol{x}_i$ as $\boldsymbol{x}_i \to \boldsymbol{x}_i/\sqrt{\boldsymbol{Q}_{i,i}}$.

This work focuses on instances of Problem (1) in which the sparsity pattern of the Hessian matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ coincides with the adjacency matrix of a graph—hereafter referred to as the *support graph*—with certain sparsity structures. Specifically, we consider support graphs characterized by a *bounded treewidth* and a *polynomial volume growth* property. The former captures the extent to which the graph resembles a tree, while the latter imposes an upper bound on the number of nodes contained within any fixed graph distance from a given node. While treewidth is a classical concept in graph theory, polynomial volume growth is a less common but equally meaningful structural property; both notions are formally introduced in Section 2.

Problem (1) arises in several domains, including sparse regression [12, 13, 27], probabilistic graphical models [43, 54, 33, 56, 50, 75, 74], outlier detection in time series [8], and network inference [33, 60]. In this paper, we focus on the problem of forecasting with outlier correction in time series, where the proposed formulation arises naturally and proves particularly effective.

1.1 An overview of our contributions

At the core of our proposed method lies a *pruning* technique that efficiently eliminates suboptimal choices of the binary vector $\boldsymbol{z} \in \{0,1\}^n$, thereby reducing the search space from exponential to polynomial size. Consider the following equivalent formulation of Problem (1):

$$\min_{\substack{\boldsymbol{x}\in\mathbb{R}^n\\ \boldsymbol{z}\in\{0,1\}^n\\ \boldsymbol{x}\circ(1-\boldsymbol{z})=0}} \left\{\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{\lambda}^\top\boldsymbol{z}\right\} = \min_{\boldsymbol{x}\in\mathbb{R}^n}\left\{\min_{\boldsymbol{z}\in\{0,1\}^n}\left\{\frac{1}{2}\boldsymbol{x}^\top\left(\boldsymbol{Q}\circ\boldsymbol{z}\boldsymbol{z}^\top\right)\boldsymbol{x} + (\boldsymbol{c}\circ\boldsymbol{z})^\top\boldsymbol{x} + \boldsymbol{\lambda}^\top\boldsymbol{z}\right\}\right\}, \tag{2}$$

where $\circ$ denotes the entry-wise product. Upon defining a convex quadratic function $p_{\boldsymbol{z}}(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\top\left(\boldsymbol{Q}\circ\boldsymbol{z}\boldsymbol{z}^\top\right)\boldsymbol{x} + (\boldsymbol{c}\circ\boldsymbol{z})^\top\boldsymbol{x} + \boldsymbol{\lambda}^\top\boldsymbol{z}$ for every fixed $\boldsymbol{z} \in \{0,1\}^n$, the above problem reduces to the following two-stage optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}), \quad \text{where} \quad f(\boldsymbol{x}) = \min_{\boldsymbol{z}\in\{0,1\}^n} p_{\boldsymbol{z}}(\boldsymbol{x}). \tag{3}$$

The above two-stage formulation induces a corresponding two-stage solution strategy for the original problem (1): first, characterize the *parametric cost* $f : \mathbb{R}^n \to \mathbb{R}$ by projecting out the binary variables $\boldsymbol{z}$, and then optimize f directly with respect to $\boldsymbol{x}$. Evidently, the first stage constitutes the computational bottleneck: efficient solution of the problem hinges on effectively characterizing f, which, as implied by the above reformulation, is a piecewise function composed of up to 2^n convex quadratic pieces. In isolation, this approach offers no apparent advantage over exhaustively enumerating all $\boldsymbol{z}$ configurations. However, we show that when the Hessian matrix $\boldsymbol{Q}$ exhibits a specific graph structure, this enumeration can be dramatically accelerated by systematically *pruning* choices of $\boldsymbol{z}$ that are provably suboptimal for all possible values of $\boldsymbol{x}$.

Our pruning strategy builds upon two key ideas. First, since $\boldsymbol{Q}$ is positive definite, any optimal solution $\boldsymbol{x}^\star$ must have a bounded norm; that is, $\|\boldsymbol{x}^\star\|_\infty \le U$ for some constant $U > 0$. Consequently, the parametric cost f needs to be characterized only within the bounded region $\mathcal{D} = \{\boldsymbol{x} : \|\boldsymbol{x}\|_\infty \le U\}$ containing the optimal solution. Second, we show that, under certain structural conditions, only a polynomial number of quadratic pieces $p_{\boldsymbol{z}}$ with $\boldsymbol{z} \in \{0,1\}^n$ are required to represent f within this

region. The key insight underlying this result is that, for any two sparsity patterns $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \{0,1\}^n$ with significant overlap (formally characterized by the notion of *m-similarity*; see Definition 4), the roots of the polynomial function $p_{\boldsymbol{z}_1} - p_{\boldsymbol{z}_2}$ grow exponentially with the length of the overlap. Hence, for a sufficiently long overlap, these roots lie outside $\mathcal{D}$, implying that either $p_{\boldsymbol{z}_1}(\boldsymbol{x}) > p_{\boldsymbol{z}_2}(\boldsymbol{x})$ or $p_{\boldsymbol{z}_1}(\boldsymbol{x}) < p_{\boldsymbol{z}_2}(\boldsymbol{x})$ within this region. In such cases, one of these quadratic pieces can be safely discarded (pruned), along with its corresponding sparsity pattern, without affecting the characterization of the parametric cost f.

While the existence of an efficient representation of the parametric cost f is encouraging, it does not by itself guarantee an efficient procedure for constructing it. To this end, we develop an efficient algorithm, called the *parametric algorithm*, for characterizing f in settings where the matrix $\boldsymbol{Q}$ admits a tree decomposition of small width. Our proposed parametric algorithm constructs f efficiently by dynamic programming (DP) operating over the tree decomposition of $\boldsymbol{Q}$. The overall computational complexity of the proposed method depends on the width of the tree decomposition, the volume growth of the support graph, and a suitable notion of margin for the problem, each of which will be discussed in detail in subsequent sections.

In practice, our parametric algorithm outperforms off-the-shelf solvers by orders of magnitude, solving problems with up to 20,000 variables in seconds to minutes, well beyond the reach of existing off-the-shelf solvers. We further demonstrate the practical relevance of the framework through an application to time-series forecasting. Leveraging a new MIQP formulation, we develop an extension of exponential smoothing that jointly performs forecasting and outlier detection. Computational results show that this integration enhances predictive performance and robustness against outliers on real-world data.

1.2 Related work

For a general positive definite matrix $\boldsymbol{Q}$, Problem (1) is known to be NP-hard [24]. Consequently, existing methods typically rely either on sequential convex relaxations, often strengthened through cutting-plane techniques, or on exploiting specific structural properties that render special cases tractable.

Methods based on sequential convex relaxation. Classical approaches based on convex relaxation employ *Big-M* formulations to encode indicator variables [37]. These formulations have been widely adopted for mixed-integer quadratic and regression-type problems, often incorporated within branch-and-bound techniques [12, 11, 26]. While effective for small- to medium-scale instances, Big-M formulations generally yield weak relaxations and suffer from numerical instability, leading to poor scalability on large problems [44]. A major breakthrough occurred with the introduction of the *perspective reformulation* technique, which provides significantly tighter convex relaxations for separable mixed-integer quadratic programs. Originally proposed by Stubbs [64] and further developed in a series of works [4, 35, 40], perspective reformulations have since become a cornerstone of modern algorithms for sparse and structured optimization [11, 73, 38, 71, 72]. In practice, these techniques form the backbone of state-of-the-art commercial solvers such as Gurobi, which integrate them into MIQP solvers via specialized cutting planes, presolve reductions, and perspective-based convexification modules. However, despite their effectiveness, such relaxation-based methods must still be embedded within branch-and-bound or branch-and-cut frameworks, whose exponential worst-case complexity continues to render them computationally prohibitive for large-scale problems.

Methods for structured $\boldsymbol{Q}$. Given the problem's exponential worst-case complexity, another line of work has studied instances in which $\boldsymbol{Q}$ possesses structural properties that enable more efficient algorithms. Examples include cases where $\boldsymbol{Q}$ is diagonal [22], Stieltjes [7, 43, 54], rank-one [63, 42], admits a sparse factorization $\boldsymbol{Q} = \boldsymbol{Q}_0^\top \boldsymbol{Q}_0$ [27], or exhibits other special structures [52, 25]. Closely related to this line of work are studies that exploit banded or tree-structured sparsity patterns in $\boldsymbol{Q}$. In the special case where $\boldsymbol{Q}$ is tridiagonal, Liu et al. [53] proposed a DP algorithm based on a shortest-path formulation that recovers the exact solution in $\mathcal{O}(n^2)$ time and memory. Building on this idea, Gómez et al. [39] extended the approach to general banded matrices and developed a *fully polynomial-time approximation scheme* (FPTAS) for Problem (1).

As an extension of the DP approach introduced by Liu et al. [53], Bhathena et al. [14] established that Problem (1) can be solved exactly in $\mathcal{O}(n^2)$ time when the sparsity graph of $\boldsymbol{Q}$ is a path or a tree. This result already motivates the use of tree decompositions, which provide a principled framework for extending tractability beyond tree-structured graphs. However, the approach by Bhathena et al. [14] relies on a delicate characterization of the conjugates of univariate convex quadratic functions, a property that does not generalize easily beyond tree graphs.

Methods based on small treewidth. Graphs with small treewidth were first introduced by Halin [41] and subsequently popularized by Robertson and Seymour [61] [see 30, for a historical account]. The notion of bounded treewidth is fundamental in algorithmic graph theory because it characterizes classes of graphs on which DP can be applied efficiently [19]. Since the late 1980s, it has been well established that several classically intractable combinatorial problems, such as *Independent Set*, *Graph Coloring*, and *Hamiltonian Cycle*, admit polynomial-time solutions via DP when restricted to graphs with small treewidth [9, 18, 5].

In contrast, Problem (1) exhibits a mixed-integer structure, involving both binary and continuous decision variables. In this broader setting, tree decompositions have been investigated extensively in the context of polynomial and mixed-integer optimization [69, 16, 55, 48, 70], encompassing subclasses of problems closely related to ours. In particular, Bienstock and Chen [15] consider quadratic optimization problems in which sparsity across disjoint variable blocks is governed by binary indicator variables subject to coupling constraints. Their framework is applicable to Problem (1) when the Hessian matrix $\boldsymbol{Q}$ has bounded treewidth. Nonetheless, their approach yields only an FPTAS algorithm, and, to the best of our knowledge, no practical implementation of the proposed algorithm has been reported in the literature.

1.3 Outline and overview

The remainder of the paper is organized as follows. Section 2 introduces the necessary preliminaries and background. Section 3 presents an application to time-series forecasting based on exponential smoothing with outlier detection. Section 4 presents an (inefficient) DP approach for solving Problem (1) by sequentially characterizing its parametric cost in exponential time. Although this DP formulation is not practical, it serves as the foundation for our efficient pruning strategy, developed in Section 5. Section 6 provides theoretical guarantees on the correctness and runtime of the proposed algorithm. Section 7 reports numerical results on synthetic instances, as well as real-world time-series data from the NAB dataset. Finally, Section 8 concludes with a summary of the main contributions.

2 Preliminaries and background

Matrices and vectors are denoted by bold uppercase and lowercase symbols (e.g., $\boldsymbol{Q}$ and $\boldsymbol{c}$), respectively, while scalars are denoted by unbolded symbols (e.g., n). Given a matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and index sets $\mathcal{I}, \mathcal{J} \subseteq \{1, \ldots, n\}$, we denote by $\boldsymbol{Q}_{\mathcal{I},\mathcal{J}}$ the submatrix of $\boldsymbol{Q}$ consisting of rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$. Similarly, for a vector $\boldsymbol{c} \in \mathbb{R}^n$, we write $\boldsymbol{c}_{\mathcal{J}}$ for the subvector of $\boldsymbol{c}$ restricted to the indices in $\mathcal{J}$. We use $\mathbb{I}(x)$ to denote the indicator function on $\mathbb{R}$, which equals 0 if $x = 0$ and 1 otherwise. For a symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, let $\mu_{\min}(\boldsymbol{Q})$ and $\mu_{\max}(\boldsymbol{Q})$ denote its smallest and largest eigenvalues, respectively. The spectral condition number of $\boldsymbol{Q}$ is defined as $\kappa_2(\boldsymbol{Q}) := \mu_{\max}(\boldsymbol{Q})/\mu_{\min}(\boldsymbol{Q})$. Similarly, the condition number of $\boldsymbol{Q}$ in the induced ∞-norm is defined as $\kappa_\infty(\boldsymbol{Q}) = \|\boldsymbol{Q}^{-1}\|_\infty \|\boldsymbol{Q}\|_\infty$. When the argument is omitted, i.e., when we write $\mu_{\min}$, $\mu_{\max}$, κ_2, or κ_∞, these quantities refer to the eigenvalues and condition numbers of the Hessian matrix $\boldsymbol{Q}$. The entrywise $\ell_{1,1}$-norm of $\boldsymbol{S}$ is defined as $\|\boldsymbol{S}\|_{1,1} := \sum_{i=1}^n \sum_{j=1}^n |\boldsymbol{S}_{ij}|$. We say that $\boldsymbol{S}$ is a *banded matrix* with bandwidth $w \in \mathbb{Z}_+$ if $\boldsymbol{S}_{ij} = 0$ for all $|i - j| > w$. We denote by $f^\star$ the optimal objective value of Problem (1), and by $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ its corresponding optimal solution.

Given a symmetric matrix $\boldsymbol{Q} \in R^{n\times n}$, its *support graph*, denoted by $\text{supp}(\boldsymbol{Q})$, is a simple unweighted graph $\mathsf{G} = (\mathcal{V}_\mathsf{G}, \mathcal{E}_\mathsf{G})$ with vertex set $\mathcal{V}_\mathsf{G} = \{1, 2, \ldots, n\}$, where an edge $(i, j) \in \mathcal{E}_\mathsf{G}$ exists if and only if $\boldsymbol{Q}_{ij} \neq 0$ for $i \neq j$. For any two nodes $u, v \in \mathcal{V}_\mathsf{G}$, let $\text{dist}(u, v)$ denote the length of the shortest path between u and v in $\text{supp}(\boldsymbol{Q})$. More generally, for subset of nodes $\mathcal{I}, \mathcal{J} \subseteq \mathcal{V}_\mathsf{G}$, we define $\text{dist}(\mathcal{I}, \mathcal{J}) = \min_{i\in\mathcal{I}, j\in\mathcal{J}} \{\text{dist}(i, j)\}$.

2.1 Tree decomposition and treewidth

Definition 1 (Tree decomposition)**.** The tree decomposition of a graph $\mathsf{G} = (\mathcal{V}_\mathsf{G}, \mathcal{E}_\mathsf{G})$ is a pair $(\mathsf{T}, \mathcal{B})$, where $\mathsf{T} = (\mathcal{V}_\mathsf{T}, \mathcal{E}_\mathsf{T})$ is a tree and $\mathcal{B} = \{\mathcal{B}_u : u \in \mathcal{V}_\mathsf{T}\}$ is a family of subsets (bags) of $\mathcal{V}_\mathsf{G}$, satisfying the following conditions:

1. $\mathcal{V}_\mathsf{G} = \bigcup_{u\in\mathcal{V}_\mathsf{T}} \mathcal{B}_u$. That is, every node of G appears in at least one bag.
2. For every edge $(i, j) \in \mathcal{E}_\mathsf{G}$, there exists a bag $\mathcal{B}_u$ containing both i and j.
3. For every node $i \in \mathcal{V}_\mathsf{G}$, the set $\{u \in \mathcal{V}_\mathsf{T} : i \in \mathcal{B}_u\}$ induces a connected subtree of T. That is, bags containing i form a connected subtree of T.

The width of a tree decomposition is given by $\max_{u\in\mathcal{V}_\mathsf{T}}\{|\mathcal{B}_u| - 1\}$. A graph may admit many different tree decompositions. A trivial decomposition places all vertices of $\mathcal{V}_\mathsf{G}$ into a single bag, yielding width $|\mathcal{V}_\mathsf{G}| - 1$. A more informative structural measure is the *treewidth*, denoted by ω, defined as the minimum width among all tree decompositions of G. Intuitively, treewidth measures how close a graph is to being a tree. For example, trees have treewidth 1. Series-parallel graphs have treewidth 2. On the other hand, fully dense graphs—being far from tree-like—have a maximum treewidth $|\mathcal{V}_\mathsf{G}| - 1$.

Next, we introduce the notion of a *balanced tree decomposition*.

Definition 2 (balanced tree decomposition)**.** Let $(\mathsf{T}, \mathcal{B})$ be a tree decomposition with width ω. We call $(\mathsf{T}, \mathcal{B})$ a balanced tree decomposition if:

1. Each bag of T contains exactly $\omega + 1$ nodes.

2. For every edge $(u, v) \in \mathcal{E}_{\mathsf{T}}$, the corresponding bags satisfy $|\mathcal{B}_u \cap \mathcal{B}_v| = \omega$.

Finding an optimal tree decomposition with the smallest possible width—i.e., one that matches the treewidth ω—is NP-hard [6]. Nevertheless, for graphs with bounded treewidth, both recognition and construction of optimal tree decompositions can be performed in linear time [20]. Beyond these theoretical results, practical heuristics such as *fill-reducing* and *nested dissection* algorithms often yield decompositions with near-optimal widths in practice [45]. Moreover, any tree decomposition with width ω can be transformed into a balanced one of the same width by adding nodes to existing bags and, if necessary, inserting or removing bags. An illustrative example is shown in Figure 1. Balanced tree decompositions always exist, and, given a tree decomposition of width ω, can be obtained in $\mathcal{O}(n)$ via the so-called "nice" tree decompositions; see [21, Section 2] for definitions and the construction.

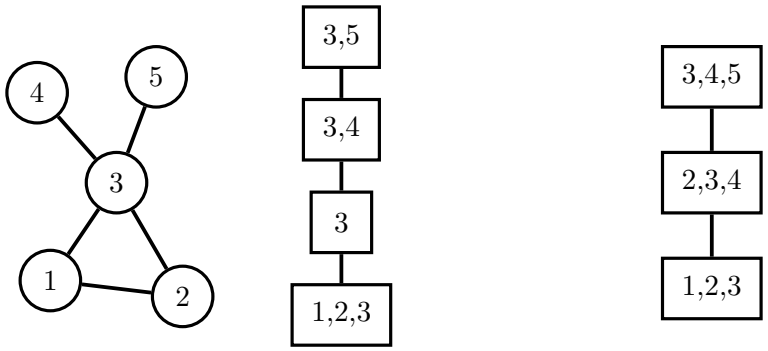

Figure 1: A graph with a tree decomposition that violates Definition 2 (left), and its corresponding balanced tree decomposition (right).

Throughout this work, we assume that a balanced tree decomposition of $\mathrm{supp}(\boldsymbol{Q})$ with width ω is available. For example, for our proposed ESOC formulation, $\mathrm{supp}(\boldsymbol{Q})$ has treewidth 2, and the corresponding balanced tree decomposition can be readily obtained (see Figure 4). We note that our proposed method does not require a tree decomposition of minimum width; any decomposition with reasonably small width suffices.

We next describe a procedure for labeling the nodes of $\mathrm{supp}(\boldsymbol{Q})$ induced by its balanced tree decomposition T. As will become evident later, this labeling plays a crucial role in establishing the efficiency of the proposed algorithm. We begin by labeling the bags in T. By the second property of the balanced tree decomposition, $(\mathsf{T}, \mathcal{B})$ contains $n - \omega$ bags. For simplicity, we assume that the edges in T are oriented naturally toward a designated root. Under this convention, each bag may have multiple parents but at most one child. We denote by $\mathrm{child}_{\mathsf{T}}(u)$ the child of bag u in T, and by $\mathrm{par}_{\mathsf{T}}(u)$ the set of its parents. Labels are assigned to the bags according to a topological ordering: for every bag u, we require $u < \mathrm{child}_{\mathsf{T}}(u)$. Since T contains $n - \omega$ bags, the root bag receives label $n - \omega$. As T is acyclic, such a topological labeling always exists and can be computed in $\mathcal{O}(n)$ time and memory [3, Algorithm 3.8].

We now proceed to label the nodes of $\mathrm{supp}(\boldsymbol{Q})$ based on its labeled balanced tree decomposition. For any non-root bag u, by the definition of a balanced tree decomposition, the set $\mathcal{B}_u \setminus \mathcal{B}_{\mathrm{child}_{\mathsf{T}}(u)}$ contains a unique node of $\mathrm{supp}(\boldsymbol{Q})$, which we label by u. The nodes in the root bag of T are labeled arbitrarily with the remaining labels $\{n - \omega + 1, \ldots, n\}$. An illustration of this labeling scheme is provided in Figure 2. Since both $\mathcal{B}_u$ and $\mathcal{B}_{\mathrm{child}_{\mathsf{T}}(u)}$ have size $\omega + 1$, computing each set difference

requires $\mathcal{O}(\omega^2)$ time. Repeating this operation for all bags yields a total labeling cost of $\mathcal{O}(n\omega^2)$ time. The memory required is $\mathcal{O}(n\omega)$, which corresponds to storing the labels for each bag.

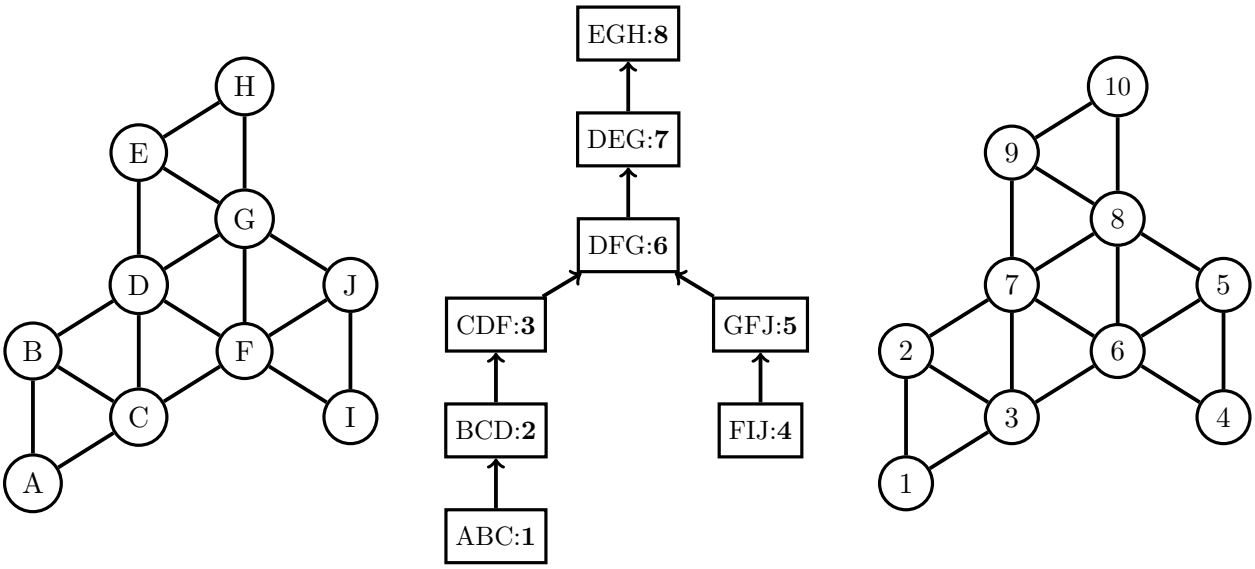

Figure 2: The figure on the left illustrates a graph G with nodes labeled arbitrarily. The center panel shows a balanced tree decomposition of G, where the bags are labeled according to the topological ordering (labels shown in bold). The panel on the right shows the resulting labeling of the nodes in G obtained by the described procedure.

For any node $u \in \mathrm{supp}(\boldsymbol{Q})$ with $u \leq n - \omega$, we define $\mathrm{supp}_u(\boldsymbol{Q})$ as the subgraph of $\mathrm{supp}(\boldsymbol{Q})$ induced by the nodes contained in the largest subtree of T comprising the bag $\mathcal{B}_u$ and all of its ancestors. For $u > n - \omega$, we set $\mathrm{supp}_u(\boldsymbol{Q}) := \mathrm{supp}(\boldsymbol{Q})$. For any u, the treewidth of $\mathrm{supp}_u(\boldsymbol{Q})$ does not exceed ω [30, Lemma 12.4.1]. As an example, for the graph shown in Figure 2, $\mathrm{supp}_5(\boldsymbol{Q})$ is the subgraph induced by the nodes in $\mathcal{B}_5 \cup \mathcal{B}_4 = \{5, 6, 8\} \cup \{4, 5, 6\}$, while $\mathrm{supp}_6(\boldsymbol{Q})$ is induced by the nodes in $\bigcup_{i=1}^{6} \mathcal{B}_i$. For a node $u \in \mathrm{supp}(\boldsymbol{Q})$, we denote by $\boldsymbol{Q}_{[u]}$ the principal submatrix of $\boldsymbol{Q}$ indexed by the nodes of $\mathrm{supp}_u(\boldsymbol{Q})$; clearly, $\mathrm{supp}(\boldsymbol{Q}_{[u]}) = \mathrm{supp}_u(\boldsymbol{Q})$. Similarly, $\boldsymbol{c}_{[u]}$ and $\boldsymbol{\lambda}_{[u]}$ denote the subvectors of $\boldsymbol{c}$ and $\boldsymbol{\lambda}$ restricted to these indices. We use $\mathcal{J}_u$ to denote the set of nodes in $\mathrm{supp}_u(\boldsymbol{Q})$ excluding those in $\mathcal{B}_u$, and let $n_u := |\mathcal{J}_u|$. Recall that the tree decomposition T contains $n - \omega$ bags, with $\mathcal{B}_{n-\omega}$ designated as the root. With a slight abuse of notation, we define auxiliary sets $\mathcal{B}_u := \{u, u + 1, \ldots, n\}$ for $u \in \{n - \omega + 1, \ldots, n\}$. Although these sets are not part of the original tree decomposition, we refer to them as *bags* for convenience. For $u > n - \omega$, we also define $\mathrm{par}_{\mathsf{T}}(u) := \{u - 1\}$. Finally, for any u, we define $\tau_u := \min\{\omega, n - u\}$, ensuring that $|\mathcal{B}_u| = \tau_u + 1$. When u is clear from context, we simply write τ.

2.2 The local parametric cost

Recall that $\boldsymbol{Q}_{[u]} \in \mathbb{R}^{n_u+\tau+1 \times n_u+\tau+1}$ is a principal submatrix of $\boldsymbol{Q}$ indexed by $\mathcal{J}_u \cup \mathcal{B}_u$. Let $\pi_u : \mathcal{J}_u \cup \mathcal{B}_u \to \{1, \ldots n_u + \tau + 1\}$ be the canonical indexing map that assigns to each $i \in \mathcal{J}_u \cup \mathcal{B}_u$ its corresponding row/column position within the submatrix $\boldsymbol{Q}_{[u]}$. For any $u \in \{1, \ldots, n\}$ and

$\tau = \min\{\omega, n-u\}$, the local parametric cost, $f_u : \mathbb{R}^{\tau+1} \to \mathbb{R}$, is defined as

$$f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) := \min_{\boldsymbol{x}\in\mathbb{R}^{n_u+\tau+1}, \boldsymbol{z}\in\{0,1\}^{n_u}} \quad \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}_{[u]}\boldsymbol{x} + \boldsymbol{c}_{[u]}^\top \boldsymbol{x} + \boldsymbol{\lambda}_{\mathcal{J}_u}^\top \boldsymbol{z} \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad i \in \pi_u(\mathcal{J}_u) \tag{4b}$$

$$\boldsymbol{x}_{\pi_u(\mathcal{B}_u)} = \boldsymbol{\alpha}_{\mathcal{B}_u}. \tag{4c}$$

We note that the subscript in $\boldsymbol{\alpha}_{\mathcal{B}_u}$ is not mathematically necessary; however, as will be seen later, it helps streamline and clarify the subsequent arguments. Intuitively, f_u denotes the optimal value of the subproblem defined over $\text{supp}_u(\boldsymbol{Q})$ after fixing the local continuous variables associated with $\mathcal{B}_u$ to $\boldsymbol{\alpha}_{\mathcal{B}_u} \in \mathbb{R}^{\tau+1}$. As will be explained, this local parametric cost establishes a structured connection between subproblems and enables the DP algorithm to efficiently propagate information through the tree decomposition.

Our next lemma establishes that the local parametric cost can be expressed as the pointwise minimum of at most 2^{n_u} strongly convex quadratic functions. The proof of this lemma is provided in Appendix A.1.

Lemma 1. *Fix any $u \in \{1, 2, \ldots, n\}$. The local parametric cost $f_u : \mathbb{R}^{\tau+1} \to \mathbb{R}$ can be written as*

$$f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \min_{\boldsymbol{s}\in\{0,1\}^{n_u}} \{p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})\} \tag{5}$$

where, for every $\boldsymbol{s} \in \{0,1\}^{n_u}$, $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha})$ is a strongly convex quadratic function. In particular, let $\mathcal{J}_{u,\boldsymbol{s}} = \{i \in \mathcal{J}_u \mid \boldsymbol{s}_i = 1\}$. Then $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})$ is given by

$$p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}\boldsymbol{A}_{u,\boldsymbol{s}}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{b}_{u,\boldsymbol{s}}^\top \boldsymbol{\alpha}_{\mathcal{B}_u} + d_{u,\boldsymbol{s}},$$

$$\text{where} \quad \begin{cases} \boldsymbol{A}_{u,\boldsymbol{s}} & = \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u} - \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}} \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^\top \\ \boldsymbol{b}_{u,\boldsymbol{s}} & = \boldsymbol{c}_{\mathcal{B}_u} - \boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^T \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^\top \\ d_{u,\boldsymbol{s}} & = -\frac{1}{2}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^\top (\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}})^{-1}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}} + \sum_{i\in\mathcal{J}_{u,\boldsymbol{s}}} \boldsymbol{\lambda}_i. \end{cases}$$

Since f_u can be expressed as the minimum of a collection of quadratic functions, it can, in principle, be stored in memory by saving the coefficients $(\boldsymbol{A}_{u,\boldsymbol{s}}, \boldsymbol{b}_{u,\boldsymbol{s}}, d_{u,\boldsymbol{s}})$ corresponding to each quadratic piece $p_{u,\boldsymbol{s}}$. Storing the coefficients of each $p_{u,\boldsymbol{s}}$ requires $\mathcal{O}(\omega^2)$ memory. However, because the number of sparsity patterns $\boldsymbol{s}$ grows exponentially with the number of nodes n_u in $\text{supp}_u(\boldsymbol{Q})$, the total memory required to store f_u also scales exponentially. Evidently, such direct storage is impractical.

When $\text{supp}(\boldsymbol{Q})$ is a tree (i.e., $\omega = 1$), Bhathena et al. [14] demonstrated that the parametric cost f_u can be represented as the minimum of only $\mathcal{O}(n_u)$ quadratic pieces, resulting in a linear memory requirement. For the general case $\omega > 1$, however, the existence of a compact representation of f_u remains an open question. In this work, we address this challenge by leveraging additional structural properties of the problem, such as its volume growth, which we describe next.

2.3 Volume growth of a graph

For any node u in $\text{supp}(\boldsymbol{Q})$, define $\mathcal{V}_{u,m} := \{\, i \in \mathcal{J}_u \mid \text{dist}(i, \mathcal{B}_u) \le m \,\}$ as the set of nodes in $\mathcal{J}_u$ lying within distance m of bag $\mathcal{B}_u$. This set is referred to as the *m-neighborhood* of $\mathcal{B}_u$ within the induced subgraph $\text{supp}_u(\boldsymbol{Q})$. Let $\Delta_m := \max_{u\in\{1,\ldots,n\}} |\mathcal{V}_{u,m}|$. The *volume growth function* of $\text{supp}(\boldsymbol{Q})$ characterizes how rapidly Δ_m increases with m.

Assumption 1 (Polynomial volume growth)**.** There exist constants $\gamma, \delta > 0$ such that

$$\Delta_m \leq \delta\, m^\gamma \qquad \text{for all } m \geq 1.$$

Assumption 1 is closely related to the notion of uniform polynomial volume growth [31, 49], although the precise definitions may vary slightly. This assumption holds for many well-studied graph families. Examples include trees with $\mathcal{O}(1)$ leaves, cycles, grid graphs, and support graphs of banded matrices. In contrast, trees with $\mathcal{O}(n)$ leaves may not satisfy this assumption. Notable examples are complete binary trees, for which $\Delta_m = \mathcal{O}(2^m)$, and star graphs, where $\Delta_1 = n - 2$.

3 Application: Exponential smoothing with outlier correction

To motivate our methodological framework for convex quadratic optimization over structured graphs, we now introduce a practical application in time-series analysis. Specifically, we adopt a unified perspective that integrates forecasting and outlier detection. Building on classical exponential smoothing, a simple and flexible method for time-series forecasting, we extend it with an MIQP-based formulation to handle outliers effectively. The next subsection reviews exponential smoothing and discusses its limitations, motivating our proposed extension.

3.1 Exponential smoothing

Exponential smoothing encompasses a family of methods widely used in operations research for time series with varying characteristics such as level, trend, and seasonality. By assigning exponentially decreasing weights to past observations, these methods smooth short-term fluctuations while preserving long-term structure, making them valuable for forecasting and decision-making in dynamic environments. Applications span retail and inventory management [59, 17, 36], market analysis and policy design [47, 66], and quality control [65]; see [46] for additional examples.

Among the various exponential smoothing methods, *Simple Exponential Smoothing* (SES)—also known as single exponential smoothing—provides a foundational formulation. Formally, let $\{\boldsymbol{y}_t\}_{t=1}^T$ denote a univariate time series signal. The smoothed sequence $\{\boldsymbol{x}_t\}_{t=1}^T$ is defined recursively as:

$$\boldsymbol{x}_t = \beta \boldsymbol{y}_t + (1-\beta)\boldsymbol{x}_{t-1} \quad \text{for } t = 2, \ldots, T; \tag{SES}$$

with initialization $\boldsymbol{x}_1 = \boldsymbol{y}_1$. The smoothing parameter $\beta \in (0, 1)$ controls the weight assigned to the most recent observation. Larger values of β make the method more responsive to changes but more sensitive to noise, while smaller values produce smoother outputs at the cost of responsiveness.

To illustrate the limitations of SES, we consider a time series of network traffic volume to a cloud server, measured in bytes at five-minute intervals. This dataset was collected by Amazon CloudWatch and is publicly available via the Numenta Anomaly Benchmark (NAB) [1]. Figure 3 (top row) illustrates SES applied to this time series with smoothing factors $\beta = 0.5, 0.2, 0.05$. The circled points indicate outliers labeled by NAB. With larger values of β, the smoothed series more closely follows the raw data, but fails to exclude the outliers in the process. Conversely, small values of β dampen the influence of the outliers while introducing lag relative to the underlying series. This behavior highlights a fundamental limitation of SES: it cannot simultaneously suppress outliers and adapt quickly to changes in the underlying signal. Motivated by this inherent trade-off in SES, we formulate a robust extension, which we call *exponential smoothing with outlier correction* (ESOC). We focus primarily on SES, but note that our formulation extends naturally to other variants, including double- and triple-exponential smoothing.

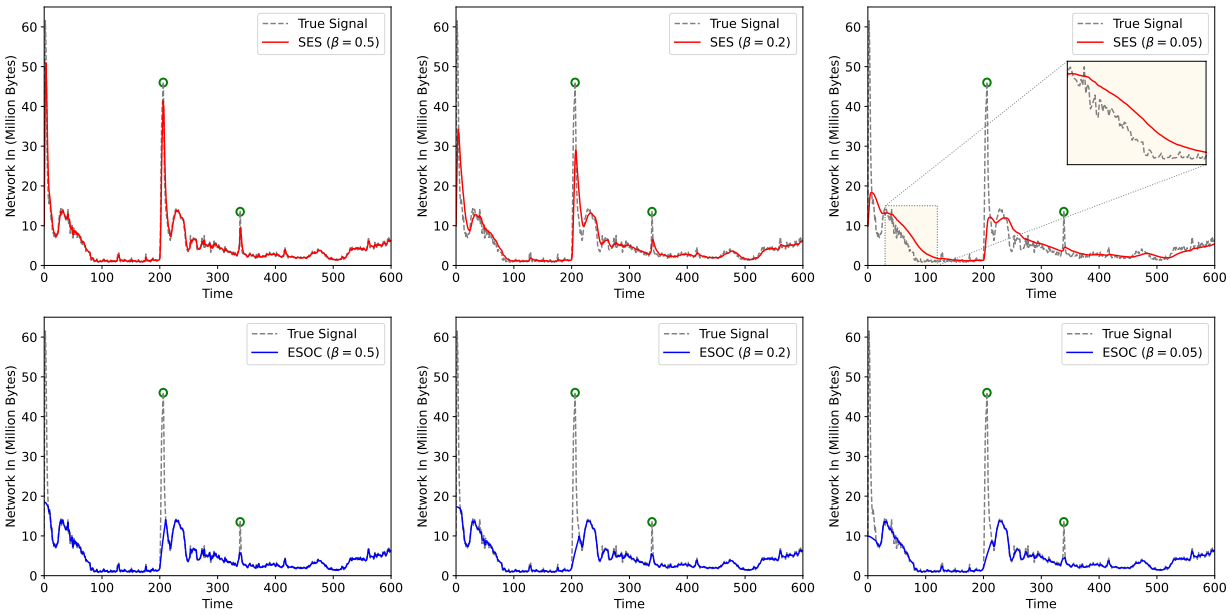

Figure 3: **Top row:** SES applied to a signal containing anomalies from the Numenta Anomaly Benchmark (NAB), with smoothing parameters $\beta = 0.5$, 0.2, and 0.05 (from left to right). Outliers identified by the NAB ground-truth labels are highlighted with circles. Lower values of β reduce sensitivity to outliers but induce a noticeable lag relative to the underlying signal. **Bottom row:** ESOC effectively detects and removes outliers across all values of β, while preserving the alignment with the true signal and avoiding lag.

3.2 Exponential smoothing with outlier correction

We now introduce an MIQP extension of SES that explicitly accounts for outliers. In this framework, the vector $\boldsymbol{x} \in \mathbb{R}^T$ denotes the smoothed signal, and $\boldsymbol{o} \in \mathbb{R}^T$ is a sparse vector capturing outliers. The associated optimization problem is

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{o}\in\mathbb{R}^T,\boldsymbol{z}\in\{0,1\}^T} \quad & \sum_{t=1}^{T} (\boldsymbol{y}_t - \boldsymbol{x}_t - \boldsymbol{o}_t)^2 + \sum_{t=1}^{T} \boldsymbol{\lambda}_t \boldsymbol{z}_t \\ \text{s.t.} \quad & \boldsymbol{x}_t = \beta(\boldsymbol{y}_t - \boldsymbol{o}_t) + (1-\beta)\boldsymbol{x}_{t-1} && \text{for } t = 2, \ldots, T \\ & \boldsymbol{o}_t(1 - \boldsymbol{z}_t) = 0 && \text{for } t = 1, \ldots, T, \end{aligned}$$

where $\boldsymbol{z} \in \{0,1\}^T$ captures the sparsity pattern of $\boldsymbol{o} \in \mathbb{R}^T$, and is controlled by a nonnegative regularization vector $\boldsymbol{\lambda} \in \mathbb{R}^T$. When $\boldsymbol{z}_t = 0$, the sample $\boldsymbol{y}_t$ is treated as noise-free, yielding the standard SES recursion. When $\boldsymbol{z}_t = 1$, the variable $\boldsymbol{o}_t$ is allowed to take nonzero values, thereby allowing for correction of the noisy observation $\boldsymbol{y}_t$.

Rather than enforcing the exponential smoothing dynamic as a hard constraint, we adopt a relaxed formulation that penalizes deviations from this constraint in the objective function, as

follows:

$$\min_{\boldsymbol{x},\boldsymbol{o}\in\mathbb{R}^T} \sum_{t=1}^{T} (\boldsymbol{y}_t - \boldsymbol{x}_t - \boldsymbol{o}_t)^2 + \sum_{t=1}^{T} \boldsymbol{\lambda}_t \boldsymbol{z}_t + \mu_1 \sum_{t=2}^{T} (\beta(\boldsymbol{y}_t - \boldsymbol{o}_t) + (1-\beta)\boldsymbol{x}_{t-1} - \boldsymbol{x}_t)^2 + \mu_2 \|\boldsymbol{o}\|_2^2 \quad \text{(ESOC)}$$
$$\text{s.t.} \quad \boldsymbol{o}_t(1 - \boldsymbol{z}_t) = 0 \qquad \text{for } t = 1, \ldots, T,$$

where $\mu_1 \geq 0$ penalizes deviations from the exponential smoothing dynamics. The additional term $\mu_2\|\boldsymbol{o}\|_2^2$, with $\mu_2 \geq 0$, ensures that the Hessian of the objective is positive definite. We note that although this regularization may induce slight shrinkage in $\boldsymbol{o}$, μ_2 is set to a very small value in practice, rendering the effect negligible while providing substantial numerical stability benefits. The resulting relaxed formulation conforms to the structure of Problem (1), with a Hessian matrix whose sparsity pattern has treewidth equal to 2 and exhibits linear volume growth with $\Delta_m \leq 3m$ (see Figure 4). Consequently, (ESOC) falls within the class of problems that can be solved efficiently using the parametric algorithm developed in this paper.

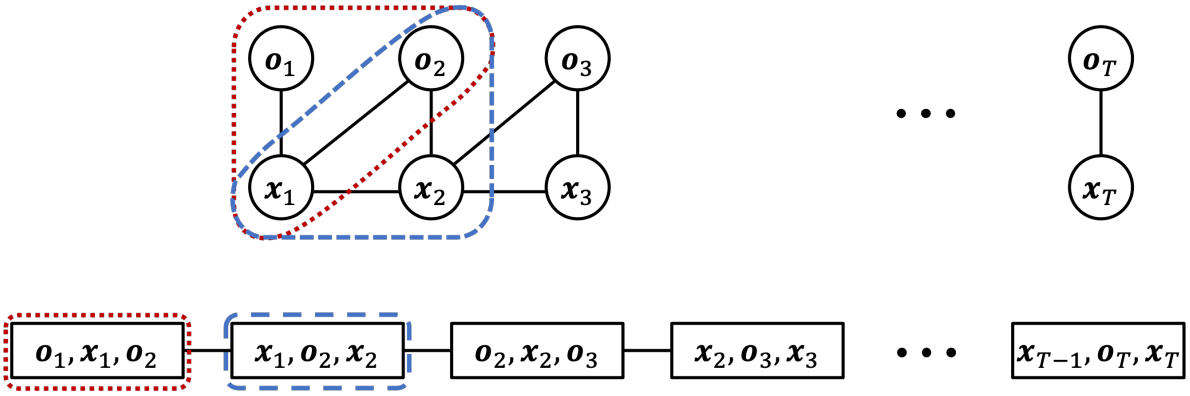

Figure 4: Hessian support graph (top) and corresponding tree decomposition (bottom) for ESOC model.

We conclude this section with preliminary empirical results; a detailed experiment study is provided in Section 7.3. Figure 3 (second row) illustrates the solution of ESOC obtained via our parametric algorithm across different smoothing parameters β. As discussed earlier, SES exhibits a trade-off between outlier sensitivity and responsiveness. In contrast, ESOC effectively suppresses outliers across all β values without sacrificing responsiveness. To quantify this comparison, we report the corresponding forecast mean squared error (MSE) values for both methods. For $\beta \in \{0.5, 0.2, 0.05\}$, SES yields MSE values of $16.89, 9.87$, and 25.70; the corresponding values under ESOC are $0.32, 0.46$, and 0.22. In each case, ESOC attains lower MSE, corresponding to improved predictive accuracy.

We next evaluate the computational scalability of ESOC. Specifically, we apply our model to five real-world time series from the NAB dataset, including AWS CloudWatch metrics and traffic speed data, using $\beta \in \{0.05, 0.2, 0.5\}$. Each resulting optimization problem is solved using both Gurobi, a state-of-the-art commercial MIQP solver, and our proposed algorithm. To assess scalability, we vary the problem size T by truncating each time series and record the corresponding solution times. For each T, we solve 15 instances (five signals and three β values each). Figure 5 reports these runtime comparisons: our algorithm solves all instances to optimality and achieves an average runtime of

54 seconds at $T = 1000$, whereas GUROBI's runtime increases drastically with T and exceeds the one-hour limit beyond $T = 500$.

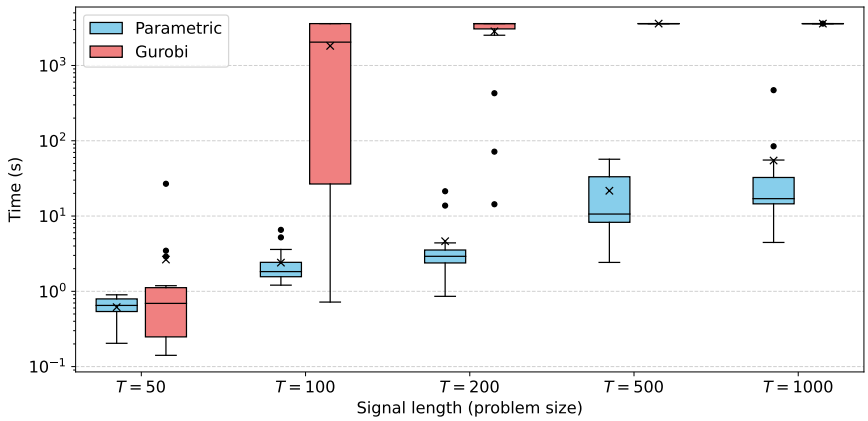

Figure 5: Runtime comparison between the proposed algorithm and GUROBI on five real-world time series from the NAB dataset, with smoothing parameters. Signal lengths are varied by truncation. For each signal, smoothing parameters are set to $\beta \in \{0.05, 0.2, 0.5\}$. "$\times$" denotes the mean runtime. GUROBI runs are terminated after one hour, resulting in flat values at larger signal lengths.

4 Dynamic programming via local parametric costs

Recall the definition of the local parametric cost f_u in (4). A key observation is that once the local variables $\boldsymbol{x}_{\pi_u(\mathcal{B}_u)}$ are fixed to $\boldsymbol{\alpha}_{\mathcal{B}_u}$, the subproblem defined over $\mathrm{supp}_u(\boldsymbol{Q})$ decomposes into independent components, each associated with the variables in the subtree of T rooted at a parent of u. This decomposition follows directly from the *running intersection property* of the tree decomposition, which states that if a node v appears in two bags $\mathcal{B}_i$ and $\mathcal{B}_j$, then it must also appear in every bag along the path between $\mathcal{B}_i$ and $\mathcal{B}_j$ in T. As a result, the nodes in the subgraph $\mathrm{supp}_u(\boldsymbol{Q})$ become connected to the rest of the graph only through the nodes in the bag $\mathcal{B}_u$, and fixing $\boldsymbol{x}_{\pi_u(\mathcal{B}_u)}$ removes all remaining couplings. The following lemma formalizes this observation and provides an explicit recursive representation of the local parametric cost f_u in terms of the corresponding quantities associated with the parent nodes of u in the tree decomposition. The proof of this lemma is provided in Appendix A.2.

Lemma 2. *For any node u, the local parametric cost $f_u : \mathbb{R}^{\tau+1} \to \mathbb{R}$ satisfies*

$$f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = h_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) + \sum_{v \in \mathrm{par}_{\mathsf{T}}(u)} \left(g_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) - \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) \right), \tag{6}$$

where the functions $h_u : \mathbb{R}^{\tau+1} \to \mathbb{R}$ and $g_v, \phi_v : \mathbb{R}^{\tau} \to \mathbb{R}$ are defined as follows

$$h_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) := \frac{1}{2} \boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u, \mathcal{B}_u} \boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{c}_{\mathcal{B}_u}^\top \boldsymbol{\alpha}_{\mathcal{B}_u}, \tag{7a}$$

$$g_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) := \min_{\boldsymbol{x}_v \in \mathbb{R}} \left\{ f_v(\boldsymbol{x}_v, \boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) + \boldsymbol{\lambda}_v \mathbb{I}(\boldsymbol{x}_v) \right\}, \tag{7b}$$

$$\phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) := \frac{1}{2} \boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}^\top \boldsymbol{Q}_{\mathcal{B}_v \setminus v, \mathcal{B}_v \setminus v} \boldsymbol{\alpha}_{\mathcal{B}_v \setminus v} + \boldsymbol{c}_{\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}}^\top \boldsymbol{\alpha}_{\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}}. \tag{7c}$$

The above equation can be interpreted as the *u-stage DP update* [10, Chapter 2], which expresses the local parametric cost (or the so-called *cost-to-go*) of bag $\mathcal{B}_u$ in terms of the local parametric costs of its parent bags $\{f_v : v \in \mathrm{par}_{\mathsf{T}}(u)\}$. The intuition behind (6) is natural: to characterize the local parametric cost f_u, three adjustments are required based on the local costs at its parent bags. First, the variables in $\mathcal{B}_v$ that do not appear in $\mathcal{B}_u$ must be eliminated, which is achieved through the minimization over $\boldsymbol{\alpha}_v$ in the definition of the function g_v. Second, the costs associated with variables in $\mathcal{B}_u$ must be added; this contribution is captured by the function h_u. Finally, the cost associated with the remaining variables in $\mathcal{B}_v \backslash v$ is removed through ϕ_v to prevent double counting across parent bags.

This naturally leads to a DP algorithm that traverses the tree decomposition from leaves to the root and, at each bag $\mathcal{B}_u$, recursively computes the parametric cost f_u and the function g_u. Since f_u is defined as the minimum of convex quadratic functions, it follows that g_u is also representable as a minimum of quadratic functions, though not necessarily convex. Moreover, if f_u consists of N quadratic pieces, then g_u contains at most $2N$ quadratic pieces. This doubling arises from the indicator function $\mathbb{I}(\boldsymbol{x}_v)$ in the definition of g_u, which splits each quadratic piece of f_u into two distinct quadratic functions depending on the value of $\mathbb{I}(\boldsymbol{x}_v)$. Moreover, the labeling scheme guarantees that each g_u is computed before any parametric cost that depends on it. Upon computing f_n, the optimal cost can be obtained as

$$f^\star = \min_{\boldsymbol{\alpha}_n \in \mathbb{R}} \{f_n(\boldsymbol{\alpha}_n) + \boldsymbol{\lambda}_n \mathbb{I}(\boldsymbol{\alpha}_n)\}.$$

With the optimal cost at hand, the remaining local parameters can be recovered by traversing from the root bag back to the leaf bags. This DP algorithm is stated in Algorithm 1. The following proposition establishes the correctness and computational complexity of this DP algorithm. Note that the algorithm is exponential in the number of binary decision variables n, thus may be impractical as stated.

Algorithm 1 DP for Problem (1)

Input: Problem (1) and a balanced tree decomposition T of $\mathrm{supp}(\boldsymbol{Q})$ with width ω;
Output: The optimal cost $f^\star$ and an optimal solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ of Problem (1);

Label the nodes of $\mathrm{supp}(\boldsymbol{Q})$ according to the scheme described in Section 2.1;
Set $f_1(\boldsymbol{\alpha}_{\mathcal{B}_1}) = \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_1}^\top \boldsymbol{Q}_{\mathcal{B}_1,\mathcal{B}_1} \boldsymbol{\alpha}_{\mathcal{B}_1} + \boldsymbol{\alpha}_{\mathcal{B}_1}^\top \boldsymbol{c}_{\mathcal{B}_1}$.
for $u = 1, \ldots, n-1$ **do**
    Calculate g_u from f_u via Equation (7b);
    Calculate f_{u+1} via Equation (6);
end for
Obtain $f^\star = \min_{\boldsymbol{\alpha}_n \in \mathbb{R}} \{f_n(\boldsymbol{\alpha}_n) + \boldsymbol{\lambda}_n \mathbb{I}(\boldsymbol{\alpha}_n)\}$, $\boldsymbol{x}_n^\star = \operatorname{argmin}_{\boldsymbol{\alpha}_n \in \mathbb{R}} \{f_n(\boldsymbol{\alpha}_n) + \boldsymbol{\lambda}_n \mathbb{I}(\boldsymbol{\alpha}_n)\}$, and $\boldsymbol{z}_n^\star = \mathbb{I}(\boldsymbol{x}_n^\star)$;
for $u = n-1, \ldots, 1$ **do**
    Set $\boldsymbol{x}_u^\star = \operatorname{argmin}_{\boldsymbol{\alpha}_u \in \mathbb{R}} \left\{ f_u(\boldsymbol{\alpha}_u, \boldsymbol{x}^\star_{\mathcal{B}_u \backslash u}) + \boldsymbol{\lambda}_u \mathbb{I}(\boldsymbol{\alpha}_u) \right\}$ and $\boldsymbol{z}_u^\star = \mathbb{I}(\boldsymbol{x}_u^\star)$;
end for
return $f^\star$ and $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$;

Proposition 1. *Algorithm 1 recovers the optimal solution to Problem (1) in $O(\delta\ \omega^2\ n\ 2^n)$ time and $O(\omega^2\ n\ 2^n)$ memory.*

Proof. We start with the correctness proof.

Correctness proof. The algorithm computes the local parametric cost f_u for each node $u \in \{1, \ldots, n\}$. For the base case $u = 1$, there are no preceding subproblems; consequently, the algorithm computes f_1 according to Line 2, which coincides with the local parametric cost defined in Equation (4). For $u > 1$, the algorithm inductively applies Lemma 2 to correctly construct the local parametric cost function f_u. After constructing f_n, the algorithm evaluates the optimal cost $f^\star$, and an optimal solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ is obtained by backtracking through the local parametric costs.

Complexity proof. We analyze the runtime by examining each step of the algorithm. The labeling step in Line 1 requires $\mathcal{O}(n\omega^2)$ time and $\mathcal{O}(n\omega)$ memory, as shown in Section 2.1. Initializing the first local parametric cost f_1 requires $\mathcal{O}(\omega^2)$ time and memory (Line 2).

Next, we analyze the complexity of the first `for` loop (Lines 3–6). For each $u = 1, \ldots, n-1$, the function g_u can be obtained by separately minimizing each piece of f_u with and without the indicator variable. Since f_u contains at most 2^{n_u} pieces (Lemma 1), computing g_u requires $\mathcal{O}((2\omega^2)2^{n_u}) = \mathcal{O}(\omega^2\ 2^{n_u})$ time and memory. In Line 5, the function f_{u+1} is computed according to (6), which we rewrite here for convenience:

$$f_{u+1}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}}) = h_{u+1}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}}) + \sum_{v \in \mathrm{par}_{\mathsf{T}}(u+1)} \Big(g_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) - \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v \setminus v}) \Big).$$

The function h_{u+1} and the collection $\{\phi_v : v \in \mathrm{par}_{\mathsf{T}}(u+1)\}$ are single-piece quadratic functions. They are computed using Equations (7a) and (7c), respectively, and each requires $\mathcal{O}(\omega^2)$ time and memory. Moreover, the functions $\{g_v : v \in \mathrm{par}_{\mathsf{T}}(u+1)\}$ have already been computed in Line 4. Constructing a single piece of f_{u+1} proceeds by selecting one piece from each parent term $g_v - \phi_v$ for all $v \in \mathrm{par}_{\mathsf{T}}(u+1)$. Since $|\mathrm{par}_{\mathsf{T}}(u+1)| \leq \Delta_1 \leq \delta$, computing each piece incurs $\mathcal{O}(\delta\ \omega^2)$ time and $\mathcal{O}(\omega^2)$ memory. By Lemma 1, f_{u+1} has at most $2^{n_{u+1}}$ pieces; hence, the total cost of computing f_{u+1} is $\mathcal{O}(\delta\ \omega^2\ 2^{n_{u+1}})$ time and $\mathcal{O}(\omega^2\ 2^{n_{u+1}})$ memory. Since the first `for` loop runs for $n-1$ iterations, it incurs a cost of $\mathcal{O}\big(\sum_{u=1}^{n-1} \delta\ \omega^2\ 2^{n_{u+1}}\big) = \mathcal{O}(n\ \delta\ \omega^2\ 2^n)$ time and $\mathcal{O}(n\ \omega^2\ 2^n)$ memory.

Obtaining $f^\star$, $\boldsymbol{x}_n^\star$, and $\boldsymbol{z}_n^\star$ in Line 7 requires $\mathcal{O}(2^n)$ time, since f_n consists of at most 2^n pieces. Each iteration of the second `for` loop (Lines 8–10), requires $\mathcal{O}(\omega^2\ 2^{n_u})$ time; we omit the details for brevity. Combining all these steps, we conclude that the algorithm runs in $\mathcal{O}(\delta\ \omega^2\ n\ 2^n)$ time and $\mathcal{O}(\omega^2\ n\ 2^n)$ memory.

□

5 Pruning

The proposed DP approach for solving Problem (1) can quickly become intractable, as the number of quadratic functions required to characterize the local parametric costs may grow exponentially. In this section, we aim to address this challenge. We begin by presenting the following lemma, establishing that the optimal solution to Problem (1) lies within a bounded region. The proof relies on a result introduced later (Lemma 5) and is deferred to Appendix A.3.

Lemma 3. *Let $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ be an optimal solution to Problem (1). Define $C_1 := \frac{1}{\mu_{\min}} \max\left\{1, \frac{(1+\sqrt{\kappa_2})^2}{2\kappa_2}\right\}$ and $\rho := \frac{\sqrt{\kappa_2}-1}{\sqrt{\kappa_2}+1}$. We have $\|\boldsymbol{x}^\star\|_\infty \leq U$, where:*

- $U = \frac{2wC_1}{1-\rho}\,\|\boldsymbol{c}\|_\infty$ *if* $\boldsymbol{Q}$ *is banded with bandwidth* w*;*
- $U = \frac{\delta\gamma!\,C_1}{(1-\rho)^{\gamma+1}}\,\|\boldsymbol{c}\|_\infty$ *if the polynomial volume growth (Assumption 1) is satisfied.*

An important feature of the above lemma is that it provides an element-wise ℓ_∞-norm bound on the optimal solution, rather than a more conventional ℓ_2-norm bound. As will be explained, this finer, coordinate-wise control is crucial for our algorithmic development. While Bertsimas et al. [12, Theorem 2.1] also derive an ℓ_∞-norm bound, their result does not exploit the sparsity structure of $\boldsymbol{Q}$, and consequently scales with the problem dimension. In contrast, Lemma 3 leverages this structure and avoids explicit dimensional dependence under the stated assumptions.

That said, even the bound in Lemma 3 can be conservative in practice. In many applications, sharper instance-specific bounds are readily available. For example, in the exponential smoothing model with outlier correction, one may safely set $U = \max_t |\boldsymbol{y}_t|$, the maximum absolute magnitude of the observed signal. Such bounds are often substantially tighter. Our computational results indicate that even the general bound remains practically effective, but incorporating tighter bounds can yield substantial computational gains.

While characterizing the local parametric cost f_u over the entire $\mathbb{R}^{\tau+1}$ may require exponentially many quadratic functions, the above lemma implies that it suffices to characterize this function only within the bounded region $\mathcal{D} = \{\boldsymbol{x} : \|\boldsymbol{x}\|_\infty \le U\}$. Recalling (5), this implies that any quadratic function $p_{u,\boldsymbol{s}}$ that satisfies $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) > f_u(\boldsymbol{\alpha}_{\mathcal{B}_u})$ within the region $\mathcal{D}_u = \{\boldsymbol{\alpha}_{\mathcal{B}_u} : \|\boldsymbol{\alpha}_{\mathcal{B}_u}\|_\infty \le U\}$ can be safely discarded.

Definition 3 (Relevant and irrelevant functions)**.** Let $f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \min_{s\in\{0,1\}^{n_u}} \{p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})\}$. A function $p_{u,\boldsymbol{s}}$ is called irrelevant if $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) > f_u(\boldsymbol{\alpha}_{\mathcal{B}_u})$ within the region $\mathcal{D}_u = \{\boldsymbol{\alpha}_{\mathcal{B}_u} : \|\boldsymbol{\alpha}_{\mathcal{B}_u}\|_\infty \le U\}$. Conversely, $p_{u,\boldsymbol{s}}$ is called relevant if $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) = f_u(\boldsymbol{\alpha}_{\mathcal{B}_u})$ for some $\boldsymbol{\alpha}_{\mathcal{B}_u} \in \mathcal{D}_u$.

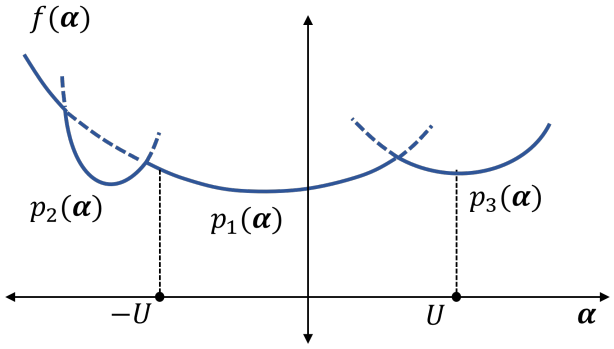

Figure 6: The piecewise quadratic function $f(\boldsymbol{\alpha}) = \min\{p_1(\boldsymbol{\alpha}), p_2(\boldsymbol{\alpha}), p_3(\boldsymbol{\alpha})\}$ is shown by the solid blue line. Here, p_2 is irrelevant since $p_2(\boldsymbol{\alpha}) > f(\boldsymbol{\alpha})$ for every $-U \le \boldsymbol{\alpha} \le U$.

Figure 6 illustrates an example distinguishing relevant and irrelevant functions. To separate the two, we present the following lemma, which provides a lower bound on the intersection points of two quadratic functions. The proof of this lemma is presented in Appendix A.4.

Lemma 4. *Let* $p_1, p_2 : \mathbb{R}^{\tau+1} \to \mathbb{R}$ *be two quadratic functions of the form* $p_1(\boldsymbol{\alpha}) := \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{A}_1\boldsymbol{\alpha} + \boldsymbol{b}_1^\top\boldsymbol{\alpha} + d_1$ *and* $p_2(\boldsymbol{\alpha}) := \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{A}_2\boldsymbol{\alpha} + \boldsymbol{b}_2^\top\boldsymbol{\alpha} + d_2$*. Let* $\bar{a} \ge \frac{1}{2}\|\boldsymbol{A}_1 - \boldsymbol{A}_2\|_{1,1}, \bar{b} \ge \|\boldsymbol{b}_1 - \boldsymbol{b}_2\|_1$ *and* $|d_1 - d_2| \ge \bar{d}$*.*

Let $\hat{\boldsymbol{\alpha}}$ be a real root of the equation $p_1(\boldsymbol{\alpha}) - p_2(\boldsymbol{\alpha}) = 0$. If no real root exists, we set $\hat{\boldsymbol{\alpha}} = +\infty$. Then, we have

$$\|\hat{\boldsymbol{\alpha}}\|_\infty \geq L(p_1, p_2), \qquad \text{where } L(p_1, p_2) := \begin{cases} \frac{-\bar{b}+\sqrt{\bar{b}^2+4\bar{a}\bar{d}}}{2\bar{a}} & \text{if } \bar{a} \neq 0, \\ \frac{\bar{d}}{\bar{b}} & \text{otherwise.} \end{cases} \tag{8}$$

The above lemma provides a criterion for identifying irrelevant functions: given a pair of sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$ and their corresponding quadratic functions $p_{u,\boldsymbol{s}^{(1)}}$ and $p_{u,\boldsymbol{s}^{(2)}}$, one can compute $L(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}})$ using Lemma 4. If $L(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}}) > U$ and $p_{u,\boldsymbol{s}^{(1)}}(\mathbf{0}) > p_{u,\boldsymbol{s}^{(2)}}(\mathbf{0})$ (or $p_{u,\boldsymbol{s}^{(1)}}(\mathbf{0} < p_{u,\boldsymbol{s}^{(2)}}(\mathbf{0})$, respectively), then $p_{u,\boldsymbol{s}^{(1)}}$ (or $p_{u,\boldsymbol{s}^{(2)}}$, respectively) can be declared irrelevant.

Having established the criterion for identifying irrelevant functions, we now describe how this criterion can be incorporated into Algorithm 1. For some $1 \leq u \leq n-1$, suppose that the local parametric function f_{u+1} has already been computed in Line 5 and consists of N quadratic pieces; that is, $f_{u+1}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}}) = \min_{\boldsymbol{s}\in\mathcal{P}_{u+1}}\{p_{u+1,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}})\}$ with $|\mathcal{P}_{u+1}| = N$. To identify and discard irrelevant pieces within this set, a pruning subroutine can be inserted immediately after Line 5. This subroutine computes $L(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}})$ for every pair $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \mathcal{P}_{u+1}$ and removes any index $\boldsymbol{s}^{(1)} \in \mathcal{P}_{u+1}$ whose corresponding function $p_{u,\boldsymbol{s}^{(1)}}$ satisfies $L(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}}) > U$ and $p_{u,\boldsymbol{s}^{(1)}}(\mathbf{0}) > p_{u,\boldsymbol{s}^{(2)}}(\mathbf{0})$. The resulting refined algorithm is presented in Algorithm 2. The key distinction between this version and Algorithm 1 lies in the inclusion of the `PRUNE` subroutine (Algorithm 3), which performs the pruning operation described above. For a function with N quadratic pieces, the `PRUNE` subroutine requires $\mathcal{O}(N^2)$ pairwise comparisons between the quadratic pieces. Each comparison involves computing the corresponding L, which can be done in $\mathcal{O}(\omega^2)$ time and memory. Hence, the overall time and memory complexities of the `PRUNE` subroutine are $\mathcal{O}(\omega^2 N^2)$ and $\mathcal{O}(\omega^2 N)$, respectively. We note in passing that the time complexity of this subroutine can be improved to $\mathcal{O}(\omega^2 N)$ when the tree decomposition of $\boldsymbol{Q}$ is a path. Further discussion on this special case is deferred to Section 7 and Appendix B.

Algorithm 2 Parametric algorithm for graphs with bounded treewidth matrices

Input: Problem (1) and a tree decomposition T of $\text{supp}(\boldsymbol{Q})$ with width ω and upper bound U;
Output: The optimal cost $f^\star$ and an optimal solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ of Problem (1);

1: Label the nodes of $\text{supp}(\boldsymbol{Q})$ according to the scheme described in Section 2.1;
2: Set $f_1(\boldsymbol{\alpha}_{\mathcal{B}_1}) = \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_1}^\top \boldsymbol{Q}_{\mathcal{B}_1,\mathcal{B}_1}\boldsymbol{\alpha}_{\mathcal{B}_1} + \boldsymbol{\alpha}_{\mathcal{B}_1}^\top \boldsymbol{c}_{\mathcal{B}_1}$.
3: **for** $u = 1, \ldots, n-1$ **do**
4: Calculate g_u from f_u via Equation (7b);
5: Calculate f_{u+1} via Equation (6);
6: Set $f_{u+1} = \texttt{PRUNE}(f_{u+1}, U)$;
7: **end for**
8: Obtain $f^\star = \min\limits_{\boldsymbol{\alpha}_n\in\mathbb{R}} \{f_n(\boldsymbol{\alpha}_n) + \boldsymbol{\lambda}_n \mathbb{I}(\boldsymbol{\alpha}_n)\}$, $\boldsymbol{x}_n^\star = \operatorname{argmin}\limits_{\boldsymbol{\alpha}_n\in\mathbb{R}} \{f_n(\boldsymbol{\alpha}_n) + \boldsymbol{\lambda}_n \mathbb{I}(\boldsymbol{\alpha}_n)\}$, and $\boldsymbol{z}_n^\star = \mathbb{I}(\boldsymbol{x}_n^\star)$;
9: **for** $u = n-1, \ldots, 1$ **do**
10: Set $\boldsymbol{x}_u^\star = \operatorname{argmin}\limits_{\boldsymbol{\alpha}_u\in\mathbb{R}} \left\{ f_u(\boldsymbol{\alpha}_u, \boldsymbol{x}^\star_{\mathcal{B}_u\setminus u}) + \boldsymbol{\lambda}_u \mathbb{I}(\boldsymbol{\alpha}_u) \right\}$ and $\boldsymbol{z}_u^\star = \mathbb{I}(\boldsymbol{x}_u^\star)$;
11: **end for**
12: **return** $f^\star$ and $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$;

While the theoretical guarantees of this pruning strategy will be discussed in detail in the next section, here we briefly highlight the empirical effectiveness of this approach. Figure 7 illustrates

Algorithm 3 `PRUNE`(f, U)

Input: function f characterized by its quadratic pieces $\{p_s\}_{s\in\mathcal{I}}$ and the constant U.
Output: pruned function f^{prune}

Initialize $\mathcal{S}^{\text{prune}}$ as an empty list;
Convert $\mathcal{I}$ into a list and store it in $\mathcal{S}$;
while $\mathcal{S} \neq \emptyset$ **do**
    Set i as the first element of $\mathcal{S}$ and delete i from $\mathcal{S}$;
    Add i to $\mathcal{S}^{\text{prune}}$; ▷ Assume p_i is relevant
    for $j \in \mathcal{S}$ **do**
        Calculate $L(p_i, p_j)$;
        if $L > U$ **then** ▷ Either p_i or p_j is irrelevant
            if $p_i(0) < p_j(0)$ **then** ▷ p_j is irrelevant
                Delete j from $\mathcal{S}$;
            else ▷ p_i is irrelevant
                Delete i from $\mathcal{S}^{\text{prune}}$;
                `break`; ▷ Exit for-loop
            end if
        end if
    end for
end while
return f^{prune} characterized by the quadratic pieces $\{p_i\}_{i\in\mathcal{S}^{\text{prune}}}$;

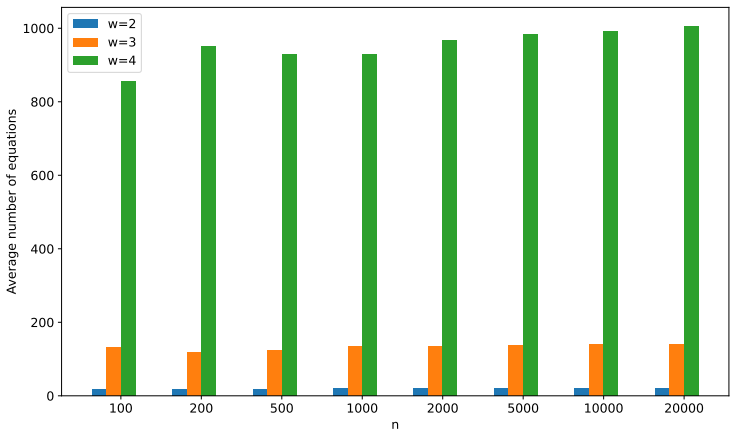

Figure 7: Number of equations computed for the pruned parametric algorithm for different values of n and w. The reported results are averaged over 5 trials.

the performance of the pruning subroutine on synthetically generated banded matrices of varying sizes n between 100 and 20,000, and varying bandwidths w between 2 and 4 (corresponding to treewidths between 2 and 4). The figure shows the number of quadratic equations retained after pruning, averaged over local parametric costs and across five independent trials. Notably, even for the largest instance with $n = 20{,}000$, the average number of quadratic equations after pruning does not exceed 1,100, whereas the number of quadratic equations without pruning scales as $2^{20{,}000}$. This shows the effectiveness of the pruning subroutine in eliminating irrelevant equations.

6 Theoretical analysis

The empirical performance of the proposed parametric algorithm with pruning suggests that, among exponentially many quadratic equations, only a small subset is relevant for characterizing the local parametric cost f_u. This observation implies that, for most pairs of quadratic functions $(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}})$, their intersection occurs outside the relevant region $\mathcal{D}_u = \{\boldsymbol{\alpha}_{\mathcal{B}_u} : \|\boldsymbol{\alpha}_{\mathcal{B}_u}\|_\infty \leq U\}$. In this section, we provide a theoretical justification for this key observation. In particular, we show that the greater the similarity between the sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$, the larger the roots of the difference $p_{u,\boldsymbol{s}^{(1)}} - p_{u,\boldsymbol{s}^{(2)}}$ become. We quantify the similarity of two sparsity patterns using the notion of *m-similarity*. For any bag $\mathcal{B}_u$ and positive integer m, recall that $\mathcal{V}_{u,m} = \{i \in \mathcal{J}_u \mid \mathrm{dist}(i, \mathcal{B}_u) \leq m\}$ is the m-neighborhood of $\mathcal{B}_u$ within the induced subgraph $\mathrm{supp}_u(\boldsymbol{Q})$, where $\mathcal{J}_u$ is the set of nodes in $\mathrm{supp}_u(\boldsymbol{Q})$ excluding those in $\mathcal{B}_u$. Recall that $\pi : \mathcal{I} \to \{1, \ldots, |\mathcal{I}|\}$ is the canonical indexing map that assigns to each $i \in \mathcal{I}$ its corresponding row/column position within the submatrix $\boldsymbol{Q}_{\mathcal{I},\mathcal{I}}$.

Definition 4 (m-similarity)**.** The two sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$ are called m-similar with respect to bag $\mathcal{B}_u$ if $\boldsymbol{s}^{(1)}_{\pi_u(i)} = \boldsymbol{s}^{(2)}_{\pi_u(i)}$ for all $i \in \mathcal{V}_{u,m}$. When the reference bag $\mathcal{B}_u$ is clear from the context, we simply say that $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$ are m-similar.

We will show that, for two m-similar sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$, the norm of the root(s) of the difference $p_{u,\boldsymbol{s}^{(1)}} - p_{u,\boldsymbol{s}^{(2)}}$ grows exponentially with m. To this end, we first establish a key decay property of $\boldsymbol{Q}^{-1}$. Specifically, we show that the entries of the inverse of any principal submatrix of $\boldsymbol{Q}$ decay exponentially with the distance between the corresponding nodes in its support graph.

Lemma 5. *Let $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix, and $\mathcal{I} \subseteq \{1, \ldots, n\}$ any subset of its rows/columns. For any $i, j \in \mathcal{I}$, we have*

$$\left| \left[\boldsymbol{Q}^{-1}_{\mathcal{I},\mathcal{I}} \right]_{\pi(i),\pi(j)} \right| \leq C_1 \rho^{\mathrm{dist}(i,j)}, \quad \textit{where} \quad C_1 := \frac{1}{\mu_{\min}} \max\left\{1, \frac{(1+\sqrt{\kappa_2})^2}{2\kappa_2}\right\}, \ \textit{and } \rho := \frac{\sqrt{\kappa_2}-1}{\sqrt{\kappa_2}+1}. \tag{9}$$

The proof is presented in Appendix A.5. The exponential off-diagonal decay of inverses of banded matrices has been extensively studied [28]. The above lemma extends this result to a more general setting, in which the decay structure of the inverse is determined by the sparsity pattern of the matrix.

Recall from Lemma 1 that the local parametric cost f_u can be expressed as $\min_{\boldsymbol{s} \in \{0,1\}^{n_u}} \{ p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) \}$,

where each $p_{u,\boldsymbol{s}}$ is given as:

$$p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}\boldsymbol{A}_{u,\boldsymbol{s}}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{b}_{u,\boldsymbol{s}}^{\top}\boldsymbol{\alpha}_{\mathcal{B}_u} + d_{u,\boldsymbol{s}},$$
$$\text{where} \begin{cases} \boldsymbol{A}_{u,s} & = \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u} - \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}} \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^{\top} \\ \boldsymbol{b}_{u,s} & = \boldsymbol{c}_{\mathcal{B}_u} - \boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^{T} \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^{\top} \\ d_{u,s} & = -\frac{1}{2}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^{\top}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}})^{-1}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}} + \sum_{i\in\mathcal{J}_{u,\boldsymbol{s}}}\lambda_i. \end{cases} \tag{10}$$

Next, we partition the elements of $\mathcal{J}_{u,\boldsymbol{s}} = \{i \in \mathcal{J}_u \mid s_i = 1\}$ according to their graph distance from the bag $\mathcal{B}_u$. For an integer $m > 0$, let

$$\mathcal{V}_{u,s,m} = \mathcal{J}_{u,\boldsymbol{s}} \cap \mathcal{V}_{u,m}, \qquad \mathcal{W}_{u,s,m} = \mathcal{J}_{u,\boldsymbol{s}} \setminus \mathcal{V}_{u,m}. \tag{11}$$

Since our subsequence analysis holds for any choice of $u \in \{1,\dots,n\}$, $\boldsymbol{s} \in \{0,1\}^{n_u}$, and $m \in \{1,\dots,n_u\}$, for simplicity, we drop the subscripts and write $\mathcal{V} = \mathcal{V}_{u,\boldsymbol{s},m}$ and $\mathcal{W} = \mathcal{W}_{u,\boldsymbol{s},m}$. Intuitively, the set $\mathcal{V}$ contains all nodes in $\mathcal{J}_{u,\boldsymbol{s}}$ that are within distance m from the nodes in the bag $\mathcal{B}_u$, while $\mathcal{W}$ collects all the nodes in $\mathcal{J}_{u,\boldsymbol{s}}$ whose distance from $\mathcal{B}_u$ exceeds m. As an illustrative example, consider the graph in Figure 2. When $u = 8$, $m = 2$, and $\boldsymbol{s} = [1,1,\dots,1]^{\top}$, we obtain $\mathcal{V} = \{2,3,\dots,7\}$ and $\mathcal{W} = \{1\}$.

Given $\mathcal{V}$ and $\mathcal{W}$, the matrix $\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}$ and its inverse admit the following block structures

$$\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}} = \begin{bmatrix} \boldsymbol{Q}_{\mathcal{W},\mathcal{W}} & \boldsymbol{Q}_{\mathcal{W},\mathcal{V}} \\ \boldsymbol{Q}_{\mathcal{V},\mathcal{W}} & \boldsymbol{Q}_{\mathcal{V},\mathcal{V}} \end{bmatrix}, \tag{12}$$

$$\boldsymbol{Q}^{-1}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}} = \begin{bmatrix} (\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1} & -(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}} \\ -\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}})^{-1} & \boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}} + \boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}} \end{bmatrix}, \tag{13}$$

where $\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}} := \boldsymbol{Q}_{\mathcal{W},\mathcal{W}} - \boldsymbol{Q}_{\mathcal{W},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}$ is the Schur complement of block $\boldsymbol{Q}_{\mathcal{W},\mathcal{W}}$ in $\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}$. The main motivation for analyzing the block structure of $\boldsymbol{Q}^{-1}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}$ is that, according to (10), the coefficients $\boldsymbol{A}_{u,\boldsymbol{s}}$ and $\boldsymbol{b}_{u,\boldsymbol{s}}$ contain terms involving the block product $\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}$, whose norm, as formally stated in the next lemma, decays exponentially with m.

Lemma 6. *We have*

$$\left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}\right\|_2 \leq \frac{2\mu_{\max}^2}{\mu_{\min}}\sqrt{\Delta_1\Delta_m}\rho^{m-1}, \tag{14}$$

where $0 < \rho < 1$ is the constant from Lemma 5.

Proof. Let $\pi : \mathcal{V} \to \{1,\dots,|\mathcal{V}|\}$ be the indexing map that assigns to each $i \in \mathcal{V}$ its corresponding row/column position within the submatrix $\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}$. Let $\mathcal{I} = \{i \in \mathcal{V} \mid \text{dist}(i,\mathcal{B}_u) = 1\}$ and $\mathcal{J} = \{j \in \mathcal{V} \mid \text{dist}(j,\mathcal{B}_u) = m\}$. Here, $\mathcal{I}$ is the set of nodes in $\mathcal{V}$ adjacent to $\mathcal{B}_u$, and $\mathcal{J}$ is the set of nodes at a distance m from $\mathcal{B}_u$. One can write

$$\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}} = \boldsymbol{Q}_{\mathcal{B}_u,\pi(\mathcal{I})}\left[\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\right]_{\pi(\mathcal{I}),\pi(\mathcal{J})}\boldsymbol{Q}_{\pi(\mathcal{J}),\mathcal{W}}.$$

Indeed, if $\mathcal{I}$ or $\mathcal{J}$ is empty, then $\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}} = 0$. Therefore, without loss of generality, we assume that neither set is empty. By Lemma 5, for any $i \in \mathcal{I}$ and $j \in \mathcal{J}$, we have

$$\left|\left[\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\right]_{\pi(i),\pi(j)}\right| \leq C_1\rho^{\mathrm{dist}(i,j)} = C_1\rho^{m-1}.$$

This implies that

$$\left\|\left[\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\right]_{\pi(\mathcal{I}),\pi(\mathcal{J})}\right\|_2 \leq \sqrt{|\mathcal{I}||\mathcal{J}|}C_1\rho^{m-1} \leq \sqrt{\Delta_1\Delta_m}C_1\rho^{m-1}.$$

The last inequality follows from the fact that $\mathcal{J} \subseteq \mathcal{V} \subseteq \mathcal{V}_{u,m}$, which implies $|\mathcal{J}| \leq |\mathcal{V}_{u,m}| \leq \Delta_m$. By the same reasoning, we have $|\mathcal{I}| \leq \Delta_1$. Therefore,

$$\begin{aligned}
\left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}\right\|_2 &= \left\|\boldsymbol{Q}_{\mathcal{B}_u,\pi(\mathcal{I})}\left[\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\right]_{\pi(\mathcal{I}),\pi(\mathcal{J})}\boldsymbol{Q}_{\pi(\mathcal{J}),\mathcal{W}}\right\|_2 \\
&\leq \|\boldsymbol{Q}_{\mathcal{B}_u,\pi(\mathcal{I})}\|_2 \left\|\left[\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\right]_{\pi(\mathcal{I}),\pi(\mathcal{J})}\right\|_2 \|\boldsymbol{Q}_{\pi(\mathcal{J}),\mathcal{W}}\|_2 \\
&\leq \mu_{\max}^2 \left\|\left[\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\right]_{\pi(\mathcal{I}),\pi(\mathcal{J})}\right\|_2 \\
&\leq \mu_{\max}^2\sqrt{\Delta_1\Delta_m}C_1\rho^{m-1} \\
&= \frac{2\mu_{\max}^2}{\mu_{\min}}\sqrt{\Delta_1\Delta_m}\rho^{m-1}.
\end{aligned}$$

The second inequality uses the property that $\mu_{\max}$ bounds the spectral norm of every submatrix of $\boldsymbol{Q}$. The last inequality follows from the fact that $\frac{(1+\sqrt{\kappa_2})^2}{2\kappa_2} \leq 2$, which implies that $C_1 = \frac{1}{\mu_{\min}} \max\left\{1, \frac{(1+\sqrt{\kappa_2})^2}{2\kappa_2}\right\} \leq \frac{2}{\mu_{\min}}$. □

The above lemma implies that $\left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}}\right\|_2$ decays nearly exponentially with m, provided that Δ_1 and Δ_m do not grow exponentially. As it turns out, this condition is ensured under the polynomial volume growth (Assumption 1). This property will play an important role in establishing our final guarantees.

Armed with the above result, we can now establish that, for any two m-similar sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$, the quadratic and linear coefficients of their corresponding quadratic pieces $p_{u,\boldsymbol{s}^{(1)}}$ and $p_{u,\boldsymbol{s}^{(2)}}$ are exponentially close to each other.

Lemma 7. *Let $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$ be two m-similar sparsity patterns, and let $p_{u,\boldsymbol{s}^{(1)}}$ and $p_{u,\boldsymbol{s}^{(2)}}$ be their corresponding quadratic pieces defined in (10). Then, the following bounds hold:*

$$\begin{aligned}
\|\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}}\|_{1,1} &\leq 4\kappa_2^4\ (\omega+1)^{3/2}\ \Delta_1\Delta_m\ \rho^{2m-2}, \\
\|\boldsymbol{b}_{u,\boldsymbol{s}^1} - \boldsymbol{b}_{u,\boldsymbol{s}^2}\|_1 &\leq 4\kappa_2^2(1+\kappa_\infty)\ U\ (\omega+1)^{3/2}\ \sqrt{\Delta_1\Delta_m}\ \rho^{m-1}.
\end{aligned}$$

Proof. For each $\boldsymbol{s}^{(i)}$ with $i \in \{1,2\}$, consider the sets $\mathcal{V}_{u,\boldsymbol{s}^{(i)},m} := \mathcal{J}_{u,\boldsymbol{s}^{(i)}} \cap \mathcal{V}_{u,m}$ and $\mathcal{W}_{u,\boldsymbol{s}^{(i)},m} := \mathcal{J}_{u,\boldsymbol{s}^{(i)}} \setminus \mathcal{V}_{u,m}$. Since $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$ are m-similar, it follows that $\mathcal{V}_{u,\boldsymbol{s}^{(1)},m} = \mathcal{V}_{u,\boldsymbol{s}^{(2)},m}$. For notational convenience, we drop the subscripts and write $\mathcal{V} := \mathcal{V}_{u,\boldsymbol{s}^{(1)},m} = \mathcal{V}_{u,\boldsymbol{s}^{(2)},m}$, $\mathcal{W}^{(1)} := \mathcal{W}_{u,\boldsymbol{s}^{(1)},m}$, and

$\mathcal{W}^{(2)} := \mathcal{W}_{u,\boldsymbol{s}^{(2)},m}$. Combining the definition of $p_{u,\boldsymbol{s}^{(1)}}$ and $p_{u,\boldsymbol{s}^{(2)}}$ from (10) with the block structure of $\boldsymbol{Q}^{-1}_{\mathcal{J}_{u,\boldsymbol{s}^{(i)}},\mathcal{J}_{u,\boldsymbol{s}^{(i)}}}$ in (13), it follows that

$$\begin{aligned}\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}} =& - \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(1)}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(1)}},\mathcal{J}_{u,\boldsymbol{s}^{(1)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\\ &+ \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(2)}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(2)}},\mathcal{J}_{u,\boldsymbol{s}^{(2)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u},\end{aligned}$$

where we use the fact that $\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}^{(i)}}} = \begin{bmatrix}\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{W}^{(i)}} & \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\end{bmatrix}$ and $\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{W}^{(i)}} = 0$. The above equality yields

$$\begin{aligned}\|\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}}\|_2 \le& \left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(1)}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(1)}},\mathcal{J}_{u,\boldsymbol{s}^{(1)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\right\|_2\\ &+ \left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(2)}}(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(2)}},\mathcal{J}_{u,\boldsymbol{s}^{(2)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\right\|_2\\ \le& \left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(1)}}\right\|_2^2 \left\|(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(1)}},\mathcal{J}_{u,\boldsymbol{s}^{(1)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\right\|_2\\ &+ \left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(2)}}\right\|_2^2 \left\|(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(2)}},\mathcal{J}_{u,\boldsymbol{s}^{(2)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\right\|_2.\end{aligned}$$

From Lemma 6, we have

$$\max\left\{\left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(1)}}\right\|_2, \left\|\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{W}^{(2)}}\right\|_2\right\} \le \frac{2\mu_{\max}^2}{\mu_{\min}}\sqrt{\Delta_1\Delta_m}\rho^{m-1}.$$

On the other hand,

$$\begin{aligned}\frac{1}{\mu_{\min}(\boldsymbol{Q})} = \mu_{\max}\left(\boldsymbol{Q}^{-1}\right) = \max_{\boldsymbol{\zeta}\neq\boldsymbol{0}}\left\{\frac{\boldsymbol{\zeta}^\top\boldsymbol{Q}^{-1}\boldsymbol{\zeta}}{\boldsymbol{\zeta}^\top\boldsymbol{\zeta}}\right\} \ge \max_{\bar{\boldsymbol{\zeta}}\neq\boldsymbol{0}}\left\{\frac{\bar{\boldsymbol{\zeta}}^\top\left[\boldsymbol{Q}^{-1}\right]_{\mathcal{W}^{(i)},\mathcal{W}^{(i)}}\bar{\boldsymbol{\zeta}}}{\bar{\boldsymbol{\zeta}}^\top\bar{\boldsymbol{\zeta}}}\right\}\\ = \left\|(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(i)}},\mathcal{J}_{u,\boldsymbol{s}^{(i)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}})^{-1}\right\|_2.\end{aligned} \tag{15}$$

Combining the above inequalities, we obtain

$$\|\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}}\|_2 \le \frac{4\mu_{\max}^4}{\mu_{\min}^3}\,\Delta_1\Delta_m\,\rho^{2m-2} \le 4\kappa_2^4\,\Delta_1\Delta_m\,\rho^{2m-2},$$

where the last inequality follows from the fact that $\mu_{\min} \le 1$, and hence, $\mu_{\max} \le \kappa_2$. The proof of the first statement is completed after noting that $\|\boldsymbol{A}_{\boldsymbol{s}^1} - \boldsymbol{A}_{\boldsymbol{s}^2}\|_{1,1} \le (\omega+1)^{3/2}\|\boldsymbol{A}_{u,s^{(1)}} - \boldsymbol{A}_{u,s^{(2)}}\|_2$.

Next, we provide the proof of the second statement. Again, from (10) and the block structure of $\boldsymbol{Q}^{-1}_{\mathcal{J}_{u,\boldsymbol{s}^{(i)}},\mathcal{J}_{u,\boldsymbol{s}^{(i)}}}$ in (13), we have

$$\begin{aligned}\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}} =& -\left(\boldsymbol{c}_{\mathcal{W}^{(1)}} - \boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{c}_{\mathcal{V}}\right)^\top\left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(1)}},\mathcal{J}_{u,\boldsymbol{s}^{(1)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}\right)^{-1}\boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\\ &+\left(\boldsymbol{c}_{\mathcal{W}^{(2)}} - \boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{c}_{\mathcal{V}}\right)^\top\left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(2)}},\mathcal{J}_{u,\boldsymbol{s}^{(2)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}\right)^{-1}\boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u},\end{aligned}$$

which implies

$$\begin{aligned}&\|\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}}\|_\infty\\ &\le \left\|\boldsymbol{c}_{\mathcal{W}^{(1)}} - \boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{c}_{\mathcal{V}}\right\|_\infty\left\|\left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(1)}},\mathcal{J}_{u,\boldsymbol{s}^{(1)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}\right)^{-1}\right\|_\infty\left\|\boldsymbol{Q}_{\mathcal{W}^{(1)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\right\|_\infty\\ &+\left\|\boldsymbol{c}_{\mathcal{W}^{(2)}} - \boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{c}_{\mathcal{V}}\right\|_\infty\left\|\left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(2)}},\mathcal{J}_{u,\boldsymbol{s}^{(2)}}}/\boldsymbol{Q}_{\mathcal{V},\mathcal{V}}\right)^{-1}\right\|_\infty\left\|\boldsymbol{Q}_{\mathcal{W}^{(2)},\mathcal{V}}\boldsymbol{Q}^{-1}_{\mathcal{V},\mathcal{V}}\boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u}\right\|_\infty.\end{aligned}$$

We now control each term on the right-hand side separately. For any $i = 1, 2$, we have

$$\left\| \boldsymbol{c}_{\mathcal{W}^{(i)}} - \boldsymbol{Q}_{\mathcal{W}^{(i)},\mathcal{V}} \boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1} \boldsymbol{c}_{\mathcal{V}} \right\|_\infty \leq \|\boldsymbol{c}\|_\infty + \|\boldsymbol{Q}\|_\infty \left\| \boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1} \right\|_\infty \|\boldsymbol{c}_{\mathcal{V}}\|_\infty \leq \|\boldsymbol{c}\|_\infty + U \ \|\boldsymbol{Q}\|_\infty.$$

The first inequality follows from the sub-multiplicative property of the induced ∞-norm and the fact that $\left\| \boldsymbol{Q}_{\mathcal{W}^{(i)},\mathcal{V}} \right\|_\infty \leq \|\boldsymbol{Q}\|_\infty$. The second inequality follows from $\left\| \boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1} \right\|_\infty \|\boldsymbol{c}_{\mathcal{V}}\|_\infty \leq U$ (see the proof of Lemma 3). Similarly, from (13), we obtain

$$\left\| \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}^{(i)}},\mathcal{J}_{u,\boldsymbol{s}^{(i)}}} / \boldsymbol{Q}_{\mathcal{V},\mathcal{V}} \right)^{-1} \right\|_\infty = \left\| \left[\boldsymbol{Q}^{-1} \right]_{\mathcal{W}^{(i)},\mathcal{W}^{(i)}} \right\|_\infty \leq \left\| \boldsymbol{Q}^{-1} \right\|_\infty.$$

Finally,

$$\left\| \boldsymbol{Q}_{\mathcal{W}^{(i)},\mathcal{V}} \boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1} \boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u} \right\|_\infty \leq \sqrt{\omega+1} \left\| \boldsymbol{Q}_{\mathcal{W}^{(i)},\mathcal{V}} \boldsymbol{Q}_{\mathcal{V},\mathcal{V}}^{-1} \boldsymbol{Q}_{\mathcal{V},\mathcal{B}_u} \right\|_2 \leq \sqrt{\omega+1} \ \frac{2\mu_{\max}^2}{\mu_{\min}} \sqrt{\Delta_1 \Delta_m} \rho^{m-1},$$

where in the last inequality, we use Lemma 6. Combining these bounds, we obtain

$$\begin{aligned} \|\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}}\|_\infty &\leq 2 \left(\|\boldsymbol{c}\|_\infty + U \ \|\boldsymbol{Q}\|_\infty \right) \ \left\| \boldsymbol{Q}^{-1} \right\|_\infty \sqrt{\omega+1} \ \frac{2\mu_{\max}^2}{\mu_{\min}} \sqrt{\Delta_1 \Delta_m} \rho^{m-1} \\ &\leq 4\kappa_2^2 (1+\kappa_\infty) U \ \sqrt{\omega+1} \ \sqrt{\Delta_1 \Delta_m} \rho^{m-1}, \end{aligned}$$

where the second inequality follows from $\left\| \boldsymbol{Q}^{-1} \right\|_\infty \|\boldsymbol{c}\|_\infty \leq U$, $\kappa_\infty = \|\boldsymbol{Q}\|_\infty \left\| \boldsymbol{Q}^{-1} \right\|_\infty$, and $\frac{\mu_{\max}^2}{\mu_{\min}} \leq \kappa_2^2$. The proof is completed after noting that $\|\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}}\|_1 \leq (\omega+1) \|\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}}\|_\infty$. □

Our next objective is to analyze the term $d_{u,\boldsymbol{s}^{(1)}} - d_{u,\boldsymbol{s}^{(2)}}$ for a pair of m-similar sparsity patterns $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$. From the expression of $d_{u,\boldsymbol{s}}$ in (10), one observes that, unlike the quadratic and linear coefficients $\boldsymbol{A}_{u,\boldsymbol{s}}$ and $\boldsymbol{b}_{u,\boldsymbol{s}}$, the constant term $d_{u,\boldsymbol{s}}$ *does not* depend on any submatrix of $\boldsymbol{Q}$ associated with the bag $\mathcal{B}_u$. Consequently, contrary to the quadratic and linear terms, Lemma 6 cannot be used to establish a decaying behavior for $|d_{u,\boldsymbol{s}^{(1)}} - d_{u,\boldsymbol{s}^{(2)}}|$. This observation is precisely what enables our pruning strategy to be effective: as established in Lemma 7, both $\|\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}}\|_{1,1}$ and $\|\boldsymbol{b}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{b}_{u,\boldsymbol{s}^{(2)}}\|_1$ decay exponentially fast in m (under polynomial growth condition in Assumption 1), whereas $|d_{u,\boldsymbol{s}^{(1)}} - d_{u,\boldsymbol{s}^{(2)}}|$ does not exhibit such decay. Therefore, by invoking Lemma 4, we conclude that for any pair of m-similar sparsity patterns $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$, the quantity $L(p_{u,\boldsymbol{s}^{(1)}}, p_{u,\boldsymbol{s}^{(2)}})$ grows exponentially with m. As a result, at least one of the functions $p_{u,\boldsymbol{s}^{(1)}}$ or $p_{u,\boldsymbol{s}^{(2)}}$ becomes irrelevant for sufficiently large m. Our next lemma formalizes this intuition.

Lemma 8. *Let $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^{n_u}$ be two m-similar sparsity patterns, and let $p_{u,\boldsymbol{s}^{(1)}}$ and $p_{u,\boldsymbol{s}^{(2)}}$ denote their corresponding quadratic pieces defined in (10). Assume that the polynomial volume growth condition (Assumption 1) holds, and that $|d_{u,\boldsymbol{s}^{(1)}} - d_{u,\boldsymbol{s}^{(2)}}| \geq \eta$, for some $0 < \eta \leq 1$. Suppose*

$$m \geq \max \left\{ \frac{2\log\left(8U^2/\eta\right) + 2\log\left(\delta(\omega+1)^{3/2}\right) + 2\log\left(\kappa_2^2(1+\kappa_\infty)\right)}{\log(1/\rho)} + 2, \ \frac{2\gamma}{\log(1/\rho)}, \left(\frac{\gamma}{\log(1/\rho)} \right)^2 \right\}. \tag{16}$$

Then, $L(\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}) \geq U$; in particular, at least one of $\boldsymbol{s}^{(1)}$ or $\boldsymbol{s}^{(2)}$ is irrelevant.

Before proving the above lemma, we first present the following auxiliary claim, which will play an important role in its proof.

Claim 1. *Suppose* $\Gamma > 0$. *Then,* $\Gamma m \geq \log(m)$ *for any* $m \geq \max\{1, 2/\Gamma, (1/\Gamma)^2\}$.

Proof. First, suppose that $\Gamma \geq 1$. Then, it is easy to verify that $\Gamma m \geq m \geq \log(m)$ for $m > 0$. Now, consider the setting where $0 < \Gamma < 1$. Define $m = \xi/\Gamma$, where $\xi \geq \max\{2, 1/\Gamma\}$. Then, we have

$$\Gamma m - \log(m) = \xi - \log(\xi) - \log(1/\Gamma) \geq \log(\xi) - \log(1/\Gamma) \geq 0,$$

where the first inequality follows from the fact that $\xi \geq 2\log(\xi)$ for $\xi \geq 2$, and the last inequality follows from the fact that $\xi \geq \max\{2, 1/\Gamma\} \geq 1/\Gamma$. □

Proof. Proof of Lemma 8. Under the polynomial growth condition, we have $\Delta_1 \leq \delta$ and $\Delta_m \leq \delta m^\gamma$. Combined with Lemma 7, this implies that

$$\begin{aligned}
\|\boldsymbol{A}_{u,\boldsymbol{s}^{(1)}} - \boldsymbol{A}_{u,\boldsymbol{s}^{(2)}}\|_{1,1} &\leq 4\kappa_2^4\ (\omega+1)^{3/2}\ \delta^2 m^\gamma\ \rho^{2m-2} \\
\|\boldsymbol{b}_{u,\boldsymbol{s}^1} - \boldsymbol{b}_{u,\boldsymbol{s}^2}\|_1 &\leq 4\kappa_2^2(1+\kappa_\infty)\ U\ (\omega+1)^{3/2}\ \delta m^{\gamma/2}\ \rho^{m-1}
\end{aligned}$$

Invoking Lemma 4 with $\bar{a} = 4\kappa_2^4\ (\omega+1)^{3/2}\ \delta^2 m^\gamma\ \rho^{2m-2} > 0$, $\bar{b} = 4\kappa_2^2(1+\kappa_\infty)\ U\ (\omega+1)^{3/2}\ \delta m^{\gamma/2}\ \rho^{m-1}$, and $\bar{d} = \eta$, we obtain

$$L(\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}) = \frac{-\bar{b} + \sqrt{\bar{b}^2 + 4\bar{a}\bar{d}}}{2\bar{a}} = \frac{4\bar{a}\bar{d}}{2\bar{a}\left(\bar{b} + \sqrt{\bar{b}^2 + 4\bar{a}\bar{d}}\right)} \geq \frac{\bar{d}}{\sqrt{\bar{b}^2 + 4\bar{a}\bar{d}}} \geq \frac{\bar{d}}{\bar{b} + 2\sqrt{\bar{a}\bar{d}}}.$$

Substituting the expressions for $\bar{a}$, $\bar{b}$, and $\bar{d}$, we arrive at

$$\begin{aligned}
\frac{\bar{d}}{\bar{b} + 2\sqrt{\bar{a}\bar{d}}} &\geq \frac{\eta}{4\kappa_2^2(1+\kappa_\infty)\ U\ (\omega+1)^{3/2}\ \delta m^{\gamma/2}\ \rho^{m-1} + 4\sqrt{\eta}\ \kappa_2^2\ (\omega+1)^{3/4}\ \delta m^{\gamma/2}\ \rho^{m-1}} \\
&= \frac{\eta}{4\delta(\omega+1)^{3/2}\left(\kappa_2^2(1+\kappa_\infty)U + \sqrt{\eta}\ \kappa_2^2\ (\omega+1)^{-3/4}\right)}\ m^{-\gamma/2}\ \rho^{-m+1} \\
&\geq K m^{-\gamma/2}\rho^{-m+1}, \quad \text{where} \quad K := \frac{\eta}{8U\ \delta(\omega+1)^{3/2}\ \kappa_2^2(1+\kappa_\infty)}.
\end{aligned}$$

In the last inequality, we use the facts that $0 < \eta \leq 1$ and $(\omega+1)^{-3/4} \leq 1$, which implies that $\sqrt{\eta}(\omega+1)^{-3/4} \leq 1$. Moreover, we use the fact that $\kappa_2^2 \leq \kappa_2^2(1+\kappa_\infty)U$. To complete the proof, it suffices to establish that $Km^{-\gamma/2}\rho^{-m+1} \geq U$. To this end, we first note that

$$\begin{aligned}
Km^{-\gamma/2}\rho^{-m+1} \geq U &\iff \log\left(m^{-\gamma/2}\rho^{-m+1}\right) \geq \log\left(U/K\right) \\
&\iff -\log(m) + (m-1)\frac{2\log\left(1/\rho\right)}{\gamma} \geq \frac{2\log\left(U/K\right)}{\gamma} \\
&\iff -\log(m) + m\frac{2\log\left(1/\rho\right)}{\gamma} \geq \frac{2\log\left(U/K\right)}{\gamma} + \frac{2\log(1/\rho)}{\gamma}.
\end{aligned}$$

Let $\Gamma := \frac{\log(1/\rho)}{\gamma}$. By our assumption, $m \geq \max\{1, 2/\Gamma, (1/\Gamma)^2\}$. Therefore, Claim 1 implies that $\Gamma m \geq \log(m)$, which leads to

$$-\log(m) + m\frac{2\log(1/\rho)}{\gamma} \geq 2\frac{\log(U/K)}{\gamma} + 2\frac{\log(1/\rho)}{\gamma} \Longleftarrow m\frac{\log(1/\rho)}{\gamma} \geq 2\frac{\log(U/K)}{\gamma} + 2\frac{\log(1/\rho)}{\gamma}$$
$$\Longleftarrow m \geq \frac{2\log(U/K)}{\log(1/\rho)} + 2.$$

Given the definition of K, one can verify that the last inequality is implied by (16).

□

To leverage the above lemma and obtain a global bound on the maximum number of quadratic pieces retained for each local parametric cost after pruning, we introduce the notion of the (k, η)-margin. Throughout our subsequent arguments, we fix m to be any quantity satisfying (16). For any $u \in \{1, \ldots, n\}$ and any $\boldsymbol{\xi} \in \{0,1\}^{|\mathcal{V}_{u,m}|}$, define $\mathcal{C}_{u,\boldsymbol{\xi}} := \{\boldsymbol{s} \in \{0,1\}^{n_u} : \boldsymbol{s}_{\mathcal{V}_{u,m}} = \boldsymbol{\xi}\}$ as the set of all m-similar sparsity patterns whose entries within the m-neighborhood of bag $\mathcal{B}_u$ in $\mathrm{supp}_u(\boldsymbol{Q})$ coincide with $\boldsymbol{\xi}$. Equivalently, $\mathcal{C}_{u,\boldsymbol{\xi}}$ is the equivalence class induced by the relation $\boldsymbol{s} \sim \boldsymbol{s}'$ if and only if $\boldsymbol{s}_{\mathcal{V}_{u,m}} = \boldsymbol{s}'_{\mathcal{V}_{u,m}}$. Recall that $|\mathcal{V}_{u,m}| \leq \Delta_m$; hence, there are at most 2^{Δ_m} such equivalence classes.

Definition 5 (η-optimal sets and (k, η)-margin)**.** Given any $\eta \geq 0$, $u \in \{1, \ldots, n\}$, and $\boldsymbol{\xi} \in \{0,1\}^{|\mathcal{V}_{u,m}|}$, the η-optimal set of the equivalence class $\mathcal{C}_{u,\boldsymbol{\xi}}$ is defined as

$$\mathcal{R}_{u,\boldsymbol{\xi},\eta} := \left\{ \boldsymbol{s} \in \mathcal{C}_{u,\boldsymbol{\xi}} : d_{u,\boldsymbol{s}} - \min_{\boldsymbol{z} \in \mathcal{C}_{u,\boldsymbol{\xi}}} \{d_{u,\boldsymbol{z}}\} \leq \eta \right\}.$$

We say that the problem has (k, η)-margin if $|\mathcal{R}_{u,\boldsymbol{\xi},\eta}| \leq k$ holds for every $u \in \{1, \ldots, n\}$ and $\boldsymbol{\xi} \in \{0,1\}^{|\mathcal{V}_{u,m-1}|}$.

To build intuition behind the notions of η-optimal sets and (k, η)-margin, one can show that, from its definition (10), $d_{u,\boldsymbol{s}}$ coincides with the optimal value of the following quadratic program

$$d_{u,\boldsymbol{s}} = \min_{\boldsymbol{x} \in \mathbb{R}^{|\mathcal{J}_{u,\boldsymbol{s}}|}} \left\{ \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\boldsymbol{x} + \boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^\top \boldsymbol{x} + \sum_{i \in \mathcal{J}_{u,\boldsymbol{s}}} \boldsymbol{\lambda}_i \right\}.$$

This value is obtained from the subproblem (4) defined on $\mathrm{supp}_u(\boldsymbol{Q})$ after setting $\boldsymbol{x}_{\mathcal{B}_u} = \boldsymbol{0}$ and fixing the binary vector $\boldsymbol{z} \in \{0,1\}^{n_u}$ according to the sparsity pattern $\boldsymbol{s}$, namely $\boldsymbol{z}_i = 1$ for $i \in \mathcal{J}_{u,\boldsymbol{s}}$ and $\boldsymbol{z}_i = 0$ for $i \notin \mathcal{J}_{u,\boldsymbol{s}}$. Consequently, $\min_{\boldsymbol{z} \in \mathcal{C}_{u,\boldsymbol{\xi}}}\{d_{u,\boldsymbol{z}}\}$ represents the optimal value among all such restricted subproblems whose sparsity patterns belong to the equivalence class $\mathcal{C}_{u,\boldsymbol{\xi}}$. The associated η-optimal set $\mathcal{R}_{u,\boldsymbol{\xi},\eta}$ then consists of those sparsity patterns whose corresponding subproblem values lie within η of this minimum. In other words, $\mathcal{R}_{u,\boldsymbol{\xi},\eta}$ captures all binary assignments in the equivalence class $\mathcal{C}_{u,\boldsymbol{\xi}}$ that are nearly optimal (with an optimality gap of η) for the local subproblem at node u. For instance, if $|\mathcal{R}_{u,\boldsymbol{\xi},\eta}| = 1$, then the corresponding subproblem admits a unique optimal sparsity pattern from $\mathcal{C}_{u,\boldsymbol{\xi}}$ with an η-margin. This observation motivates the definition of the (k, η)-margin: the problem is said to have (k, η)-margin if, for every bag $\mathcal{B}_u$ and every equivalence class $\mathcal{C}_{u,\boldsymbol{\xi}}$, the associated η-optimal set has cardinality at most k; that is, there are at most k solutions whose corresponding optimal values lie within an η-margin of the minimum.

This assumption serves as a structural "margin" condition that controls the multiplicity of nearly indistinguishable discrete solutions. Conceptually, this is analogous to margin conditions in statistical learning—such as the classical Tsybakov low-noise assumption [67] and its refinements [29]—which limit the mass of points lying near the decision boundary. Our framework adapts this idea by imposing a bound on the size of the η-optimal set.

Indeed, the effectiveness of the proposed pruning strategy critically depends on the parameters of the (k, η)-margin. According to Lemma 8, the pruning strategy guarantees the identification of irrelevant pieces once they fall outside the η-margin. Consequently, within each equivalence class $\mathcal{C}_{u,\boldsymbol{\xi}}$, at most k equations remain relevant after pruning. Since there are at most $2^{|\mathcal{V}_{u,m}|}$ equivalence classes, the total number of equations retained after the pruning step is at most $k\ 2^{|\mathcal{V}_{u,m}|}$. This is formalized in the following lemma.

Lemma 9. *Suppose that Problem (1) has (k, η)-margin and that the polynomial growth condition in Assumption 1 holds. For every $u \in \{1, \dots, n-1\}$, let $\mathcal{P}_{u+1}$ denote the pruned index set obtained after the pruning step (Line 6 of Algorithm 2). Then, $\mathcal{P}_{u+1}$ satisfies $|\mathcal{P}_{u+1}| \leq k\ 2^{|\mathcal{V}_{u+1,m}|}$, where m is the smallest integer satisfying (16).*

Proof. Choose any $\boldsymbol{\xi} \in \{0,1\}^{|\mathcal{V}_{u+1,m}|}$ and consider the corresponding equivalence class $\mathcal{C}_{u+1,\boldsymbol{\xi}}$, whose elements are m-similar by definition. By Lemma 8, for any two sparsity patterns $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \mathcal{C}_{u+1,\boldsymbol{\xi}}$ with $|d_{u+1,\boldsymbol{s}^{(1)}} - d_{u+1,\boldsymbol{s}^{(2)}}| \geq \eta$, at least one of the two patterns is irrelevant and is therefore removed by the pruning step. Consequently, after pruning, the only sparsity patterns in $\mathcal{C}_{u+1,\boldsymbol{\xi}}$ retained are those whose constant term $d_{u+1,\boldsymbol{s}}$ lies within η of the minimum over the class, namely those in

$$\mathcal{R}_{u+1,\boldsymbol{\xi},\eta} = \Big\{ \boldsymbol{s} \in \mathcal{C}_{u+1,\boldsymbol{\xi}} : d_{u+1,\boldsymbol{s}} - \min_{\boldsymbol{z} \in \mathcal{C}_{u+1,\boldsymbol{\xi}}} d_{u+1,\boldsymbol{z}} \leq \eta \Big\}.$$

By the (k, η)-margin assumption, we have $|\mathcal{R}_{u+1,\boldsymbol{\xi},\eta}| \leq k$ for every $\boldsymbol{\xi}$, and thus at most k sparsity patterns remain in each equivalence class after pruning. Each $\boldsymbol{\xi} \in \{0,1\}^{|\mathcal{V}_{u+1,m}|}$ defines a unique equivalence class $\mathcal{C}_{u+1,\boldsymbol{\xi}}$, and hence there are $2^{|\mathcal{V}_{u+1,m}|}$ such classes in total. Summing over all classes yields $|\mathcal{P}_{u+1}| \leq k\ 2^{|\mathcal{V}_{u+1,m}|}$, which proves the statement. □

Equipped with the above lemma, we are now ready to provide our main theorem on the correctness and runtime of Algorithm 2.

Theorem 1. *Suppose that Problem (1) has (k, η)-margin and that the polynomial growth condition in Assumption 1 holds. Then, the proposed parametric algorithm (Algorithm 2) solves Problem (1) in $\mathcal{O}(n\omega^2\ \delta\ k^{2\delta}\ 4^{\Delta_{m+1}})$ time and $\mathcal{O}(n\omega^2\ k^{\delta}\ 2^{\Delta_{m+1}})$ memory, where*

$$m = \left\lceil \max\left\{ \frac{2\log\left(8U^2/\eta\right) + 2\log\left(\delta(\omega+1)^{3/2}\right) + 2\log\left(\kappa_2^2(1+\kappa_\infty)\right)}{\log(1/\rho)} + 2,\ \frac{2\gamma}{\log(1/\rho)},\ \left(\frac{\gamma}{\log(1/\rho)}\right)^2 \right\} \right\rceil.$$

Proof. We first prove the correctness of the algorithm, then analyze its time and memory complexities.

Correctness proof. Let $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ be an optimal solution of Problem (1), and let $f^\star$ be its objective value. To establish correctness, it suffices to show that $\boldsymbol{z}^\star_{\mathcal{J}_n}$ belongs to the pruned set $\mathcal{P}_n$ of the local parametric cost f_n. The proof proceeds by contradiction. Suppose that $\boldsymbol{z}^\star_{\mathcal{J}_n} \notin \mathcal{P}_n$. Then there exists an index $u \leq n$ at which the sparsity pattern of the optimal solution restricted to nodes

in $\mathcal{J}_u$, namely $\boldsymbol{z}^\star_{\mathcal{J}_u}$, is discarded by the pruning step. Let u denote the smallest index for which $\boldsymbol{z}^\star_{\mathcal{J}_u} \notin \mathcal{P}_u$. Since $\boldsymbol{z}^\star_{\mathcal{J}_u}$ is discarded by the pruning step, the corresponding piece $p_{u,\boldsymbol{z}^\star_{\mathcal{J}_u}}$ is irrelevant by definition of the pruning procedure. Therefore, there exists $\bar{\boldsymbol{s}} \in \mathcal{P}_u$ such that $p_{u,\boldsymbol{z}^\star_{\mathcal{J}_u}}(\boldsymbol{\alpha}) > p_{u,\bar{\boldsymbol{s}}}(\boldsymbol{\alpha})$ for all $\|\boldsymbol{\alpha}\|_\infty < U$.

To arrive at a contradiction, we evaluate the objective of Problem (1) at the optimal solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$. Before doing so, we partition the nodes of $\mathrm{supp}(\boldsymbol{Q})$ into three disjoint sets: $\mathcal{A} = \{1, \ldots, n\} \backslash (\mathcal{J}_u \cup \mathcal{B}_u)$, $\mathcal{B}_u$, and $\mathcal{J}_u$. From the properties of a tree decomposition, there is no edge between nodes in $\mathcal{A}$ and $\mathcal{J}_u$ in $\mathrm{supp}(\boldsymbol{Q})$; hence $\boldsymbol{Q}_{\mathcal{A},\mathcal{J}_u} = \boldsymbol{Q}^\top_{\mathcal{J}_u,\mathcal{A}} = \boldsymbol{0}$. With respect to this partition, $\boldsymbol{Q}$ and $\boldsymbol{c}$ take the block form:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{\mathcal{A},\mathcal{A}} & \boldsymbol{Q}_{\mathcal{A},\mathcal{B}_u} & \boldsymbol{0} \\ \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{A}} & \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u} & \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_u} \\ \boldsymbol{0} & \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{B}_u} & \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{J}_u} \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_{\mathcal{A}} \\ \boldsymbol{c}_{\mathcal{B}_u} \\ \boldsymbol{c}_{\mathcal{J}_u} \end{bmatrix}.$$

Evaluating the objective function of Problem (1) at the optimal solution yields:

$$\begin{aligned} f^\star &= \frac{1}{2}(\boldsymbol{x}^\star)^\top \boldsymbol{Q}\boldsymbol{x}^\star + \boldsymbol{c}^\top \boldsymbol{x}^\star + \boldsymbol{\lambda}^\top \boldsymbol{z}^\star \\ &= \frac{1}{2}(\boldsymbol{x}^\star_{\mathcal{A}})^\top \boldsymbol{Q}_{\mathcal{A},\mathcal{A}}\boldsymbol{x}^\star_{\mathcal{A}} + \boldsymbol{c}^\top_{\mathcal{A}}\boldsymbol{x}^\star_{\mathcal{A}} + \sum_{i\in\mathcal{A}} \lambda_i \boldsymbol{z}^\star_i + \boldsymbol{x}^\star_{\mathcal{A}} Q_{\mathcal{A},\mathcal{B}_u}\boldsymbol{x}^\star_{\mathcal{B}_u} + \sum_{i\in\mathcal{B}_u} \lambda_i \boldsymbol{z}^\star_i + p_{u,\boldsymbol{z}^\star_{\mathcal{J}_u}}(\boldsymbol{x}^\star_{\mathcal{B}_u}) \\ &> \frac{1}{2}(\boldsymbol{x}^\star_{\mathcal{A}})^\top \boldsymbol{Q}_{\mathcal{A},\mathcal{A}}\boldsymbol{x}^\star_{\mathcal{A}} + \boldsymbol{c}^\top_{\mathcal{A}}\boldsymbol{x}^\star_{\mathcal{A}} + \sum_{i\in\mathcal{A}} \lambda_i \boldsymbol{z}^\star_i + \boldsymbol{x}^\star_{\mathcal{A}} Q_{\mathcal{A},\mathcal{B}_u}\boldsymbol{x}^\star_{\mathcal{B}_u} + \sum_{i\in\mathcal{B}_u} \lambda_i \boldsymbol{z}^\star_i + p_{u,\bar{\boldsymbol{s}}}(\boldsymbol{x}^\star_{\mathcal{B}_u}). \end{aligned}$$

This shows that the vector $\bar{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{z}^\star_{\mathcal{A}} & \boldsymbol{z}^\star_{\mathcal{B}_u} & \bar{\boldsymbol{s}} \end{bmatrix}^\top$ is a strictly better choice for the binary variables, thereby contradicting the optimality of $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$.

Complexity proof. To analyze the runtime, we examine each step of the algorithm. The topological ordering and the subsequent labeling can be carried out in $\mathcal{O}(n)$ and $\mathcal{O}(n\omega^2)$ time, respectively, and in $\mathcal{O}(n\omega)$ memory (see Section 2.1). Initializing f_1 requires $\mathcal{O}(\omega^2)$ time and memory (Line 2).

Next, we analyze the complexity of the first `for` loop (Lines 3–7). For each $u = 1, \ldots, n-1$, the function g_u can be obtained by separately minimizing each piece of f_u with and without the indicator variable, with a total cost of $\mathcal{O}(|\mathcal{P}_u|\omega^2)$ time and memory. On the other hand, by Lemma 9, we have $|\mathcal{P}_u| \le k\ 2^{|\mathcal{V}_{u,m}|}$. This implies that the cost of computing g_u can be bounded by $\mathcal{O}(\omega^2\ k\ 2^{|\mathcal{V}_{u,m}|}) = \mathcal{O}(\omega^2\ k\ 2^{\Delta_m})$. Moreover, given $\{g_v\}_{v\in\mathrm{par}_T(u+1)}$, the function f_{u+1} in Line 5 can be computed according to (6), which we rewrite here:

$$f_{u+1}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}}) = h_{u+1}(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}}) + \sum_{v\in\mathrm{par}_T(u+1)} \Big(g_v(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}\backslash v}) - \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_{u+1}\backslash v}) \Big).$$

As established in the runtime analysis of Algorithm 1, h_{u+1} and $\{\phi_v : v \in \mathrm{par}_{\mathsf{T}}(u+1)\}$ are single-piece quadratic functions and can be computed in $\mathcal{O}(\omega^2)$ time and memory.

Next, we turn to the computation of g_v. Since the number of pieces of g_v is at most twice that of the corresponding f_v, each g_v contains at most $2k\ 2^{|\mathcal{V}_{v,m}|}$ pieces. Consequently, f_{u+1} can be constructed by considering all combinations of the pieces of $g_v - \phi_v$ for $v \in \mathrm{par}_T(u+1)$, leading to

at most

$$\prod_{v \in \mathrm{par}_T(u+1)} 2k\ 2^{|\mathcal{V}_{v,m}|} = k^{|\mathrm{par}_T(u+1)|}\ 2^{\sum_{v \in \mathrm{par}_T(u+1)}(|\mathcal{V}_{v,m}|+1)}$$

pieces. Indeed, we have $|\mathrm{par}_T(u+1)| \leq \Delta_1 \leq \delta$. Moreover, by definition of $\mathcal{V}_{u+1,m+1}$, we have $\mathcal{V}_{u+1,m+1} = \bigcup_{v \in \mathrm{par}_\mathsf{T}(u+1)}(\mathcal{V}_{v,m} \cup \{v\})$. Since the sets $\{\mathcal{V}_{v,m} \cup \{v\}\}_{v \in \mathrm{par}_\mathsf{T}(u+1)}$ are disjoint, it follows that $|\mathcal{V}_{u+1,m+1}| = \sum_{v \in \mathrm{par}_\mathsf{T}(u+1)}(|\mathcal{V}_{v,m}| + 1)$. This implies that f_{u+1}, before pruning, can have at most $k^\delta\ 2^{|\mathcal{V}_{u+1,m+1}|}$ pieces, and can be formed in $\mathcal{O}(\delta\ \omega^2\ k^\delta\ 2^{|\mathcal{V}_{u+1,m+1}|})$ time and $\mathcal{O}(\omega^2\ k^\delta\ 2^{|\mathcal{V}_{u+1,m+1}|})$ memory. Finally, Line 6 invokes the pruning subroutine `PRUNE`, which runs in $\mathcal{O}\left(\omega^2\ \left(k^\delta\ 2^{|\mathcal{V}_{u+1,m+1}|}\right)^2\right) = \mathcal{O}\left(\omega^2\ k^{2\delta}\ 4^{\Delta_{m+1}}\right)$ time and $\mathcal{O}\left(\omega^2\ k^\delta\ 2^{|\mathcal{V}_{u+1,m+1}|}\right) = \mathcal{O}\left(\omega^2\ k^\delta\ 2^{\Delta_{m+1}}\right)$ memory, as discussed in Section 5. Since the first `for` loop runs for $n-1$ iterations, it incurs a total cost of $\mathcal{O}(n\omega^2\ \delta\ k^{2\delta}\ 4^{\Delta_{m+1}})$ time and $\mathcal{O}(n\omega^2\ k^\delta\ 2^{\Delta_{m+1}})$ memory.

Finally, obtaining $f^\star$, $\boldsymbol{x}_n^\star$, and $\boldsymbol{z}_n^\star$ in Line 8, and each iteration of the second `for` loop (Lines 9–11), can be carried out in $\mathcal{O}(\omega^2\ 2^{\Delta_m})$ time and memory, and we omit the details for brevity. Combining all these steps, we conclude that the algorithm runs in $\mathcal{O}(n\omega^2\ \delta\ k^{2\delta}\ 4^{\Delta_{m+1}})$ time and $\mathcal{O}(n\omega^2\ k^\delta\ 2^{\Delta_{m+1}})$ memory. □

To provide further insight into Theorem 1, we focus on special classes of problems whose sparsity graph $\mathrm{supp}(\boldsymbol{Q})$ exhibits *linear volume growth*, i.e., Assumption 1 holds with $\gamma = 1$, as our complexity bound takes a more crisp form in this setting.

Corollary 1. *Suppose that Problem (1) has (k,η)-margin and satisfies linear volume growth (Assumption 1 with $\gamma = 1$). Then, the proposed parametric algorithm (Algorithm 2) solves Problem (1) in*

$$\mathcal{O}\left(n\omega^2\ \delta\ k^{2\delta}\ \max\left\{\left(\frac{U^2}{\eta}\ \delta(\omega+1)^{3/2}\ \kappa_2^2(1+\kappa_\infty)\right)^{\frac{4\delta}{\log(1/\rho)}},\ 4^{\delta\theta}\right\}\right),$$

time, where $\theta := \max\left\{\frac{2}{\log(1/\rho)},\ \frac{1}{\log^2(1/\rho)}\right\}$.

Proof. The result follows directly from Theorem 1 after setting $\gamma = 1$. □

The significance of the above corollary lies in the fact that, for fixed treewidth ω, margin parameters (k,η), condition numbers κ_2, κ_∞, and the volume growth parameters (δ,γ), the runtime scales *linearly* with n. In the next section, we examine the extent to which this theoretical dependence is reflected in practice. Here, we highlight this result in the context of two important special cases for which existing guarantees are available: trees with a bounded number of leaves and banded matrices with bandwidth w. The former has treewidth equal to 1, while the latter has treewidth at most w.

- **Tree-structured problems.** When $\mathrm{supp}(\boldsymbol{Q})$ is a tree, [14] shows that a specialized version of the parametric algorithm solves Problem (1) in $\mathcal{O}(n^2)$ time and memory. Although the theoretical bound is quadratic in n, the empirical performance reported therein is nearly linear. The above corollary partially explains this phenomenon: for trees with a bounded number of leaves, the theoretical complexity is in fact linear in n under the mild (k,η)-margin condition. This observation aligns the theoretical guarantees with empirical performance and clarifies why tree structures are particularly favorable for our method.

- **Banded matrices.** Suppose that $\boldsymbol{Q}$ is banded with bandwidth w. For the special case $w = 2$ (corresponding to the tridiagonal structure), [53] proved that Problem (1) can be solved to optimality in $\mathcal{O}(n^2)$ time. More recently, Gómez et al. [39] developed an FPTAS for Problem (1) with banded $\boldsymbol{Q}$ with bandwidth $w \geq 2$, computing ϵ-accurate solutions in time polynomial in n and in $\|\boldsymbol{c}\|_\infty/\epsilon$, provided both δ and κ_2 are fixed. Under the aforementioned margin assumption, our result yields an exact algorithm and reduces the dependence on n for both problem classes.

7 Numerical experiments

Next, we evaluate the performance of our algorithm across a range of synthetic and realistic instances. In Subsection 7.1, we describe our experimental setup and implementation details. In Subsection 7.2, we examine the dependence of the algorithm on various problem parameters using synthetic data. Finally, in Subsection 7.3, we demonstrate the effectiveness of our approach on the exponential smoothing problem with outlier correction using real-world data.

7.1 Methods and settings

All experiments were conducted on a computer with 8-core 3.0 GHz Xeon Gold 6154 processors and 16 GB of memory. We compare the performance of the proposed parametric algorithm against Gurobi v10.0.2. For all Gurobi runs, we impose a time limit of one hour and terminate the solver once the optimality gap falls below 0.01%. If Gurobi does not attain an optimality gap of 0.01% or smaller within this limit, we report the best optimality gap achieved by the solver. All results reported in the tables and figures are averaged across five independent trials. The Python implementation of our algorithm, along with the code used for the case studies presented in this paper, is available at:

https://github.com/aareshfb/Treewidth-Parametric-Algorithm.

- **Gurobi** We reformulate Problem (1) as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n,\, \boldsymbol{z}\in\{0,1\}^n} \quad \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top \boldsymbol{z}, \tag{17a}$$

$$\text{s.t.} \quad -U\boldsymbol{z}_i \leq \boldsymbol{x}_i \leq U\boldsymbol{z}_i, \qquad i = 1, 2, \ldots, n. \tag{17b}$$

  where U is the same upper bound also used in our parametric algorithm. We note that the problem also admits a perspective reformulation [34]. We experimented with explicitly incorporating this reformulation within Gurobi; however, we found that the solver's default formulation consistently outperformed our manually implemented perspective reformulation. We conjecture that this is because Gurobi internally exploits perspective-based strengthening more effectively. Consequently, all numerical results reported in this paper are based on Gurobi's built-in configuration.

- **Parametric method** We compare the performance of Gurobi with that of our proposed parametric algorithm (Algorithm 2). We note that the pruning subroutine (Algorithm 3) has a runtime of $\mathcal{O}(\omega^2 N^2)$, as it exhaustively compares all pairs of the N quadratic pieces that

constitute a local parametric cost. In the special case where the support graph of $\boldsymbol{Q}$ admits a tree decomposition that is a path (e.g., banded matrices), we can leverage an enhanced pruning procedure that processes the pieces in a single pass, requiring only $\mathcal{O}(N)$ comparisons. This improvement is possible when the quadratic functions are stored such that every consecutive pair is m-similar, which can be ensured at no additional cost when the tree decomposition has a path structure. The details of this implementation are deferred to Appendix B.

7.2 Results for synthetically generated instances

Recall that, according to Theorem 1, the runtime of the parametric algorithm scales linearly with the problem size, polynomially with the margin parameters (k, η), the volume growth parameter γ, and the conditioning parameters κ_2 and κ_∞, and exponentially with the volume growth parameter δ. In this subsection, we empirically study these dependencies across a wide range of synthetic instances.

The Hessian $\boldsymbol{Q}$ is constructed as $\boldsymbol{Q} = \boldsymbol{Y}^\top \boldsymbol{Y} + \nu I$, where $\boldsymbol{Y}$ is an upper triangular matrix with entries sampled uniformly from $[-1, 1]$ within the band w and zero otherwise (i.e., $\boldsymbol{Y}_{ij} \sim \mathcal{U}(-1, 1)$ for $i \leq j \leq i + w$ and zero otherwise). Simple calculation reveals that $\operatorname{supp}(\boldsymbol{Q})$ is banded with bandwidth w. The parameter $\nu > 0$ serves to regulate the condition number of $\boldsymbol{Q}$. The vector $\boldsymbol{c}$ is generated with independent entries uniformly distributed on $[-10, 10]$. Unless indicated otherwise, the regularization parameter $\boldsymbol{\lambda}$ is generated with independent entries uniformly distributed from the interval $(3.5, 4.5)$, which typically results in solutions with roughly 50% nonzero entries. In some experiments, we vary ν (to change the condition number) and $\boldsymbol{\lambda}$ (to change the sparsity level), while keeping the construction of $\boldsymbol{Q}$ and $\boldsymbol{c}$ fixed as described above.

Effect of problem size. In our first experiment, we examine the performance of the parametric algorithm on problems with varying n. Table 1 reports the runtime for bandwidths $w = 2$ and $w = 4$. For instances exceeding $n = 200$, Gurobi is unable to solve the instance to optimality within the time limit of one hour (indicated by TL). In contrast, the parametric algorithm can solve much larger instances within a few seconds. To verify correctness, we report the optimal objective values of both methods. It can be seen that the value obtained by the parametric algorithm is at least as good as that obtained by Gurobi (and in some cases strictly better). In addition, for the parametric algorithm, we report the average number of quadratic pieces retained after pruning. It can be seen that this number increases only modestly with n, showcasing the effectiveness of the proposed pruning approach in eliminating the irrelevant pieces.

We next evaluate the runtime of the parametric algorithm over a broader range of problem sizes and bandwidths. Figure 8 (left) reports performance for $w = 2, 3, 4, 5$ with n ranging up to 20,000. In these experiments, we tune ν such that $\kappa_2 \approx 6.50$. We observe that the parametric algorithm solves the largest instances with $w = 4$ in under one hour, whereas for $w = 5$, it solves instances with n up to 2,000 within one hour. The runtime curves exhibit nearly identical slopes across all values of w, indicating that the bandwidth influences the overall runtime scale but not its linear dependence on n, which is consistent with Corollary 1.

Effect of the bandwidth. We next isolate the effect of the bandwidth w on the runtime of the parametric algorithm. Figure 8 (right) reports the runtime as a function of w for problem sizes $n = 50, 200$ and condition number $\kappa_2 \approx 4.63$, with w ranging from 2 to 6. This log-linear plot shows that runtime scales exponentially with w, consistent with Corollary 1.

Table 1: Performance comparison for varying sizes

w	Metric	Method	$n = 100$	$n = 200$	$n = 500$	$n = 1000$	$n = 2000$
2	Condition no. (κ_2)	—	$\kappa_2 \approx 7.10$	$\kappa_2 \approx 7.43$	$\kappa_2 \approx 7.43$	$\kappa_2 \approx 8.11$	$\kappa_2 \approx 8.02$
	Time(s)	Parametric	0.17	0.35	0.91	1.82	3.70
		Gurobi	11.01	565.43	TL	TL	TL
	Objective value	Parametric	-588.89	-1143.57	-3033.48	-6363.05	-12367.60
		Gurobi	-588.89	-1143.57	-3033.28	-6362.32	-12365.20
	Avg no. eqs	Parametric	23	23	25	25	25
	B&B nodes	Gurobi	31326	2963376	4882224	2938436	1169022
	Opt. gap		0.00%	0.00%	1.11%	1.30%	1.50%
4	Condition no. (κ_2)	—	$\kappa_2 \approx 6.18$	$\kappa_2 \approx 6.20$	$\kappa_2 \approx 6.18$	$\kappa_2 \approx 6.43$	$\kappa_2 \approx 6.79$
	Time(s)	Parametric	7.82	17.11	42.45	83.88	176.98
		Gurobi	9.20	1634.54	TL	TL	TL
	Objective value	Parametric	-277.17	-547.57	-1383.69	-2886.91	-5837.25
		Gurobi	-277.17	-547.57	-1383.44	-2885.99	-5833.97
	Avg no. eqs	Parametric	995	1103	1082	1092	1139
	B&B nodes	Gurobi	13265	3162950	3669416	2583014	1017048
	Opt. gap		0.00%	0.41%	1.82%	2.32%	2.51%

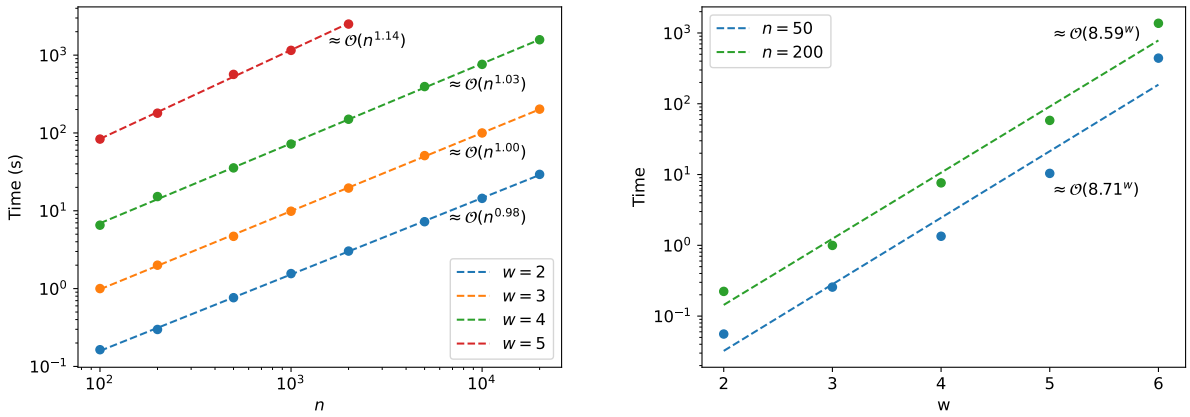

Figure 8: (Left) Runtime of the parametric algorithm as a function of n for fixed w, shown on a log-log scale. The empirical runtime scales linearly with n. (Right) Runtime as a function of w for fixed n, shown on a log-linear scale. The approximately linear trends indicate a dominant exponential dependence on w.

Effect of the condition number. For $n = 1{,}000$, we vary the condition number κ_2 while keeping the bandwidth w fixed. The regularization parameter $\boldsymbol{\lambda}$ is generated with independent entries uniformly distributed on $(2, 3)$ to maintain comparable sparsity levels in the optimal solution across instances. Figure 9 (left) reports the corresponding runtimes for $w = 2, 3, 4, 5$. According to Corollary 1, the runtime of the algorithm grows polynomially with κ_2, with the exponent of this growth increasing as the bandwidth w increases. This empirical observation is fully consistent with our theoretical result. For the small bandwidth of $w = 2$, the runtime scales as $\mathcal{O}(\kappa_2^{1.31})$, whereas larger bandwidths yield substantially steeper growth; for instance, $w = 5$ exhibits a scaling of approximately $\mathcal{O}(\kappa_2^{4.20})$.

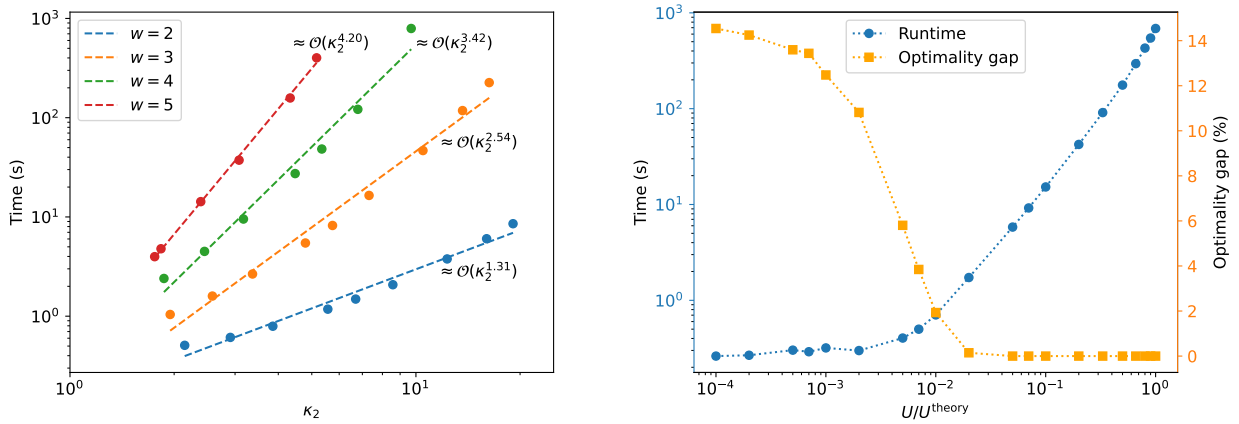

Figure 9: (Left) Runtime of the parametric algorithm as a function of κ_2 for fixed w, shown on a log–log scale. The empirical runtime scales polynomially with κ_2. (Right) Runtime (left axis) and optimality gap (right axis) as functions of the normalized parameter U/U^{theory}, shown on a semi-logarithmic scale. Reducing U reduces runtime at the cost of a larger optimality gap.

Effect of the solution upper bound. In our next experiment, we study how the choice of U affects the performance of the parametric algorithm. Let U^{theory} denote the value of U prescribed by Lemma 3. For this choice, the optimal solution satisfies $\|\boldsymbol{x}^\star\|_\infty \leq U^{\text{theory}}$, and therefore the parametric algorithm is guaranteed to recover the exact optimal solution. When $U < U^{\text{theory}}$, the pruning subroutine becomes more aggressive, which reduces the runtime of the algorithm. However, this runtime improvement may come at the cost of eliminating potentially optimal sparsity patterns, thereby leading to a nonzero optimality gap.

Figure 9 (right) reports the runtime (blue circles, left y-axis) and the optimality gap (orange squares, right y-axis) as functions of the normalized parameter U/U^{theory}. For this experiment, we fix $n = 1{,}000$, $w = 4$ and $\kappa_2 \approx 9.69$. As expected, when $U/U^{\text{theory}} = 1$, the algorithm recovers the exact optimal solution. As U/U^{theory} decreases below one, the runtime decreases. Interestingly, for $U/U^{\text{theory}} \geq 0.05$, the algorithm consistently recovers the optimal solution, while the runtime drops dramatically from 600 seconds to just 6 seconds. This suggests that the theoretical bound on U may be conservative in practice.

Effect of the sparsity parameter. The goal of the next experiment is to evaluate how the choice of the sparsity parameter $\boldsymbol{\lambda}$ affects the performance of the parametric algorithm. We fix the problem size at $n = 1{,}000$, the bandwidth at $w = 4$, and the condition number at approximately $\kappa_2 \approx 6.56$. For each trial, every coordinate λ_i is set to the same constant value, denoted by $\bar{\lambda}$. The results are reported in Table 2. As the optimal solutions become denser, the runtime of the parametric algorithm increases modestly, reaching 135.41 seconds when the solution has 93.7% nonzero (NZ) entries. By contrast, GUROBI is only able to return an optimal solution in this single dense case. In all other settings, GUROBI times out, and the reported optimality gaps increase sharply with $\bar{\lambda}$. For extremely sparse solutions (i.e., with only 0.2% nonzeros), GUROBI reports an optimality gap of $+\infty$.

Table 2: Performance comparison for varying regularization

Metric	**Method**	$\bar{\lambda} = 0.07$ NZ $\approx 93.7\%$	$\bar{\lambda} = 1$ NZ $\approx 75.2\%$	$\bar{\lambda} = 5$ NZ $\approx 50.8\%$	$\bar{\lambda} = 7$ NZ $\approx 35.0\%$	$\bar{\lambda} = 14$ NZ $\approx 7.38\%$	$\bar{\lambda} = 20$ NZ $\approx 0.2\%$
Time(s)	Parametric	135.41	119.85	86.71	67.44	36.23	21.97
	Gurobi	10.04	TL	TL	TL	TL	TL
B&B nodes	Gurobi	1	1780599	2549880	2840523	4025755	4416035
Opt. gap		0.00%	0.15%	2.26%	9.63%	210.25%	∞

NZ refers to the percentage of non-zero elements in the optimal solution $\boldsymbol{x}^\star$.

This behavior highlights a key strength of the parametric algorithm. In applications such as ESOC, typically only a small fraction of observed values are corrupted with outlier noise. In precisely these regimes, where GUROBI returns solutions with large optimality gaps, the parametric algorithm recovers the optimum with much smaller runtime.

Beyond banded structures. Next, we show that structural properties implied by small treewidth can be exploited beyond simple banded structures. To this end, we compare the performance of two variants of our algorithm: one that fully exploits the low-treewidth structure of the problem, and another that leverages only the banded structure. We construct instances for $\boldsymbol{Q}$ where the treewidth is fixed at $\omega = 2$, while the bandwidth varies over $w = \{3, 4, 5\}$. One such graph is illustrated in Figure 10. To generate a matrix $\boldsymbol{Q}$ with the desired properties, we first form a symmetric positive definite matrix with the specified bandwidth, following the procedure outlined in the previous section. We then selectively zero out certain off-diagonal entries to reduce the treewidth to $\omega = 2$, while preserving the original banded structure.

Table 3 reports the performance of our methods for a fixed treewidth $\omega = 2$ and bandwidths $w = 3, 4, 5$. All experiments are conducted with problem size $n = 1{,}000$, and the corresponding condition numbers κ_2 are reported in the table. The row labeled "Parametric" corresponds to the parametric algorithm applied to a tree decomposition of width $\omega = 2$. The row labeled "Banded" corresponds to the same parametric algorithm applied under the assumption of a banded structure only; in this case, the width of the induced tree decomposition equals $w > 2$. As shown in the table, exploiting the small treewidth of the graph—beyond merely its banded structure—substantially reduces both the runtime and the average number of quadratic pieces. In contrast, GUROBI is unable to solve any of these instances within the one-hour time limit.

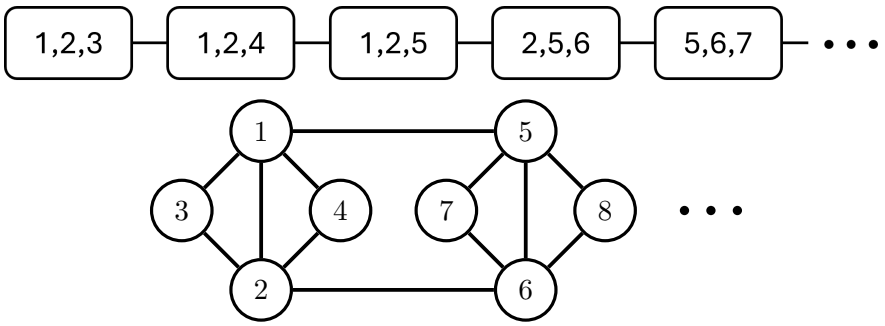

Figure 10: Tree decomposition of a graph derived from a banded matrix with bandwidth $w = 4$ and treewidth $\omega = 2$. The treewidth is reduced by eliminating edges while preserving the original bandwidth.

Table 3: Performance comparison for varying bandwidth

Metric	**Method**	$\boldsymbol{w = 3}$	$\boldsymbol{w = 4}$	$\boldsymbol{w = 5}$
Condition no. (κ_2)	—	8.71	9.25	8.12
Time(s)	Parametric	8.60	48.58	167.83
	Banded	8.52	64.36	658.24
	GUROBI	TL	TL	TL
Avg no. eqs	Parametric	81	442	1477
	Banded	123	848	3385
B&B nodes	Gurobi	3193465	3594783	3552258
Opt. gap		10.02%	14.02%	12.67%

7.3 Results for real-world instances

Next, we apply the proposed parametric algorithm to solve the problem of exponential smoothing with outlier correction (ESOC), introduced in Section 3.2. In addition to its runtime, we focus on the forecasting accuracy of this model relative to simple exponential smoothing (SES).

Datasets Our experiments use four real-world time series from the Numenta Anomaly Benchmark (NAB) [1]. NAB is a dataset designed to evaluate anomaly detection algorithms on streaming data. It includes real-world time-series data from diverse domains, such as cloud infrastructure metrics and traffic data. For detailed descriptions of the dataset, evaluation protocols, and related methods, see [51, 2]. The specific datasets used in our experiments are:

1. `ec2_cpu_utilization_53ea38.csv`; referred to as `CPU-1`,
2. `ec2_cpu_utilization_ac20cd.csv`; referred to as `CPU-2`,
3. `rds_cpu_utilization_e47b3b.csv`; referred to as `CPU-3`,
4. `speed_7578.csv`; referred to as `Traffic`.

Below, we briefly explain these datasets.

CPU utilization data The first three datasets consist of CPU utilization percentages collected from Amazon Web Services (AWS) servers. In such datasets, anomalies may stem from sudden workload surges or Distributed Denial-of-Service (DDoS) attacks [32]. The first two datasets correspond to general-purpose compute instances (Amazon EC2), while the third is obtained from a database server (Amazon RDS). Each dataset contains 2,000 CPU utilization measurements recorded at 5-minute intervals. As shown in the first three rows of Figure 11, these time series exhibit distinct temporal patterns, providing a diverse range of behaviors for evaluating the accuracy of the ESOC model.

Traffic data The final dataset consists of average traffic speed measurements from the Twin Cities Metro area in Minnesota and contains 1,127 observations recorded at irregular time intervals. Accurate short-term traffic speed forecasting is critical for anticipating changing road conditions and enabling effective traffic management [58]. In this context, anomalies may signal a major crash causing traffic obstruction [76, 68]. This dataset is shown in the last row of Figure 11 and is included to evaluate the proposed method on data exhibiting sampling irregularity and noise characteristics that differ substantially from those of the CPU utilization signals.

Experimental setup Let $\boldsymbol{y} \in \mathbb{R}^T$ denote a time series of length T, where $\boldsymbol{y}_t$ represents the observed value at time t. Our goal is to produce one-step-ahead point forecasts at each time t. We denote the resulting forecasts by $\hat{\boldsymbol{y}}_t^{\text{SES}}$ for SES, and by $\hat{\boldsymbol{y}}_t^{\text{ESOC}}$ for ESOC. For both models, the one-step-ahead forecast at time $t+1$ is obtained from the smoothed value at the previous time t [46]. Specifically, for SES, the forecast is given by $\hat{\boldsymbol{y}}_{t+1}^{\text{SES}} := \boldsymbol{x}_t^{\text{SES}} = \beta \boldsymbol{y}_t + (1-\beta)\boldsymbol{x}_{t-1}^{\text{SES}}$, where $\boldsymbol{x}_t^{\text{SES}}$ denotes the exponentially smoothed signal. For ESOC, the forecast is defined as $\hat{\boldsymbol{y}}_{t+1}^{\text{ESOC}} := \boldsymbol{x}_t^{\text{ESOC}}$, where $\boldsymbol{x}^{\text{ESOC}} \in \mathbb{R}^T$ is the smoothed signal obtained by solving ESOC. To evaluate the forecasting accuracy of SES, we use the mean squared error (MSE):

$$\text{MSE}^{\text{SES}} = \frac{1}{T-1} \sum_{t=2}^{T} \big(\hat{\boldsymbol{y}}_t^{\text{SES}} - \boldsymbol{y}_t\big)^2. \tag{18}$$

Unlike the model in (SES), the model in (ESOC) is capable of identifying and discounting outliers. Accordingly, we evaluate its forecasting accuracy only over the outlier-free region. Specifically, we define

$$\text{MSE}^{\text{ESOC}} = \frac{1}{T - 1 - \sum_{t=2}^{T} \mathbb{I}(\boldsymbol{o}_t)} \sum_{t=2}^{T} \big(1 - \mathbb{I}(\boldsymbol{o}_t)\big)\big(\hat{\boldsymbol{y}}_t^{\text{ESOC}} - \boldsymbol{y}_t\big)^2, \tag{19}$$

where $\mathbb{I}(\boldsymbol{o}_t)$ is an indicator that equals one if $\boldsymbol{y}_t$ is identified as an outlier and zero otherwise.

The parameters of both models are selected via grid search over a predefined set of candidate values. For parameter tuning, we split each time series into a training set consisting of the first half of the observations, $\{\boldsymbol{y}_t\}_{t=1}^{\lfloor T/2 \rfloor}$, and a test set consisting of the remaining observations, $\{\boldsymbol{y}_t\}_{t=\lfloor T/2 \rfloor + 1}^{T}$. For each method and parameter configuration, we compute the MSE on the training set and select the parameters that minimize this quantity. Using the selected parameters, we run each algorithm on the entire signal and compare the performance by computing MSE on the test set, according to (18) for (SES) and (19) for (ESOC).

For the model in (ESOC), we perform a grid search over $\beta \in \{0.01, 0.1, 0.2, \ldots, 0.9, 0.99\}$ and $\boldsymbol{\lambda}_t = \lambda, t = 1, \ldots, T$, where $\lambda \in \{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}\}$,

while fixing $\mu_1 = 1.2$ and $\mu_2 = 0.001$. To avoid degenerate solutions, we restrict the grid search to parameter settings that classify fewer than 10% of the observations as anomalies. Since the model in (SES) has a single tunable parameter β, we select β using the same candidate set as above.

Results Table 4 reports both the training and test MSE values of the smoothed signals, using the parameters obtained over the training set. For (ESOC), we also report the percentage of detected outliers. In all cases, (ESOC) achieves lower MSE than (SES), demonstrating its ability to suppress anomalies and produce more accurate forecasts. Notably, the largest improvements in test MSE are observed for the `traffic` and `CPU-3` signals. We note that the fraction of detected outliers is constrained to be below 10% only during the training phase. The results in this table are obtained by solving the optimization problem over the entire signal (training and test) using the parameters learned from training. Consequently, for the `traffic` signal, the observed outlier proportion exceeds this threshold.

Table 4: Comparison of MSE for SES and ESOC.

Signal	**Segment**	**MSE**		**Outliers detected**
		SES	**ESOC**	
`CPU-1`	Train	0.0101	0.0063	3.2%
	Test	0.0106	0.0068	2.8%
`CPU-2`	Train	9.1930	3.0941	2.9%
	Test	5.3983	3.1840	3.0%
`CPU-3`	Train	6.4085	0.1404	6.0%
	Test	0.8021	0.1649	16.6%
`Traffic`	Train	20.2650	6.7490	11.5%
	Test	65.5931	6.0920	20.7%

Table 5 compares the runtime of Gurobi and the proposed parametric algorithm for solving (ESOC). For all instances, Gurobi reaches the one-hour time limit; we therefore report only the optimality gaps at termination. In contrast, the parametric algorithm returns provably optimal solutions for all instances. Most datasets are solved within approximately 30 seconds, with one notably larger instance (`CPU-3`) requiring about 958 seconds. Overall, these results demonstrate that the parametric algorithm consistently attains optimal solutions and does so far more efficiently than Gurobi.

Table 5: Performance comparison between the parametric algorithm and Gurobi for solving ESOC.

Metric	**Method**	`CPU-1`	`CPU-2`	`CPU-3`	`Traffic`
Signal length (T)	—	2000	2000	2000	1127
Time(s)	Parametric	21.24	24.94	958.93	27.10
	Gurobi	TL	TL	TL	TL
B&B nodes	Gurobi	2077648	3909773	1609698	7360291
Opt. gap		78.88%	2.04%	1.06%	1.94%

Finally, Figure 11 visualizes the recovered signals obtained using the models in (SES) and (ESOC) across all four datasets.

Each row corresponds to a single dataset, with the output of SES shown in the left column and that of ESOC shown in the right column. In all panels, the observed signal is plotted together with

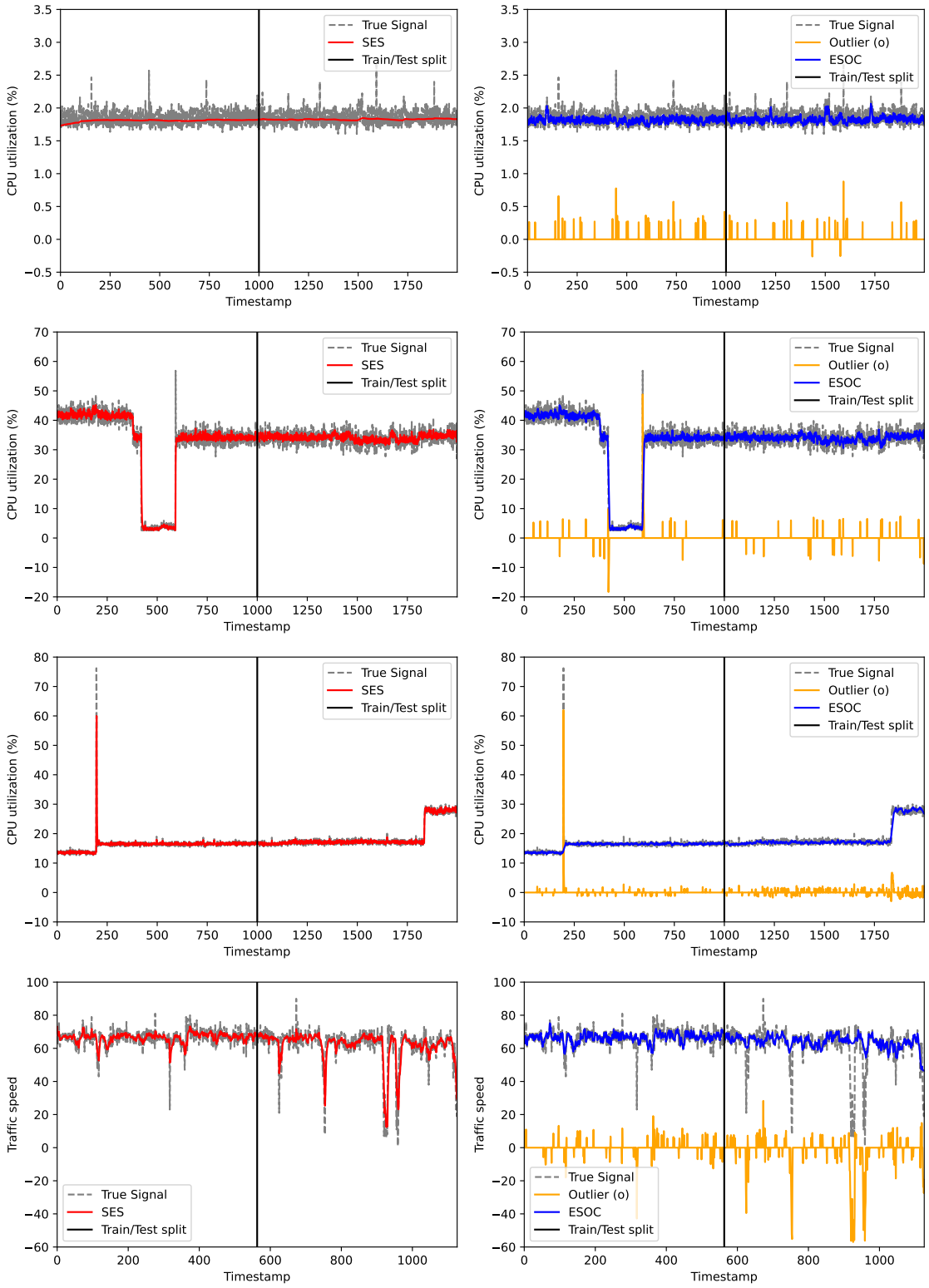

Figure 11: Comparison of SES (left column) and ESOC (right column) on four representative real-world signals. Each row corresponds to a different signal instance, with the observed signal shown alongside the smoothed estimates.

the corresponding smoothed estimate, and for ESOC, the estimated outlier vector $\boldsymbol{o}$ is highlighted in orange.

For the first dataset (`CPU-1`), the trend in the training set is representative of the full series. Under SES, the forecast is relatively smooth due to the small value of β selected on the training data, but exhibits a noticeable lag in the early portion of the series, where the recursion is dominated by the initial observation $\boldsymbol{y}_1$ (see Section 3.1). In contrast, the forecast from ESOC adapts more quickly, tracks the underlying signal more closely, and, by explicitly modeling outliers, shows reduced sensitivity to the initial value $\boldsymbol{y}_1$.

For the second dataset (`CPU-2`), the training and test segments exhibit visibly different behaviors. The training portion contains abrupt changes (between $t = 420$ and $t = 600$) followed by a large outlier, both of which are absent in the test set (the second half of the signal). While both SES and ESOC attenuate this outlier, ESOC produces a smoother estimate of the underlying trend. As shown in Table 4, this leads to a substantially lower test MSE for ESOC.

For the third dataset (`CPU-3`), the training set contains a large outlier followed by a single jump (around $t \approx 240$), while the test set exhibits an additional jump. Indeed, SES adapts to the jumps but incorporates the large outlier into its forecast. In contrast, ESOC successfully isolates and removes the outlier, accurately tracks the jumps, and yields a smoother estimate of the underlying trend.

Finally, for the fourth dataset (`Traffic`), the signal is less smooth and exhibits frequent short-term fluctuations across both the training and test sets. Toward the end of the time horizon, two large spikes occur. While SES retains both small and large fluctuations, including the large spikes, ESOC captures the underlying trend and smaller variations while effectively discarding the large spikes.

8 Conclusion

In summary, this paper demonstrates that convex quadratic optimization problems with indicator variables can be solved efficiently by exploiting graph-structured sparsity and a margin condition. By leveraging these properties, the proposed parametric algorithm achieves polynomial time and memory complexity—and, in certain regimes, even linear time and memory complexity—under suitable assumptions, substantially outperforming general-purpose mixed-integer solvers on large-scale instances. Beyond its theoretical guarantees, the framework offers a principled extension of exponential smoothing for joint forecasting and outlier detection, and exhibits strong empirical performance on challenging real-world time-series datasets.

9 Code and Data Disclosure

The code and data to support the numerical experiments in this paper can be found at

https://github.com/aareshfb/Treewidth-Parametric-Algorithm.

Note that the implementation provided is restricted to instances where the tree decomposition is a path. The Numenta Anomaly Benchmark (NAB) [1] used in the experiments containing real-world data can be accessed at

https://github.com/numenta/NAB/tree/master/data.

Acknowledgments

This research is supported, in part, by NSF grants 2152776, 2337776, ONR grants N00014-26-1-2074, N00014-22-1-2602, N00014-26-12117, and AFOSR grant FA9550-24-1-0086.

References

[1] Subutai Ahmad and Alexander Lavin. The numenta anomaly benchmark. https://github.com/numenta/NAB, 2015.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. Neurocomputing, 262:134–147, 2017.

[3] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network Flows. Prentice Hall, 1988.

[4] M Selim Aktürk, Alper Atamtürk, and Sinan Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Operations Research Letters, 37(3):187–191, 2009.

[5] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Applied Mathematics, 23(1):11–24, 1989.

[6] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods, 8(2):277–284, 1987.

[7] Alper Atamtürk and Andrés Gómez. Strong formulations for quadratic optimization with M-matrices and indicator variables. Mathematical Programming, 170(1):141–176, 2018.

[8] Alper Atamtürk, Andrés Gómez, and Shaoning Han. Sparse and smooth signal estimation: Convexification of ℓ_0-formulations. Journal of Machine Learning Research, 22(52):1–43, 2021.

[9] Marshall W. Bern, Eugene L. Lawler, and Alice L Wong. Linear-time computation of optimal subgraphs of decomposable graphs. Journal of Algorithms, 8(2):216–235, 1987.

[10] Dimitri Bertsekas. Dynamic Programming and Optimal Control: Volume I, volume 4. Athena scientific, 2012.

[11] Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression. The Annals of Statistics, 48(1):300–323, 2020.

[12] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. The Annals of Statistics, 44(2):813 – 852, 2016.

[13] Dimitris Bertsimas, Vassilis Digalakis Jr, Michael Lingzhi Li, and Omar Skali Lami. Slowly varying regression under sparsity. Operations Research, 73(3):1581–1597, 2025.

[14] Aaresh Bhathena, Salar Fattahi, Andrés Gómez, and Simge Küçükyavuz. A parametric approach for solving convex quadratic optimization with indicators over trees. Mathematical Programming, pages 1–46, 2025.

[15] Daniel Bienstock and Tongtong Chen. Solving convex QPs with structured sparsity under indicator conditions. arXiv preprint arXiv:2411.11722, 2024.

[16] Daniel Bienstock and Gonzalo Munoz. LP formulations for polynomial optimization problems. SIAM Journal on Optimization, 28(2):1121–1150, 2018.

[17] Baki Billah, Maxwell L King, Ralph D Snyder, and Anne B Koehler. Exponential smoothing model selection for forecasting. International Journal of Forecasting, 22(2):239–247, 2006.

[18] Hans L Bodlaender. Dynamic programming on graphs with bounded treewidth. In Automata, Languages and Programming: 15th International Colloquium Tampere, Finland, July 11–15, 1988 Proceedings 15, pages 105–118. Springer, 1988.

[19] Hans L Bodlaender. A Tourist Guide Through Treewidth, volume 92. Unknown Publisher, 1992.

[20] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing, 25(6):1305–1317, 1996.

[21] Hans L Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. Journal of Algorithms, 21(2):358–402, 1996.

[22] Sebastián Ceria and João Soares. Convex programming for disjunctive convex optimization. Mathematical Programming, 86(3):595–614, 1999.

[23] Charalambos A Charalambides. Enumerative Combinatorics. Chapman and Hall/CRC, 2018.

[24] Xiaojun Chen, Dongdong Ge, Zizhuo Wang, and Yinyu Ye. Complexity of unconstrained minimization. Mathematical Programming, 143(1-2):371–383, 2014.

[25] Abhimanyu Das and David Kempe. Algorithms for subset selection in linear regression. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pages 45–54, 2008.

[26] Antoine Dedieu, Hussein Hazimeh, and Rahul Mazumder. Learning sparse classifiers: Continuous and mixed integer optimization perspectives. Journal of Machine Learning Research, 22 (135):1–47, 2021.

[27] Alberto Del Pia, Santanu S Dey, and Robert Weismantel. Subset selection in sparse matrices. SIAM Journal on Optimization, 30(2):1173–1190, 2020.

[28] Stephen Demko, William F Moss, and Philip W Smith. Decay rates for inverses of band matrices. Mathematics of Computation, 43(168):491–499, 1984.

[29] Ilias Diakonikolas and Nikos Zarifis. A near-optimal algorithm for learning margin halfspaces with massart noise. Advances in Neural Information Processing Systems, 37:88605–88626, 2024.

[30] Reinhard Diestel. Graph Theory, volume 173. Springer Nature, 2025.

[31] Farzam Ebrahimnejad and James R Lee. On planar graphs of uniform polynomial growth. Probability Theory and Related Fields, 180(3):955–984, 2021.

[32] Mahmoud Said El Sayed, Nhien-An Le-Khac, Marianne A Azer, and Anca D Jurcut. A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns. IEEE Transactions on Cognitive Communications and Networking, 8(4):1862–1880, 2022.

[33] Salar Fattahi and Andres Gómez. Scalable inference of sparsely-changing Gaussian Markov random fields. Advances in Neural Information Processing Systems, 34:6529–6541, 2021.

[34] Antonio Frangioni and Claudio Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. Mathematical programming, 106(2):225–236, 2006.

[35] Antonio Frangioni and Claudio Gentile. A computational comparison of reformulations of the perspective relaxation: SOCP vs. cutting planes. Operations Research Letters, 37(3):206–210, 2009.

[36] Everette S Gardner Jr. Exponential smoothing: The state of the art—Part II. International Journal of Forecasting, 22(4):637–666, 2006.

[37] Fred Glover. Improved linear integer programming formulations of nonlinear integer problems. Management Science, 22(4):455–460, 1975.

[38] Andrés Gómez and Weijun Xie. A note on quadratic constraints with indicator variables: Convex hull description and perspective relaxation. Operations Research Letters, 52:107059, 2024.

[39] Andres Gómez, Shaoning Han, and Leonardo Lozano. Real-time solution of quadratic optimization problems with banded matrices and indicator variables. arXiv preprint arXiv:2405.03051, 2024.

[40] Oktay Günlük and Jeff Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. Mathematical Programming, 124(1):183–205, 2010.

[41] Rudolf Halin. S-functions for graphs. Journal of Geometry, 8(1-2):171–186, 1976.

[42] Shaoning Han and Andrés Gómez. Compact extended formulations for low-rank functions with indicator variables. Mathematics of Operations Research, 50(3):1992–2016, 2025.

[43] Shaoning Han, Andrés Gómez, and Jong-Shi Pang. On polynomial-time solvability of combinatorial Markov random fields. arXiv preprint arXiv:2209.13161, 2022.

[44] Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. Mathematical Programming, 196(1):347–388, 2022.

[45] Pinar Heggernes. Minimal triangulations of graphs: A survey. Discrete Mathematics, 306(3): 297–317, 2006.

[46] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. Forecasting with exponential smoothing: the state space approach. Springer Science & Business Media, 2008.

[47] Esma Kahraman and Ozlem Akay. Comparison of exponential smoothing methods in forecasting global prices of main metals. Mineral Economics, 36(3):427–435, 2023.

[48] Masakazu Kojima, Sunyoung Kim, and Hayato Waki. Sparsity in sums of squares of polynomials. Mathematical Programming, 103(1):45–62, 2005.

[49] George Kontogeorgiou and Martin Winter. (Random) trees of intermediate uniform growth. arXiv preprint arXiv:2212.01883, 2022.

[50] Simge Küçükyavuz, Ali Shojaie, Hasan Manzour, Linchuan Wei, and Hao-Hsiang Wu. Consistent second-order conic integer programming for learning Bayesian networks. Journal of Machine Learning Research, 24(322):1–38, 2023.

[51] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In 2015 IEEE 14th international conference on machine learning and applications (ICMLA), pages 38–44. IEEE, 2015.

[52] Jisun Lee, Andrés Gómez, and Alper Atamtürk. Convexification of multi-period quadratic programs with indicators. arXiv preprint arXiv:2412.17178, 2024.

[53] Peijing Liu, Salar Fattahi, Andrés Gómez, and Simge Küçükyavuz. A graph-based decomposition method for convex quadratic optimization with indicators. Mathematical Programming, 200 (2):669–701, 2023.

[54] Peijing Liu, Alper Atamtürk, Andrés Gómez, and Simge Küçükyavuz. Polyhedral analysis of quadratic optimization problems with Stieltjes matrices and indicators. To appear in Mathematical Programming, pages 1–27, 2025.

[55] Ramtin Madani, Somayeh Sojoudi, Ghazal Fazelnia, and Javad Lavaei. Finding low-rank solutions of sparse linear matrix inequalities using convex optimization. SIAM Journal on Optimization, 27(2):725–758, 2017.

[56] Hasan Manzour, Simge Küçükyavuz, Hao-Hsiang Wu, and Ali Shojaie. Integer programming for learning directed acyclic graphs from continuous data. INFORMS Journal on Optimization, 3(1):46–73, 2021.

[57] Günter Meinardus. Approximation of functions: Theory and numerical methods, volume 13. Springer Science & Business Media, 2012.

[58] Yi Ouyang, Richard Y Zhang, Javad Lavaei, and Pravin Varaiya. Large-scale traffic signal offset optimization. IEEE Transactions on Control of Network Systems, 7(3):1176–1187, 2020.

[59] S Ratnasari, R Zakaria, et al. Demand forecasting with five parameter exponential smoothing. In IOP Conference Series: Materials Science and Engineering, volume 495, page 012014. IOP Publishing, 2019.

[60] Visweswaran Ravikumar, Aaresh Bhathena, Wajd N Al-Holou, Salar Fattahi, and Arvind Rao. Efficient inference of dynamic gene regulatory networks using discrete penalty. arXiv preprint arXiv:2507.23106, 2025.

[61] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. Journal of algorithms, 7(3):309–322, 1986.

[62] Walter Rudin. Functional Analysis. McGraw–Hill, 1973.

[63] Soroosh Shafiee and Fatma Kılınç-Karzan. Constrained optimization of rank-one functions with indicator variables: S. shafiee, f. kılınç-karzan. Mathematical Programming, 208(1):533–579, 2024.

[64] Robert Alan Stubbs. Branch-and-cut methods for mixed 0-1 convex programming. PhD thesis, Northwestern University, 1996.

[65] Bilal Taghezouit, Fouzi Harrou, Ying Sun, Amar Hadj Arab, and Cherif Larbes. A simple and effective detection strategy using double exponential scheme for photovoltaic systems monitoring. Solar Energy, 214:337–354, 2021.

[66] Quang Thanh Tran, Li Hao, and Quang Khai Trinh. A comprehensive research on exponential smoothing methods in modeling and forecasting cellular traffic. Concurrency and Computation: Practice and Experience, 32(23):e5602, 2020.

[67] Alexander B Tsybakov. Optimal aggregation of classifiers in statistical learning. The Annals of Statistics, 32(1):135–166, 2004.

[68] Franco Van Wyk, Yiyang Wang, Anahita Khojandi, and Neda Masoud. Real-time sensor anomaly detection and identification in automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 21(3):1264–1276, 2019.

[69] Martin J Wainwright and Michael I Jordan. Treewidth-based conditions for exactness of the sherali-adams and lasserre relaxations. Technical report, Technical Report 671, University of California, Berkeley, 2004.

[70] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM Journal on Optimization, 17(1):218–242, 2006.

[71] Linchuan Wei, Andrés Gómez, and Simge Küçükyavuz. Ideal formulations for constrained convex optimization problems with indicator variables. Mathematical Programming, 192(1): 57–88, 2022.

[72] Linchuan Wei, Alper Atamtürk, Andrés Gómez, and Simge Küçükyavuz. On the convex hull of convex quadratic optimization problems with indicators. Mathematical Programming, 204(1): 703–737, 2024.

[73] Weijun Xie and Xinwei Deng. Scalable algorithms for the sparse ridge regression. SIAM Journal on Optimization, 30(4):3359–3386, 2020.

[74] Tong Xu, Simge Küçükyavuz, Ali Shojaie, and Armeen Taeb. An asymptotically optimal coordinate descent algorithm for learning Bayesian networks from Gaussian models. To appear in Journal of Machine Learning Research, 2025. URL https://arxiv.org/abs/2408.11977.

[75] Tong Xu, Armeen Taeb, Simge Küçükyavuz, and Ali Shojaie. Integer programming for learning directed acyclic graphs from non-identifiable Gaussian models. Biometrika, 112(3):asaf032, 04 2025.

[76] Chao Zhao, Xiaokun Chang, Tian Xie, Hamido Fujita, and Jian Wu. Unsupervised anomaly detection based method of risk evaluation for road traffic accident. Applied Intelligence, 53(1): 369–384, 2023.

A Omitted proofs

A.1 Proof of Lemma 1

Recall that $\mathcal{J}_u$ was defined as the set of nodes in $\mathrm{supp}_u(\boldsymbol{Q})$, excluding the nodes in $\mathcal{B}_u$. Let us define $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})$ as

$$p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \min_{\boldsymbol{x}\in\mathbb{R}^{n_u}} \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{c}_{\mathcal{B}_u}^\top \boldsymbol{\alpha}_{\mathcal{B}_u} + \left(\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{J}_u}\boldsymbol{x} + \boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_u}\boldsymbol{x} + \boldsymbol{c}_{\mathcal{J}_u}^\top \boldsymbol{x} + \boldsymbol{\lambda}_{\mathcal{J}_u}^\top \boldsymbol{s}\right)$$

$$\text{s.t.} \quad x_i(1-s_i) = 0 \quad i \in \mathcal{J}_u.$$

It is easy to verify that $f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \min_{\boldsymbol{s}\in\{0,1\}^{n_u}}\{p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})\}$. Therefore, it remains to characterize the explicit form of $p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u})$ for every $\boldsymbol{s} \in \{0,1\}^{n_u}$, and show that it is strongly convex and quadratic.

First, let $\mathcal{J}_{u,s} = \{i \in \mathcal{J}_u \mid s_i = 1\}$. Therefore, we have

$$\begin{aligned} p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) =& \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{c}_{\mathcal{B}_u}^\top \boldsymbol{\alpha}_{\mathcal{B}_u} \\ &+ \min_{\boldsymbol{x}\in\mathbb{R}^{|\mathcal{J}_{u,\boldsymbol{s}}|}} \left\{ \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\boldsymbol{x} + \boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}\boldsymbol{x} + \boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^\top \boldsymbol{x} + \sum_{i\in\mathcal{J}_{u,\boldsymbol{s}}} \boldsymbol{\lambda}_i \right\}. \end{aligned}$$

From the Karush-Kuhn-Tucker conditions, it follows that

$$\begin{aligned} p_{u,\boldsymbol{s}}(\boldsymbol{\alpha}_{\mathcal{B}_u}) =& \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}^\top \left(\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u} - \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}} \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^\top\right) \boldsymbol{\alpha}_{\mathcal{B}_u} \\ &+ \left(\boldsymbol{c}_{\mathcal{B}_u} - \boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^T \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^\top\right)^\top \boldsymbol{\alpha}_{\mathcal{B}_u} \\ &+ \left(-\frac{1}{2}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}}^\top (\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}})^{-1}\boldsymbol{c}_{\mathcal{J}_{u,\boldsymbol{s}}} + \sum_{i\in\mathcal{J}_{u,\boldsymbol{s}}} \lambda_i\right). \end{aligned}$$

Furthermore, note that $\left(\boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u} - \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}} \left(\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}},\mathcal{J}_{u,\boldsymbol{s}}}\right)^{-1} \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{J}_{u,\boldsymbol{s}}}^\top\right)$ is the Schur complement of $\boldsymbol{Q}_{\mathcal{J}_{u,\boldsymbol{s}}\cup\{\mathcal{B}_u\},\mathcal{J}_{u,\boldsymbol{s}}\cup\{\mathcal{B}_u\}}$, which, owing to the positive definiteness of $\boldsymbol{Q}$, is positive definite. This completes the proof. □

A.2 Proof of Lemma 2

First, we derive an explicit expression for g_v for each $v \in \mathrm{par}_{\mathsf{T}}(u)$. Next, we show that the optimization problem defining f_u can be decomposed into independent subproblems for each v.

Finally, we substitute the expressions for g_v within each subproblem to obtain the stated equation for f_u.

We start with the first step and derive an explicit expression for g_v. Fix $v \in \mathrm{par}(u)$ and consider the local parametric cost f_v. Setting the variables $\boldsymbol{x}_{\pi_v(\mathcal{B}_v)} = \boldsymbol{\alpha}_{\mathcal{B}_v}$ according to Constraint (4c) in the definition of f_v, we obtain:

$$\begin{aligned} f_v(\boldsymbol{\alpha}_{\mathcal{B}_v}) =& \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_v}^\top \boldsymbol{Q}_{\mathcal{B}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v} + \boldsymbol{c}_{\mathcal{B}_v}^\top \boldsymbol{\alpha}_{\mathcal{B}_v} \\ &+ \min_{\boldsymbol{x}\in\mathbb{R}^{n_v},\boldsymbol{z}\in\{0,1\}^{n_v}} \left\{ \frac{1}{2}\boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{J}_v}\boldsymbol{x}_{\pi_v(\mathcal{J}_v)} + \boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v} + \boldsymbol{c}_{\mathcal{J}_v}^\top \boldsymbol{x}_{\pi_v(\mathcal{J}_v)} + \boldsymbol{\lambda}_{\mathcal{J}_v}^\top \boldsymbol{z}_{\pi_v(\mathcal{J}_v)} \right\} \end{aligned} \tag{23a}$$

$$\text{s.t.} \qquad \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad \forall i \in \pi_v(\mathcal{J}_v). \tag{23b}$$

After isolating the contribution of the variable $\boldsymbol{\alpha}_v$, $\frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_v}^\top \boldsymbol{Q}_{\mathcal{B}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v} + \boldsymbol{c}_{\mathcal{B}_v}^\top \boldsymbol{\alpha}_{\mathcal{B}_v}$ can be rewritten as

$$\begin{aligned} \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_v}^\top \boldsymbol{Q}_{\mathcal{B}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v} + \boldsymbol{c}_{\mathcal{B}_v}^\top \boldsymbol{\alpha}_{\mathcal{B}_v} &= \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}^\top \boldsymbol{Q}_{\mathcal{B}_v\setminus v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{c}_{\mathcal{B}_v\setminus v}^\top \boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} \\ &\quad + \frac{1}{2}\boldsymbol{Q}_{v,v}\boldsymbol{\alpha}_v^2 + \boldsymbol{\alpha}_v \boldsymbol{Q}_{v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{c}_v\boldsymbol{\alpha}_v \\ &= \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) + \frac{1}{2}\boldsymbol{Q}_{v,v}\boldsymbol{\alpha}_v^2 + \boldsymbol{\alpha}_v \boldsymbol{Q}_{v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{c}_v\boldsymbol{\alpha}_v. \end{aligned} \tag{24}$$

Similarly, the term $\boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v}$ that appears in (23a) can be decomposed as

$$\boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v}\boldsymbol{\alpha}_{\mathcal{B}_v} = \boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,v}\boldsymbol{\alpha}_v. \tag{25}$$

We now recall the definition of g_v:

$$g_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) = \min_{\boldsymbol{x}_v\in\mathbb{R}} \left\{ f_v(\boldsymbol{x}_v, \boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) + \boldsymbol{\lambda}_v \mathbb{I}(\boldsymbol{x}_v) \right\}.$$

Substituting (24) and (25) together with (23), into the expression for g_v yields

$$\begin{aligned} g_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) = \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) + & \\ \min_{\boldsymbol{x}\in\mathbb{R}^{n_v+1},\boldsymbol{z}\in\{0,1\}^{n_v+1}} & \left\{ \frac{1}{2}\boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{J}_v}\boldsymbol{x}_{\pi_v(\mathcal{J}_v)} + \frac{1}{2}\boldsymbol{Q}_{v,v}\boldsymbol{x}_{\pi_v(v)}^2 + \right. \\ & \boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{x}_{\pi_v(v)}\boldsymbol{Q}_{v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \\ & \boldsymbol{x}_{\pi_v(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,v}\boldsymbol{x}_{\pi_v(v)} + \boldsymbol{c}_v\boldsymbol{x}_{\pi_v(v)} + \boldsymbol{c}_{\mathcal{J}_v}^\top \boldsymbol{x}_{\pi_v(\mathcal{J}_v)} + \\ & \left. \boldsymbol{\lambda}_v \boldsymbol{z}_{\pi_v(v)} + \boldsymbol{\lambda}_{\mathcal{J}_v}^\top \boldsymbol{z}_{\pi_v(\mathcal{J}_v)} \right\} \end{aligned} \tag{26a}$$

$$\text{s.t.} \qquad \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad \forall i \in \pi_v(\mathcal{J}_v \cup \{v\}). \tag{26b}$$

Having derived the expression for g_v, we now establish the equation for f_u stated in the lemma.

The nodes in the induced subgraph $\mathrm{supp}_u(\boldsymbol{Q})$ can be written as the disjoint union $\mathcal{B}_u \cup \mathcal{J}_u$, and the set $\mathcal{J}_u$ further decomposes as

$$\mathcal{J}_u = \bigcup_{v\in\mathrm{par}_\mathsf{T}(u)} (\mathcal{J}_v \cup \{v\}).$$

To formalize the properties of this decomposition, we state the following claim.

Claim 2. *For any two distinct $v_i, v_j \in \mathrm{par}_{\mathsf{T}}(u)$, the following holds*

1. $\big(\mathcal{J}_{v_i} \cup \{v_i\}\big) \cap \big(\mathcal{J}_{v_j} \cup \{v_j\}\big) = \emptyset$;
2. *For any* $\bar{i} \in \mathcal{J}_{v_i} \cup \{v_i\}$ *and* $\bar{j} \in \mathcal{J}_{v_j} \cup \{v_j\}$, *we have* $\boldsymbol{Q}_{\bar{i},\bar{j}} = 0$.

Proof of Claim 2. Suppose by contradiction there exists $\ell \in \big(\mathcal{J}_{v_i} \cup \{v_i\}\big) \cap \big(\mathcal{J}_{v_j} \cup \{v_j\}\big)$. By the running intersection property (the third condition of Definition 1) of a tree decomposition, the collection of bags containing ℓ induces a connected subtree of T. Since the bag $\mathcal{B}_u$ lies on the unique path connecting the subtrees rooted at v_i and v_j, it follows that $\ell \in \mathcal{B}_u$. On the other hand, $J_{v_i} \cup \{v_i\} \subset \mathcal{J}_u$ and $\mathcal{J}_{v_j} \cup \{v_j\} \subset \mathcal{J}_u$; moreover, by construction, $\mathcal{J}_u \cap \mathcal{B}_u = \emptyset$. Hence $\big(\mathcal{J}_{v_i} \cup v_i\big) \cap \mathcal{B}_u = \emptyset$ and $(\mathcal{J}_{v_j} \cup v_j) \cap \mathcal{B}_u = \emptyset$. This leads to a contradiction since $\ell \in \mathcal{B}_u$. Therefore, $\ell \in \big(\mathcal{J}_{v_i} \cup \{v_i\}\big) \cap \big(\mathcal{J}_{v_j} \cup \{v_j\}\big) = \emptyset$, thereby completing the proof of the first statement.

We proceed to prove the second statement. Fix $\bar{i} \in \mathcal{J}_{v_i} \cup \{v_i\}$ and $\bar{j} \in \mathcal{J}_{v_j} \cup \{v_j\}$. Suppose by contradiction $\boldsymbol{Q}_{\bar{i},\bar{j}} \neq 0$. Then, from the second condition of Definition 1, there must exist a bag that contains both $\bar{i}$ and $\bar{j}$. We proceed to show that this bag cannot exist.

By the first statement of the claim, the sets $\{\mathcal{J}_v \cup \{v\}\}_{v \in \mathrm{par}_{\mathsf{T}}(u)}$ are pairwise disjoint. Hence, since $\bar{i} \in \mathcal{J}_{v_i} \cup \{v_i\}$, we have $\bar{i} \notin \mathcal{J}_v \cup \{v\}, \forall v \in \mathrm{par}_{\mathsf{T}}(u) \backslash \{v_i\}$, and similarly $\bar{j} \notin \mathcal{J}_v \cup \{v\}, \forall v \in \mathrm{par}_{\mathsf{T}}(u) \backslash \{v_j\}$. Consequently, the only bags in the tree decomposition that can possibly contain both $\bar{i}$ and $\bar{j}$ are $\mathcal{B}_u$ and the bags $\{\mathcal{B}_v\}_{v \in \mathrm{par}_{\mathsf{T}}(u)}$. By construction, however, $\mathcal{B}_u$ contains neither $\bar{i}$ nor $\bar{j}$. Therefore, there must exist some $v \in \mathrm{par}_{\mathsf{T}}(u)$ such that $\mathcal{B}_v$ contains both $\bar{i}$ and $\bar{j}$. Now, by the running intersection property of tree decomposition, the collection of bags containing $\bar{i}$ forms a connected subtree of T, and the same holds for $\bar{j}$. Since $\mathcal{B}_v$ contains both nodes and u is adjacent to v, it follows that $\mathcal{B}_u$ must belong to at least one of these two connected subtrees. In particular, this implies that $\bar{i} \in \mathcal{B}_u$ or $\bar{j} \in \mathcal{B}_u$. This contradicts the fact that $\mathcal{B}_u$ contains neither $\bar{i}$ nor $\bar{j}$, thereby completing the proof of the second statement. □

Consider the local parametric cost f_u at node u. Similar to (23), we have

$$
\begin{aligned}
f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{B}_u}^\top \boldsymbol{Q}_{\mathcal{B}_u,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{c}_{\mathcal{B}_u}^\top \boldsymbol{\alpha}_{\mathcal{B}_u} + \\
\min_{\boldsymbol{x}\in\mathbb{R}^{n_u}, \boldsymbol{z}\in\{0,1\}^{n_u}} \left\{ \frac{1}{2}\boldsymbol{x}_{\pi_u(\mathcal{J}_u)}^\top \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{J}_u}\boldsymbol{x}_{\pi_u(\mathcal{J}_u)} + \boldsymbol{x}_{\pi_u(\mathcal{J}_u)}^\top \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{c}_{\mathcal{J}_u}^\top \boldsymbol{x}_{\pi_u(\mathcal{J}_u)} + \boldsymbol{\lambda}_{\mathcal{J}_u}^\top \boldsymbol{z}_{\pi_u(\mathcal{J}_u)} \right\} \\
\text{s.t.} \qquad \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad \forall i \in \pi_u(\mathcal{J}_u).
\end{aligned}
$$

From (7a), the first two terms of the objective function coincide with $h_u(\boldsymbol{\alpha}_{\mathcal{B}_u})$. We now turn to the remaining terms in the objective function. First, note that

$$
\begin{aligned}
\frac{1}{2}\boldsymbol{x}_{\pi_u(\mathcal{J}_u)}^\top \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{J}_u}\boldsymbol{x}_{\pi_u(\mathcal{J}_u)} &= \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\frac{1}{2}\boldsymbol{x}_{\pi_u(\mathcal{J}_v\cup v)}^\top \boldsymbol{Q}_{\mathcal{J}_v\cup v,\mathcal{J}_v\cup v}\boldsymbol{x}_{\pi_u(\mathcal{J}_v\cup v)}\Big) \\
&= \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\frac{1}{2}\boldsymbol{x}_{\pi_u(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{J}_v}\boldsymbol{x}_{\pi_u(\mathcal{J}_v)} + \frac{1}{2}\boldsymbol{Q}_{v,v}\boldsymbol{x}_{\pi_u(v)}^2 + \boldsymbol{x}_{\pi_u(v)}\boldsymbol{Q}_{v,\mathcal{J}_v}x_{\pi_u(\mathcal{J}_v)}\Big).
\end{aligned}
$$

This decomposition uses Claim 2, which asserts that the union $\mathcal{J}_u = \bigcup_{v\in\mathrm{par}_{\mathsf{T}}(u)}\big(\mathcal{J}_v \cup \{v\}\big)$ is disjoint and the cross blocks satisfy $\boldsymbol{Q}_{\mathcal{J}_{v_i}\cup\{v_i\},\,\mathcal{J}_{v_j}\cup\{v_j\}} = \boldsymbol{0}$, for all distinct $v_i, v_j \in \mathrm{par}_{\mathsf{T}}(u)$.

Next, we consider the the bilinear term coupling variables $\boldsymbol{x}_{\pi_u(\mathcal{J}_u)}$ and $\boldsymbol{\alpha}_{\mathcal{B}_u}$.

$$\begin{aligned}
\boldsymbol{x}_{\pi_u(\mathcal{J}_u)}^\top \boldsymbol{Q}_{\mathcal{J}_u,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} &= \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\boldsymbol{x}_{\pi_u(\mathcal{J}_v\cup v)}^\top \boldsymbol{Q}_{\mathcal{J}_v\cup v,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u}\Big) \\
&= \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\boldsymbol{x}_{\pi_u(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u} + \boldsymbol{x}_{\pi_u(v)}^\top \boldsymbol{Q}_{v,\mathcal{B}_u}\boldsymbol{\alpha}_{\mathcal{B}_u}\Big) \\
&= \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\boldsymbol{x}_{\pi_u(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{x}_{\pi_u(v)}^\top \boldsymbol{Q}_{v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}\Big).
\end{aligned}$$

To see the third equality, first note that the bag $\mathcal{B}_u$ admits the disjoint decomposition $\mathcal{B}_u = \big(\mathcal{B}_v \setminus \{v\}\big) \cup \big(\mathcal{B}_u \setminus \mathcal{B}_v\big)$, for any $v \in \mathrm{par}_{\mathsf{T}}(u)$. Furthermore, $\boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_u\setminus\mathcal{B}_v} = \boldsymbol{0}$, since there are no edges between nodes in $\mathcal{J}_v$ and nodes in $\mathcal{B}_u \setminus \mathcal{B}_v$.

We now turn to the remaining terms:

$$\boldsymbol{c}_{\mathcal{J}_u}^\top \boldsymbol{x}_{\pi_u(\mathcal{J}_u)} + \boldsymbol{\lambda}_{\mathcal{J}_u}^\top \boldsymbol{z}_{\pi_u(\mathcal{J}_u)} = \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(\boldsymbol{c}_{\mathcal{J}_v}^\top \boldsymbol{x}_{\pi_u(\mathcal{J}_v)} + \boldsymbol{c}_v \boldsymbol{x}_{\pi_u(v)} + \boldsymbol{\lambda}_{\mathcal{J}_v}^\top \boldsymbol{z}_{\pi_u(\mathcal{J}_v)} + \boldsymbol{\lambda}_v \boldsymbol{z}_{\pi_v(v)}\Big).$$

Substituting the terms derived above into the expression for f_u, we obtain

$$\begin{aligned}
f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = h_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) + \\
\min_{\boldsymbol{x}\in\mathbb{R}^{n_u},\boldsymbol{z}\in\{0,1\}^{n_u}} \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} \Big(&\frac{1}{2}\boldsymbol{x}_{\pi_u(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{J}_v}\boldsymbol{x}_{\pi_u(\mathcal{J}_v)} + \frac{1}{2}\boldsymbol{Q}_{v,v}\boldsymbol{x}_{\pi_u(v)}^2 + \boldsymbol{x}_{\pi_u(v)}\boldsymbol{Q}_{v,\mathcal{J}_v}x_{\pi_u(\mathcal{J}_v)} \\
&+ \boldsymbol{x}_{\pi_u(\mathcal{J}_v)}^\top \boldsymbol{Q}_{\mathcal{J}_v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} + \boldsymbol{x}_{\pi_u(v)}^\top \boldsymbol{Q}_{v,\mathcal{B}_v\setminus v}\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v} \\
&+ \boldsymbol{c}_{\mathcal{J}_v}^\top \boldsymbol{x}_{\pi_u(\mathcal{J}_v)} + \boldsymbol{c}_v \boldsymbol{x}_{\pi_u(v)} + \boldsymbol{\lambda}_{\mathcal{J}_v}^\top \boldsymbol{z}_{\pi_u(\mathcal{J}_v)} + \boldsymbol{\lambda}_v \boldsymbol{z}_{\pi_u(v)}\Big) \\
\text{s.t.} \quad & \boldsymbol{x}_i(1-\boldsymbol{z}_i) = 0 \qquad \forall i \in \pi_u(\mathcal{J}_u).
\end{aligned}$$

Since each term in the sum depends only on the variables in $\mathcal{J}_v \cup \{v\}$, the optimization problem decomposes into independent subproblems for each $v \in \mathrm{par}_{\mathsf{T}}(u)$. Furthermore, each subproblems equals $g_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}) - \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v})$ by (26). Hence, we conclude that,

$$f_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) = h_u(\boldsymbol{\alpha}_{\mathcal{B}_u}) + \sum_{v\in\mathrm{par}_{\mathsf{T}}(u)} g_v(\mathcal{B}_v\setminus v) - \phi_v(\boldsymbol{\alpha}_{\mathcal{B}_v\setminus v}).$$

□

A.3 Proof of Lemma 3

Let $\mathcal{J} = \{i : \boldsymbol{z}_i^\star = 1\}$. Then, $\boldsymbol{x}^\star = (\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\boldsymbol{c}_{\mathcal{J}}$, which implies

$$\|\boldsymbol{x}^\star\|_\infty = \big\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\boldsymbol{c}_{\mathcal{J}}\big\|_\infty \le \big\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\big\|_\infty \|\boldsymbol{c}_{\mathcal{J}}\|_\infty \le \big\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\big\|_\infty \|\boldsymbol{c}\|_\infty.$$

We now derive an upper bound on $\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\|_\infty$. By Lemma 5,

$$\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\|_\infty \le \max_{i\in\mathcal{J}} \sum_{j\in\mathcal{J}} \big|[\boldsymbol{Q}_{\mathcal{I},\mathcal{I}}^{-1}]_{\pi(i),\pi(j)}\big| \le \max_{i\in\mathcal{J}} \sum_{j\in\mathcal{J}} C_1\rho^{\mathrm{dist}(i,j)}.$$

Let $N_{r,i}$ denote the number of nodes in $\mathrm{supp}(\boldsymbol{Q})$ at distance exactly r from node i, and set $N_r := \max_i\{N_{r,i}\}$. Then

$$\max_{i\in\mathcal{J}} \sum_{j\in\mathcal{J}} C_1\rho^{\mathrm{dist}(i,j)} \;\le\; C_1 \sum_{r=0}^{n} N_r\rho^r.$$

We distinguish two cases:

- If $\boldsymbol{Q}$ is banded with bandwidth w, then $N_r \le 2w$ for all $0 \le r \le n$. Hence,

$$\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\|_\infty \le C_1 \sum_{r=0}^{n} 2w\,\rho^r \le 2wC_1 \sum_{r=0}^{\infty} \rho^r = \frac{2wC_1}{1-\rho}.$$

- If $\mathrm{supp}(\boldsymbol{Q})$ satisfies the polynomial volume-growth condition (Assumption 1), then $N_r \le \delta r^\gamma$. Therefore,

$$\|(\boldsymbol{Q}_{\mathcal{J},\mathcal{J}})^{-1}\|_\infty \le C_1 \sum_{r=0}^{n} \delta r^\gamma\rho^r \le \delta C_1 \sum_{r=0}^{\infty} r^\gamma\rho^r \le \frac{\delta\gamma!\,C_1}{(1-\rho)^{\gamma+1}},$$

where the last inequality uses the classical bound $\sum_{r=0}^{\infty} r^\gamma\rho^r \le \frac{\gamma!}{(1-\rho)^{\gamma+1}}$; see, e.g., [23, Proposition 1.4.4].

□

A.4 Proof of Lemma 4

Consider the quadratic equation $p_1(\boldsymbol{\alpha}) - p_2(\boldsymbol{\alpha}) = 0$. Let $\hat{\boldsymbol{\alpha}}$ denote its solution. Then

$$\begin{aligned}
& p_1(\hat{\boldsymbol{\alpha}}) - p_2(\hat{\boldsymbol{\alpha}}) = 0 \\
\Longrightarrow\; & \frac{1}{2}\hat{\boldsymbol{\alpha}}^\top(\boldsymbol{A}_1 - \boldsymbol{A}_2)\hat{\boldsymbol{\alpha}} + (\boldsymbol{b}_1 - \boldsymbol{b}_2)^\top\hat{\boldsymbol{\alpha}} + (d_1 - d_2) = 0 \\
\Longrightarrow\; & \frac{1}{2}\|\boldsymbol{A}_1 - \boldsymbol{A}_2\|_{1,1}\|\hat{\boldsymbol{\alpha}}\|_\infty^2 + \|\boldsymbol{b}_1 - \boldsymbol{b}_2\|_1\|\hat{\boldsymbol{\alpha}}\|_\infty + (d_1 - d_2) \ge 0 \\
\Longrightarrow\; & \bar{a}\|\hat{\boldsymbol{\alpha}}\|_\infty^2 + \bar{b}\|\hat{\boldsymbol{\alpha}}\|_\infty \ge \bar{d},
\end{aligned}$$

where, without loss of generality, we assume $d_2 \ge d_1$. We consider two cases. Indeed, if $\bar{a} = 0$, then

$$\|\boldsymbol{\alpha}\|_\infty \ge \frac{\bar{d}}{\bar{b}}.$$

On the other hand, if $\bar{a} > 0$, we have

$$\begin{aligned}
& \|\hat{\boldsymbol{\alpha}}\|_\infty^2 + \frac{\bar{b}}{\bar{a}}\|\hat{\boldsymbol{\alpha}}\|_\infty \ge \frac{\bar{d}}{\bar{a}} \\
\Longrightarrow\; & \|\hat{\boldsymbol{\alpha}}\|_\infty^2 + \frac{\bar{b}}{\bar{a}}\|\hat{\boldsymbol{\alpha}}\|_\infty + \frac{\bar{b}^2}{4\bar{a}^2} \ge \frac{\bar{d}}{\bar{a}} + \frac{\bar{b}^2}{4\bar{a}^2} \\
\Longrightarrow\; & \|\hat{\boldsymbol{\alpha}}\|_\infty \ge -\frac{\bar{b}}{2\bar{a}} + \sqrt{\frac{\bar{d}}{\bar{a}} + \frac{\bar{b}^2}{4\bar{a}^2}}.
\end{aligned}$$

Rearranging the terms, we get

$$\|\hat{\boldsymbol{\alpha}}\|_\infty \geq \frac{-\bar{b} + \sqrt{\bar{b}^2 + 4\bar{a}\bar{d}}}{2\bar{a}}.$$

Combining the two cases, we conclude that

$$\|\hat{\boldsymbol{\alpha}}\|_\infty \geq \begin{cases} \frac{-\bar{b}+\sqrt{\bar{b}^2+4\bar{a}\bar{d}}}{2\bar{a}} & \text{if } \bar{a} \neq 0, \\ \frac{\bar{d}}{\bar{b}} & \text{if } \bar{a} = 0. \end{cases}$$

□

A.5 Proof of Lemma 5

The proof proceeds via a polynomial approximation of the function $\psi(x) = x^{-1}$ on a compact interval, for which we use the following result.

Proposition 2 (Polynomial approximation of x^{-1},[57]). *Let $0 < a < b$. For $\ell \geq 0$, let Π_ℓ denote the set of polynomials of degree at most ℓ, and define*

$$e_\ell([a,b]) := \inf_{\bar{p}\in\Pi_\ell} \max_{x\in[a,b]} \left|x^{-1} - \bar{p}(x)\right|.$$

Set $r = b/a$ and

$$\rho := \frac{\sqrt{r}-1}{\sqrt{r}+1}.$$

Then

$$e_\ell([a,b]) = \frac{(1+\sqrt{r})^2}{2ar}\,\rho^{\ell+1}.$$

Let $a = \mu_{\min}(\boldsymbol{Q}), b = \mu_{\max}(\boldsymbol{Q})$, $C_0 = \frac{(1+\sqrt{\kappa_2})^2}{2\kappa_2\mu_{\min}(\boldsymbol{Q})}$ and $\sigma(\boldsymbol{Q})$ denote the set of eigen values of $\boldsymbol{Q}$. Since $\boldsymbol{Q}$ is positive definite and invertible, we have $0 < a < b$. From the Proposition 2 there exist a sequence of polynomials $\bar{p}_\ell \in \Pi_\ell$ satisfying

$$\max_{x\in[a,b]} \left\{\left|x^{-1} - \bar{p}_\ell(x)\right|\right\} = C_0\rho^{\ell+1}.$$

From spectral theory [62]

$$\begin{aligned} \|\boldsymbol{Q}^{-1} - \bar{p}_\ell(\boldsymbol{Q})\|_2 &= \max_{x\in\sigma(A)} \left\{\left|x^{-1} - \bar{p}_\ell(x)\right|\right\} \\ &\leq \max_{x\in[a,b]} \left\{\left|x^{-1} - \bar{p}_\ell(x)\right|\right\} \\ &= C_0\rho^{\ell+1}. \end{aligned}$$

Moreover, for every integer $k \geq 0$ we have

$$(\boldsymbol{Q}^k)_{ij} = 0 \qquad \text{whenever } \operatorname{dist}(i,j) > k.$$

Hence, if $\bar{p}_k \in \Pi_k$ then $\bar{p}_k(\boldsymbol{Q})$ satisfies

$$\bar{p}_k(\boldsymbol{Q})_{ij} = 0 \qquad \text{whenever } \operatorname{dist}(i,j) > k,$$

since $\bar{p}_k(\boldsymbol{Q})$ is a linear combination of $\mathbf{I}, \boldsymbol{Q}, \ldots, \boldsymbol{Q}^k$.

Let $i \neq j$ and choose $\ell = \mathrm{dist}(i,j) - 1$. Then $\bar{p}_\ell(\boldsymbol{Q})_{ij} = 0$, and hence

$$\left|\boldsymbol{Q}^{-1}_{ij}\right| = \left|\boldsymbol{Q}^{-1}_{ij} - \bar{p}_\ell(\boldsymbol{Q})_{ij}\right| \leq \left\|\boldsymbol{Q}^{-1} - \bar{p}_\ell(\boldsymbol{Q})\right\|_2 \leq C_0 \rho^{\mathrm{dist}(i,j)}.$$

If $i = j$, note that

$$\left|\boldsymbol{Q}^{-1}_{ii}\right| \leq \left\|\boldsymbol{Q}^{-1}\right\|_2 = \frac{1}{a},$$

which is consistent with the stated bound. This completes the proof. □

B Heuristic for PRUNE

To determine the set of relevant functions, the exact version of `PRUNE` requires comparing every pair of quadratic functions, leading to a runtime that is quadratic in N, where N is the total number of functions of the input.

To mitigate this quadratic cost, we provide a heuristic version, given in Algorithm 4. This procedure processes the functions in a single pass, requiring only $N-1$ comparisons. This linear-time approach may retain some functions that are formally irrelevant, but the total number of retained functions remains consistent with the bound established in Lemma 9.

Algorithm 4 heuristic-`PRUNE`

Input: $\tilde{f}(\boldsymbol{\alpha}) \equiv [p_i(\boldsymbol{\alpha})]\ \forall i = 1, \ldots, N$ and the constant U.
Output: f^{prune}

$f^{\mathrm{prune}} \leftarrow [p_1]$ ▷ Add p_1 to f^{prune}
$j = 1$
for $i = 2, \ldots, N$ **do**
  Calculate $L(p_i, p_j)$ from Equation (8) ▷ Lower bound of roots of $p_i - p_j = 0$
  if $L(p_i, p_j) > U$ **then**
    if $d_i < d_j$ **then**
      $f^{\mathrm{prune}} \leftarrow \texttt{delete}\,(f^{\mathrm{prune}}, \texttt{end}\,(f^{\mathrm{prune}}))$ ▷ Remove the last function p_j from f^{prune}
      $f^{\mathrm{prune}} \leftarrow \texttt{append}\,(f^{\mathrm{prune}}, p_i)$ ▷ Add p_i to f^{prune}
      $j = i$
    end if
  else
    $f^{\mathrm{prune}} \leftarrow \texttt{append}(f^{\mathrm{prune}}, p_i)$
    $j = i$
  end if
end for
return f^{prune}

This improvement is possible when the quadratic functions are stored where every consecutive pair of functions is m-similar. In this case, it suffices to check for irrelevance only among consecutive pairs of functions. When $\boldsymbol{Q}$ has a tree decomposition that is a path (including the banded case), we note that this ordering can be maintained with no additional cost. However, in the general case, storing the functions in the desired order requires additional steps, which can increase computational

cost, making the heuristic inefficient. For this reason, we rely on the version of `PRUNE` presented in Algorithm 3 for the general case.