FULL LENGTH PAPER

Series B

A parametric approach for solving convex quadratic optimization with indicators over trees

Aaresh Bhathena[1] · **Salar Fattahi**[1] · **Andrés Gómez**[2] ·
Simge Küçükyavuz[3]

Abstract

This paper investigates convex quadratic optimization problems involving n indicator variables, each associated with a continuous variable, particularly focusing on scenarios where the matrix Q defining the quadratic term is positive definite and its sparsity pattern corresponds to the adjacency matrix of a tree graph. We introduce a graph-based dynamic programming algorithm that solves this problem in time and memory complexity of $\mathcal{O}(n^2)$. Central to our algorithm is a precise parametric characterization of the cost function across various nodes of the graph corresponding to distinct variables. Our computational experiments conducted on both synthetic and real-world datasets demonstrate the superior performance of our proposed algorithm compared to existing algorithms and state-of-the-art mixed-integer optimization solvers. An important application of our algorithm is in the real-time inference of Gaussian hidden Markov models from data affected by outlier noise. Using a real on-body accelerometer dataset, we solve instances of this problem with over 30,000 variables in under a minute, and its online variant within milliseconds on a standard computer. A Python implementation of our algorithm is available at https://github.com/aareshfb/Tree-Parametric-Algorithm.git.

✉ Salar Fattahi
fattahi@umich.edu

Aaresh Bhathena
aareshfb@umich.edu

Andrés Gómez
gomezand@usc.edu

Simge Küçükyavuz
simge@northwestern.edu

[1] Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA

[2] Daniel J. Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA, USA

[3] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA

1 Introduction

Given a symmetric and positive definite matrix $Q \in \mathbb{R}^{n \times n}$ and vectors $\lambda, c \in \mathbb{R}^n$, we study the following mixed-integer quadratic optimization (MIQP) problem:

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

$$\text{s.t.} \quad x_i (1 - z_i) = 0 \qquad\qquad i = 1, 2, \ldots, n \tag{1b}$$

Specifically, we assume that the sparsity pattern of $Q \in \mathbb{R}^{n \times n}$ is the adjacency matrix of a connected tree. The binary vector $z \in \{0, 1\}^n$ is used to model the support of the vector $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^n$ is a regularization parameter on the sparsity of x. If $z_i = 0$, then from constraint (1b), we obtain $x_i = 0$. On the other hand, if $z_i = 1$ then x_i is unconstrained. Without loss of generality, we assume that the diagonal elements of Q are equal to 1. This can be ensured by replacing $x_i \leftarrow x_i / \sqrt{Q_{i,i}}$ for all $i = 1, \ldots, n$. We also assume $\lambda_i > 0$ for every $i \in \{1, 2, \ldots, n\}$, as any $\lambda_i \leq 0$ implies $z_i = 1$ at optimality. The regularizer λ can model the sparsity of the solution, which makes the above problem useful in network inference [22, 44], sparse regression [6, 19], and probabilistic graphical models [37, 40, 41], to name a few.

1.1 Gaussian hidden Markov models

An important application of Problem (1) is in the inference of *Gaussian hidden Markov models* (GHMM) [3, 22], where the goal is to estimate hidden states $\{x_t\}_{t=1}^T$ of a random process from K_t observations $\{y_{k,t}\}_{k=1}^{K_t}$ at each time t. We consider the Besag model [8, 9], where the hidden states are assumed to be jointly Gaussian:

$$p(x_1, \ldots, x_T) \propto \exp\left(-\frac{1}{2\sigma_1} x_1^2 - \sum_{t=2}^T \frac{1}{2\sigma_t^2} (x_t - x_{t-1})^2 \right). \tag{2}$$

Each hidden state x_t is indirectly observed via noisy observations $y_{k,t} = x_t + \epsilon_{k,t} + \delta_{k,t}, k = 1, \ldots, K_t$, where $\epsilon_{k,t}$ is a dense, but light-tailed noise drawn from $\mathcal{N}(0, v_t^2)$, whereas $\delta_{k,t}$ is an outlier noise that corrupts only a small subset of the observations. An example of a GHMM is given in Fig. 1.

One of the earliest applications of GHMMs can be traced back to signal processing, aimed at predicting the evolution of a random signal over time by effectively filtering out observational noise [10, 35]. A more contemporary application of GHMMs lies
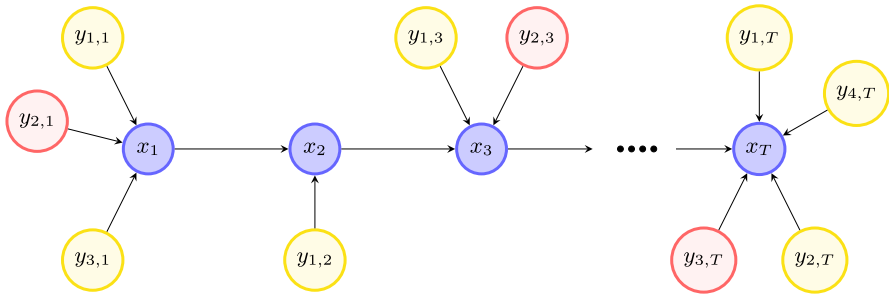
Fig. 1 An illustration of a GHMM. At every time t, observations $\{y_{k,t}\}_{k=1}^{K_t}$ of the hidden state x_t are collected, some of which may be corrupted with outlier noise (shown in red). The goal is to infer the hidden states $\{x_t\}_{t=1}^{T}$ from these corrupted observations (color figure online)

in activity recognition utilizing on-body wearable accelerometers [36, 47]. In such contexts, additional consideration may involve assuming sparsity in the underlying hidden state $\{x_t\}_{t=1}^{T}$, which corresponds to the inactive state of the body. Under such settings, it is natural to consider the *maximum a posteriori estimate* of the hidden states with ℓ_0 regularization to promote the sparsity prior on the outliers as well as the underlying hidden states. This problem can be formulated as follows:

$$
\min_{x,z,w,s} \quad \sum_{t=1}^{T}\sum_{k=1}^{K_t} \frac{1}{v_t^2}\left(y_{k,t}-x_t-w_{k,t}\right)^2 + \frac{1}{\sigma_1^2}x_1^2 + \sum_{t=2}^{T}\frac{1}{\sigma_t^2}(x_t-x_{t-1})^2
$$

$$
+ \sum_{t=1}^{T}\sum_{k=1}^{K_t}\lambda_{k,t}z_{k,t} + \sum_{t=1}^{T}\gamma_t s_t \tag{3a}
$$

$$
\text{s.t.} \quad w_{k,t}(1-z_{t,k}) = 0 \qquad\qquad t=1,2\ldots T; k=1,\ldots K_t \tag{3b}
$$

$$
x_t(1-s_t) = 0 \qquad\qquad t=1,2\ldots T \tag{3c}
$$

$$
w_{\cdot,t}\in\mathbb{R}^{K_t},\ z_{\cdot,t}\in\{0,1\}^{K_t} \qquad t=1,2\ldots T \tag{3d}
$$

$$
x\in\mathbb{R}^T,\ s\in\{0,1\}^T. \tag{3e}
$$

In the optimization problem (3), the binary variables $\{z_{k,t}\}$ capture the presence of outlier noise in observations $\{y_{k,t}\}$. Specifically, $z_{k,t}=1$ indicates that $y_{k,t}$ is likely to be tainted with noise. This can be understood by noting that when $z_{k,t}=1$, the continuous variable $w_{k,t}$ takes the value $y_{k,t}-x_t$ at optimality, thereby nullifying the impact of the observation $y_{k,t}$ on the estimated state x_t. Conversely, $z_{k,t}=0$ implies $w_{k,t}=0$, indicating that the observation $y_{k,t}$ is devoid of outlier noise. Moreover, the binary variables $\{s_t\}$ capture the support of the hidden state $\{x_t\}$, enforcing $x_t=0$ if and only if $s_t=0$. The above optimization problem is a special case of Problem (1), where the matrix Q is positive definite and its support is the adjacency matrix of a tree graph (as can be seen in Fig. 1). An important variant of problem (3), arising in real-time monitoring and detection of events, is the online variant where observations $\{y_{k,t}\}$ become available over time [53]. In such scenarios, where rapid action is necessary

upon detecting anomalous events, re-optimization of Problem (3) must be performed within milliseconds.

In most cases, Problem (3) is rarely tackled in the literature directly. Indeed, mixed-integer nonlinear optimization problems are typically regarded as intractable. Moreover, big-M relaxations of (3) result in poor relaxations with trivial lower bounds, thus simply resorting to off-the-shelf solvers may prove ineffective. Thus, practitioners often resort to simpler approximations, consisting of either using ℓ_1-norm penalty to induce sparsity on variables x and w [53], or using iterative procedures and heuristics to remove outliers [15, 48]. Naturally, such approximations admit fast algorithms, but the solution quality can be negatively affected. Recently, there has been a renewed interest in developing improved mixed-integer optimization formulations for problems with sparsity and outliers [22, 23, 29, 34]. The results indicate that exact methods can indeed deliver substantially better solutions, especially when outliers are clustered. Typical runtimes of mixed-integer optimization methods with strong formulations is measured in minutes for problems with T in the hundreds, which is adequate for small-sized offline versions for (3), but far from practical for online problems. In this paper, we propose a method that solves the online problem to optimality within milliseconds on a standard computer.

1.2 Related work

For general dense matrix Q, Problem (1) is NP-hard [16, 33]. Earlier methods based on mixed-integer programming using big-M formulation [6, 7, 18] work reliably for small instances, but exhibit poor scalability [31, 32]. Since then, there has been a significant improvement in solving these problems over large instances. One key contribution was the *perspective-reformulation* technique that obtains high-quality convex relaxations of the feasible region. Initially introduced in [46], perspective reformulations have served as the cornerstone for numerous methods aimed at solving Problem (1) with general Q, either exactly or approximately [5, 24, 27, 28, 49–51].

Due to the NP-hardness of Problem (1) with a general Q, recent endeavors have shifted focus towards cases where Q possesses a special structure. When Q exhibits a diagonal structure, it has been demonstrated that a perspective reformulation already yields the ideal convex hull characterization of Problem (1) [13]. Moreover, if the matrix Q can be factorized as $Q = Q_0^\top Q_0$, where Q_0 is sparse, the problem can be solved in polynomial time under appropriate conditions [19]. In [17], a cardinality-constrained version of Problem (1) is explored, where Q corresponds to a tree with a maximum degree d, and all coefficients λ_i are identical. The authors propose a dynamic programming algorithm that operates in $\mathcal{O}(n^3 d)$ time. Consequently, this leads to a $\mathcal{O}(n^4)$ algorithm for the regularized version discussed in this paper, with the additional restriction that all coefficients λ_i are identical.

Perhaps most closely related to our work are two lines of research that investigate Problem (1) when Q possesses either a path or Stieltjes structure. When Q is a Stieltjes matrix, it is recently shown that Problem (1) can be converted into a sub-modular minimization problem and thus solved in polynomial time [2, 30]. Any Q that has a tree structure can be turned into a Stieltjes matrix with a simple change

of variables. Therefore, Problem (1) can be solved in polynomial time. An application of the state-of-the-art submodular minimization algorithm leads to a runtime of $\mathcal{O}(n^5 \text{EO})$, where EO is the complexity of solving a certain quadratic program [43]. Although this complexity is expected to be improved with more recent algorithms such as those introduced in [14, 39], they remain inefficient in medium to large-scale instances. When Q takes the form of a tridiagonal matrix, the works [20, 21, 40] have introduced dynamic programming (DP) algorithms capable of solving Problem (1) in $\mathcal{O}(n^2)$. However, in Sect. 2, we provide a detailed discussion on why these dynamic programming algorithms cannot be readily extended to the more general case of tree structures for Q.

1.3 Preliminaries and notations

Given a matrix $Q \in \mathbb{R}^{n \times n}$ and index sets I and J, we denote by $Q_{I,J}$ the sub-matrix of Q whose rows and columns correspond to I and J, respectively. Similarly, given a vector $c \in \mathbb{R}^n$, we denote by c_J a sub-vector of c with indices corresponding to J. For integers $k < l$, we define $[k : l] = [k, k+1, \ldots, l]$. We use $\mathbb{1}_x$ to denote the indicator function defined over $\mathbb{R}$ that returns 0 for $x = 0$ and returns 1 for all $x \neq 0$. The notations $f^\star$ and $(x^\star, z^\star)$ are used to denote the optimal objective value and optimal solution of Problem (1) respectively.

Definition 1 Given a symmetric matrix $Q \in \mathbb{R}^{n \times n}$, the support graph of Q, denoted by $\text{supp}(Q)$, is an undirected graph $\mathcal{G} = (N, E)$, where $N = \{1, \ldots, n\}$ and $(i, j) \in E$ if and only if $Q_{i,j} \neq 0$ for $i \neq j$.

Note that the diagonal entries of Q do not affect the construction of $\text{supp}(Q)$.

In this paper, we consider problems where $\text{supp}(Q)$ is a tree. Without loss of generality, we assume that $\text{supp}(Q)$ is connected and rooted. Moreover, we assume that the edges have a natural orientation toward the root node.[1] We use $\text{child}(u)$ to denote the child node of u in the rooted tree $\text{supp}(Q)$. Similarly, we use $\text{par}(u)$ to denote the set of parent nodes of u. We assume a topological ordering for the nodes in $\text{supp}(Q)$. More specifically, we assume $u < \text{child}(u)$ for every node in $\text{supp}(Q)$. Therefore, node n is always the root node. Since $\text{supp}(Q)$ is a tree, its topological labeling always exists and can be obtained in $\mathcal{O}(n)$ time and memory [1, Algorithm 3.8]. Moreover, due to the considered directions, each node can only have a single child, but potentially multiple parents. Figure 2 illustrates the topological ordering of an exemplary tree. The degree of each node u in $\text{supp}(Q)$ is defined as the number of edges connected to u, which is equal to the total number of its parent and child nodes, and is denoted as $\deg(u)$. If $\deg(u) \geq 3$, we say u is a *branch*. Trees with only one branch are referred to as *extended star trees*.

Given any node u, $\text{supp}_u(Q)$ denotes the largest connected sub-tree of $\text{supp}(Q)$ comprised of u and its ancestors, that is, any node $v \leq u$ that has a path to u. The symbol n_u denotes the number of nodes in $\text{supp}_u(Q)$. Given a node u, we define $Q_{[u]}$ as the sub-matrix of Q with rows and columns corresponding to the nodes in

---

[1] Recall that $\text{supp}(Q)$ is undirected; the natural orientation assumption is only to streamline the presentation of our algorithm.

Fig. 2 An example of the
topological labeling of nodes of
a tree. In this example,
child(4) = 5 and par(4) = {2, 3}
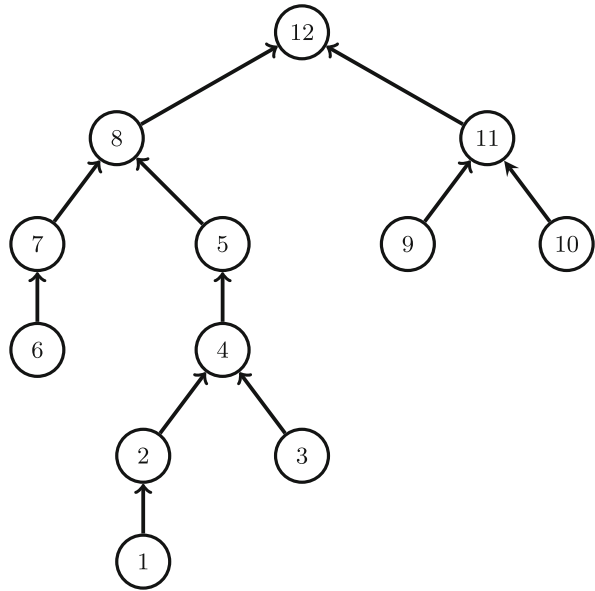
$\text{supp}_u(Q)$. It follows that $\text{supp}_u(Q) = \text{supp}(Q_{[u]})$. Similarly, we define $c_{[u]}$ and $\lambda_{[u]}$ as the sub-vectors of c and λ with indices corresponding to the nodes of $\text{supp}(Q_{[u]})$.

We refer to $f_u(\alpha)$ as the *parametric cost at node u*, defined as:

$$f_u(\alpha) := \min_{x \in \mathbb{R}^{n_u}, z \in \{0,1\}^{n_u}} \frac{1}{2} x^\top Q_{[u]} x + c_{[u]}^\top x + \lambda_{[u]}^\top z \tag{4a}$$

$$\text{s.t.} \quad x_i(1 - z_i) = 0 \qquad i = 1, 2 \ldots, n_u \tag{4b}$$

$$x_{n_u} = \alpha. \tag{4c}$$

In simpler terms, $f_u(\alpha)$ represents the optimal cost of the sub-problem defined over the sub-tree $\text{supp}_u(Q)$ when the root node variable x_{n_u} is set to α. We define $f_u^\star = \min_\alpha f_u(\alpha)$. Indeed, we have $f^\star = f_n^\star$.

Our next lemma shows that $f_u(\alpha)$ can be written as the sum of an indicator function and the minimum of at most 2^{n_u-1} quadratic and strongly convex functions, $p_{u,s}(\cdot)$, where s is a binary vector of dimension $n_u - 1$ that indicates which variables in the subtree are nonzero. For each configuration of s, the resulting optimization problem can be shown to be strongly convex quadratic. The proof of this lemma is presented in Appendix A.

Lemma 1 *Suppose that Q is positive definite and $\text{supp}(Q)$ is a tree. For any $1 \le u \le n$, $f_u(\alpha)$ can be written as:*

$$f_u(\alpha) = \min_{s \in \{0,1\}^{n_u-1}} \{p_{u,s}(\alpha)\} + \lambda_u \mathbb{1}_\alpha,$$

where, for every $s \in \{0, 1\}^{n_u - 1}$, $p_{u,s}(\alpha)$ *is quadratic and strongly convex. In particular, let* $J_s = \{i : i \in J, s_i = 1\}$, *where* J *is the set of nodes in* $\mathrm{supp}_u(Q)$, *excluding* u. *Then* $p_{u,s}(\alpha)$ *is given by:*

$$p_{u,s}(\alpha) = \frac{1}{2}\left(1 - Q_{u,J_s}\left(Q_{J_s,J_s}\right)^{-1} Q_{u,J_s}^\top\right)\alpha^2 + \left(c_u - c_{J_s}^T\left(Q_{J_s,J_s}\right)^{-1} Q_{u,J_s}^\top\right)\alpha$$
$$+ \left(-\frac{1}{2}c_{J_s}^\top (Q_{J_s,J_s})^{-1} c_{J_s} + \sum_{i \in J_s} \lambda_i\right).$$

This paper extensively uses conjugate functions. Recall that, given a function $f : \mathbb{R} \to \mathbb{R}$, its Fenchel conjugate is defined as

$$f^*(\beta) = \sup_{\alpha}\{\alpha\beta - f(\alpha)\}.$$

A fundamental property of $f^*(\beta)$ is that it is convex, even if $f(\alpha)$ is not. Moreover, $f(\alpha) + f^*(\beta) \geq \beta\alpha$, for every $\alpha, \beta \in \mathbb{R}$. Geometrically, $-f^*(\beta)$ corresponds to the intercept of a tangent to f with slope β.

2 Dynamic programming over trees

When $\mathrm{supp}(Q)$ is a path graph, Problem (1) reduces to the following optimization problem:

$$\min_{x \in \mathbb{R}^n, z \in \{0,1\}^n} \quad \left(\frac{1}{2}\sum_{i=1}^n x_i^2 + \sum_{i=2}^n Q_{i,i-1}x_i x_{i-1}\right) + c^\top x + \lambda^\top z \tag{5a}$$
$$\text{s.t.} \quad x_i(1 - z_i) = 0 \quad i = 1, 2 \ldots, n. \tag{5b}$$

Liu et al. [40] introduced a DP approach for solving the above problem. To explain this method, let $q^\star_{[k:l]}$ denote the optimal cost of (5) with additional constraints $z_i = 1$ for every $k \leq i \leq l$ and $z_i = 0$ otherwise. A simple calculation reveals that

$$q^\star_{[k:l]} = \begin{cases} -\frac{1}{2}c_{[k:l]}^\top \left(Q_{[k:l],[k:l]}\right)^{-1} c_{[k:l]} + \sum_{i=k}^l \lambda_i & 1 \leq k \leq l \leq n \\ 0 & k > l. \end{cases}$$

Let s be the largest index such that $z_s^\star = 0$. Upon setting $z_s = 0$ in (5), the problem decomposes into two sub-problems: one defined over nodes $\{1, \ldots, s-1\}$ and the other over nodes $\{s+1, \ldots, n\}$ with the additional constraint that $z_i = 1$ for every $i > s$. This decomposition implies that $f^\star = q^\star_{[s+1:n]} + f^\star_{s-1}$. More generally, one can write:

$$f_i^\star = \min_{0 \leq s \leq i}\left\{q^\star_{[s+1:i]} + f^\star_{s-1}\right\}, \qquad f_u^\star = 0 \text{ for } u \leq 0. \tag{6}$$

Fig. 3 Simple tree with one branch at its root. A possible choice of s is highlighted in red (color figure online)
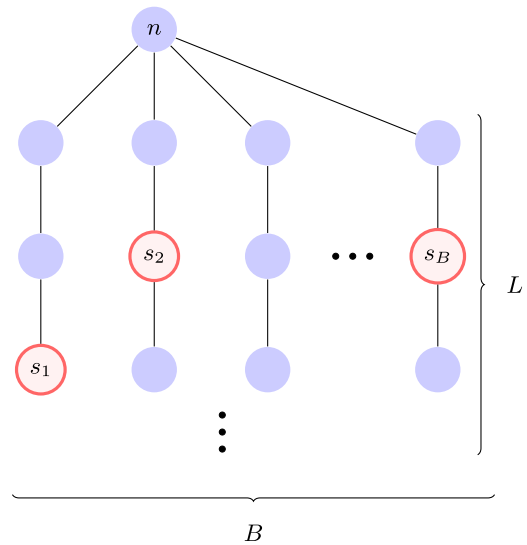
The values of $\{q_{[k:l]}\}_{k \leq l}$ can be computed in $\mathcal{O}(n^2)$ according to [40, Proposition 2]. Given the values of $\{q_{[k:l]}\}_{k \leq l}$, the values of $f_1^\star, f_2^\star, \ldots, f_n^\star$ can be obtained via the recursive equation (6) in $\mathcal{O}(n^2)$. Consequently, an overall $\mathcal{O}(n^2)$ algorithm for solving (5) emerges. The corresponding optimal solution can also be recovered with a negligible overhead (see [40] for more details).

A similar DP approach can be extended to trees beyond paths. This extension is particularly viable due to trees inheriting a similar decomposability property: when $z_s = 0$ for some node s in the tree, Problem (1) decomposes into smaller sub-problems defined over the sub-trees, each rooted at one of the parents of s, along with a simple quadratic program over the remaining nodes of the tree. Unfortunately, our next example illustrates that this decomposability property is not enough to guarantee the efficiency of the corresponding DP, especially when the tree possesses multiple branches.

Example 1 (Extended star trees) Consider an extended star tree with only one branch located at the root. Let B denote the number of branches in the tree, each composed of L nodes. We define a vector $s \in \{0, \ldots, L\}^B$, where each s_b is either 0 or it corresponds to the s_b-th node in branch b, with the indices increasing away from the leaf node. If $s_b = 0$, then the vector s excludes any node from branch b. Figure 3 depicts the structure of this graph.

For any $s \in \{0, \ldots, L\}^B$ and $b \in \{1, \ldots, B\}$, let $f_{s_b,b}^\star$ denote the optimal cost of the sub-problem defined over the sub-tree rooted at node s_b within branch b. Since this sub-tree is a path, the corresponding $f_{s_b,b}^\star$ can be obtained efficiently via the aforementioned DP algorithm. We set $f_{u,b}^\star = 0$ for every $u \leq 0$. Let $s \in \{0, \ldots, L\}^B$ denote the set of nodes with the largest indices in each branch such that $z_s^\star = 0$. Accordingly, let $q_s^\star$ denote the optimal cost of Problem (1) with additional constraints that $z_i = 0$ for all nodes i within the sub-tree rooted at node s_b for branches $b = 1, \ldots, B$, and $z_i = 1$

otherwise. The optimal cost $f^\star$ can be written as:

$$f^\star = \min_{s \in \{0,\dots,L\}^B} \left\{ q_s^\star + \sum_{b=1}^{B} f_{s_b-1,b}^\star \right\}. \tag{7}$$

The aforementioned equation implies that, even if $f_{s_b-1,b}^\star$ and $q_s^\star$ can be obtained efficiently, one needs to perform up to $(L+1)^B$ comparisons to determine the optimal cost $f^\star$, a process that becomes inefficient with the increasing number of branches.

To address the inefficiency inherent in the direct DP approach when applied to general tree structures, we introduce a parametric characterization of the optimal cost for Problem (1). Through this characterization, we demonstrate a significant reduction in the search space of the DP approach, without sacrificing the optimality of the obtained solution. Toward this goal, in Sect. 3.1, we revisit Problem (1) for path graphs. Here, we present a parametric algorithm for this problem with runtime and memory complexities of $\mathcal{O}(n^2)$, matching the worst-case complexity of the DP approach proposed in [40], but significantly outperforming it in practice. Building upon our parametric algorithm for path graphs, in the remainder of Sect. 3, we extend our approach to general tree structures, showing that it can solve these problems in a similar $\mathcal{O}(n^2)$ time and memory. In Sect. 4, we discuss an important practical consideration regarding our algorithm. Finally, in Sect. 5, we assess the performance of our proposed approach across various case studies. Surprisingly, while its worst-case complexity is $\mathcal{O}(n^2)$, we observe that the practical runtime of our proposed algorithm is close to linear in our computations on synthetic and real-world accelerometer datasets.

3 Parametric algorithm

We first provide a high-level intuition of the proposed algorithm. Recall the definition of $f(\alpha)$, which we repeat for convenience:

$$f_u(\alpha) := \min_{x \in \mathbb{R}^{n_u}, z \in \{0,1\}^{n_u}} \quad \frac{1}{2}x^\top Q_{[u]}x + c_{[u]}^\top x + \lambda_{[u]}^\top z \tag{8a}$$

$$\text{s.t.} \quad x_i(1-z_i) = 0 \qquad i = 1,2\dots,n_u \tag{8b}$$

$$x_{n_u} = \alpha. \tag{8c}$$

Note that, as the value function of a mixed-integer problem, the parametric cost f_u is not convex. Nonetheless, a key observation is that, since the support graph is a tree, once the value of x_{n_u} is fixed, the problem decomposes into independent subproblems, one for each parent of u. More specifically, Problem (8) reduces to

$$f_u(\alpha) = \frac{1}{2}Q_{u,u}\alpha^2 + c_u\alpha + \lambda_u \mathbb{1}_\alpha + \sum_{v \in \text{par}(u)} \min_{\xi \in \mathbb{R}} \{f_v(\xi) + Q_{uv}\xi\alpha\}$$

$$= \frac{1}{2} Q_{u,u}\alpha^2 + c_u\alpha + \lambda_u \mathbb{1}_\alpha - \sum_{v \in par(u)} f_v^*(-Q_{uv}\alpha). \tag{9}$$

Since the parametric cost f_u can be characterized merely based on the conjugate of the parametric cost of its parents, we can imagine an algorithm that traverses the graph in topological order, and recursively computes and stores in each node u either the parametric cost f_u or its conjugate f_u^*. These functions turn out to be piece-wise quadratic, as elaborated in the following definition.

Definition 2 A continuous function $f : \mathbb{R} \to \mathbb{R}$ is termed piece-wise quadratic with N pieces if there exist scalars $-\infty = \tau_0 < \tau_1 < \cdots < \tau_N = +\infty$ (also referred to as breakpoints) and quadratic functions (also referred to as pieces) $p_1, \ldots, p_N$ such that $f(\alpha) = p_k(\alpha)$ for $\tau_{k-1} \le \alpha \le \tau_k$, where $k = 1, \ldots, N$ and $p_k(\alpha) \ne p_{k+1}(\alpha)$ for some $\alpha \in \mathbb{R}$.

Upon assuming $p_k(\alpha) = \gamma_{k,1}\alpha^2 + \gamma_{k,2}\alpha + \gamma_{k,3}; k = 1, \ldots, N$, the condition $p_k(\alpha) \ne p_{k+1}(\alpha)$ for some $\alpha \in \mathbb{R}$ is equivalent to $\gamma_{k,i} \ne \gamma_{k+1,i}$ for some $i \in \{1, 2, 3\}$. Moreover, to store and represent a piece-wise quadratic function with N pieces, it suffices to store an ordered list of the breakpoints, along with the coefficients of their corresponding quadratic pieces $[(\tau_k, \gamma_{k,1}, \gamma_{k,2}, \gamma_{k,3})]_{k=1}^N$.

Equation (9) involves sums of value functions, thus the next lemma is critical to our analysis. The proof of this lemma is provided in Appendix B.

Lemma 2 *Consider L piece-wise quadratic functions $\{f_l\}_{l=1}^L$, each with a sorted list of breakpoints Γ_l. The function $g = \sum_{l=1}^L f_l$ is a piece-wise quadratic function with breakpoints belonging to $\bigcup_{l=1}^L \Gamma_l$. Moreover, given $\{f_l\}_{l=1}^L$ and $\{\Gamma_l\}_{l=1}^L$, g can be characterized in $\mathcal{O}(LN)$ time and memory, where $N = |\bigcup_{l=1}^L \Gamma_l|$.*

In order to design an efficient algorithm, recursive equations of the form (9) need to be obtained efficiently. A property that will allow us to do so is the notion of consistency, defined next.

Definition 3 A piece-wise quadratic function f with N pieces $p_1, \ldots, p_N$ is called **consistent** if:

1. $p_1, \ldots, p_N$ are strongly convex;
2. $f(\alpha) = \min_{1 \le k \le N}\{p_k(\alpha)\}$ for all $\alpha \in \mathbb{R}$,
3. $p_k(\alpha) \ne p_{k+1}(\alpha)$ for some $\alpha \in \mathbb{R}$.

Figure 4 depicts two instances of piece-wise quadratic functions, with only one being consistent.

From recursion (9) we see that the algorithm requires the computation of conjugate functions of piece-wise quadratic functions with an indicator variable. Naturally, the overall complexity of the algorithm depends on the number of pieces required to represent the conjugate functions. The next proposition, whose proof we defer to Sect. 3.3, shows that the number of pieces can increase by at most 2.
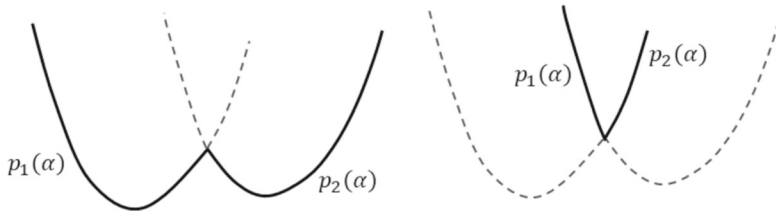
Fig. 4 Both functions are piece-wise quadratic. The left figure is consistent, while the right figure is not consistent as it violates the second condition outlined in Definition 3

Proposition 1 *Consider $f = \tilde{f} + \lambda \mathbb{1}_\alpha$, where $\lambda > 0$ and $\tilde{f}$ is consistent with N pieces. There exist an integer M and scalars $-\infty = \tau_0 < \tau_1 < \cdots < \tau_M = +\infty$[2] such that the conjugate function f^* can be written as*

$$f^*(\beta) = q_k(\beta), \quad for \quad \tau_{k-1} \leq \beta \leq \tau_k; \ k = 1, \ldots, M, \tag{10}$$

where

1. $q_1, \ldots, q_M$ are quadratic and convex;
2. $q_k(\beta) \neq q_{k+1}(\beta)$ for some β, for $k = 1, \ldots, M - 1$;
3. $f^*(\beta) = \max_{1 \leq k \leq M} \{q_k(\beta)\}$ for all $\beta \in \mathbb{R}$;
4. $M \leq N + 2$.

We note that the first three properties in Proposition 1 follow directly from the definition of consistent functions and their conjugates. The primary contribution of this proposition lies in establishing that the number of pieces of $f^*(\beta)$ is upper bounded by $N + 2$. While the second property may initially appear redundant, it is ultimately crucial for proving this result. To streamline the presentation, we include a high-level sketch of the proof here and defer the complete proof to Sect. 3.3.

At a high level, Proposition 1 follows from the geometric interpretation of conjugate functions. For simplicity, let us assume that $\lambda = 0$. Recall that for any strongly convex and quadratic function p_k, its conjugate p_k^* is likewise strongly convex and quadratic. Moreover, $-p_k^*(\beta)$ corresponds to the intercept of a tangent to p_k with slope β. For any $\beta \in \mathbb{R}$, let $I(\beta)$ denote the minimum index of the piece at which a tangent to f with slope β intersects f. The proof of the above proposition relies on two key points: (1) $f^*(\beta) = p_{I(\beta)}^*(\beta)$ for every $\beta \in \mathbb{R}$; and (2) $I(\beta)$ is a non-decreasing function of β. The first observation implies that f^* is also piece-wise quadratic. The second observation suggests that $I(\beta)$ can have at most N changes, or equivalently, f^* can possess at most N pieces (the additional two pieces in Proposition 1 arise only if $\lambda > 0$). Figure 5 depicts this intuition on a simple consistent function.

A few observations are in order based on the above proposition. First, the conjugate function f^* is not guaranteed to be consistent, even if f is consistent (a property that holds when $\lambda = 0$). Second, the number of pieces of the conjugate can, in fact, decrease. Intuitively, by computing the conjugate, we implicitly compute the closed

---

[2] We use the same notation for breakpoints as in Definition 2 to emphasize that the conjugate function f^* is indeed piece-wise quadratic.
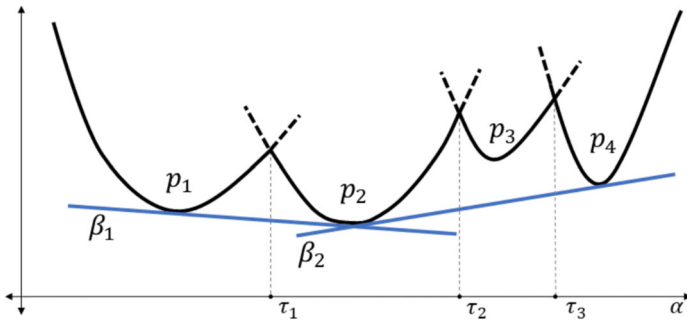
Fig. 5 A consistent function $f(\alpha)$ with four strongly convex quadratic pieces. For this function, we have $f^*(\beta) = p_1^*(\beta)$ for $\beta \leq \beta_1$, $f^*(\beta) = p_2^*(\beta)$ for every $\beta_1 \leq \beta < \beta_2$, and $f^*(\beta) = p_4^*(\beta)$ for every $\beta_2 \leq \beta$. As a result, $f^*(\beta)$ has three pieces

convex envelope of the function f, that is, we compute and only store the information relevant for optimization instead of the complete function. In general, computing convex envelopes is notoriously challenging. A common approach involves approximating them using an *outer linearization* of the form $f_Y^*(\beta) = \sup_{\alpha \in Y} \{\alpha\beta - f(\alpha)\}$, where $Y \subseteq \mathbb{R}$ (often a finite set) [4]. Clearly, $f_Y^*(\beta) \leq f^*(\beta)$ for all $\beta \in \mathbb{R}$, providing an outer approximation of the convex envelope. However, our next proposition shows that the convex envelope of consistent functions can be computed exactly and efficiently.

Proposition 2 *Given $f = \tilde{f} + \lambda \mathbb{1}_\alpha$, where $\lambda > 0$ and $\tilde{f}$ is consistent with N pieces, the conjugate function f^* can be obtained in $\mathcal{O}(N)$ time and memory.*

The proof of Proposition 2 is presented in Sect. 3.4. Equipped with these results, we are now ready to present our parametric algorithm for path graphs.

3.1 Path graphs

Assume supp(Q) is a path graph. The following lemma is a direct consequence of Propositions 1 and 2. It characterizes the parametric cost at every node u based on the parametric cost of its parent node $u - 1$.

Lemma 3 *Suppose that supp(Q) is a path graph. Moreover, given any node u, suppose that $f_{u-1} = \tilde{f}_{u-1} + \lambda_{u-1} \mathbb{1}_\alpha$, where $\tilde{f}_{u-1}$ is consistent with N pieces. Then, we can express $f_u = \tilde{f}_u + \lambda_u \mathbb{1}_\alpha$, where $\tilde{f}_u$ is consistent with at most $N + 2$ pieces. Moreover, given f_{u-1}, f_u can be found in $\mathcal{O}(N)$ time and memory.*

Proof Since $\tilde{f}_{u-1}$ is consistent with N pieces, due to Proposition 1, there exist an integer $M \leq N + 2$ and scalars $-\infty = \tau_{u-1,0} < \tau_{u-1,1} < \cdots < \tau_{u-1,M} = +\infty$ such that the conjugate function f_{u-1}^* can be written as:

$$f_{u-1}^*(\beta) = q_{u-1,k}(\beta), \quad \text{for} \quad \tau_{u-1,k-1} < \beta \leq \tau_{u-1,k}; \ k = 1, \ldots, M,$$

where

1. $q_{u-1,1}, \ldots, q_{u-1,M}$ are quadratic and convex;
2. $q_{u-1,k}(\beta) \neq q_{u-1,k+1}(\beta)$ for some β, for $k = 1, \ldots, M - 1$;
3. $f_{u-1}^*(\beta) = \max\limits_{1 \leq k \leq M} \{q_{u-1,k}(\beta)\}$ for all $\beta \in \mathbb{R}$.

Combined with (9), this implies that $f_u = \tilde{f}_u + \lambda_u \mathbb{1}_\alpha$, where

$$\tilde{f}_u(\alpha) = \underbrace{\frac{1}{2}\alpha^2 + c_u \alpha - q_{u-1,k}(-Q_{u,u-1}\alpha)}_{:=p_{u,k}(\alpha)},$$

$$\text{for} \quad \underbrace{-\frac{\tau_{u-1,k}}{Q_{u,u-1}}}_{:=\tau_{u,k-1}} < \alpha \leq \underbrace{-\frac{\tau_{u-1,k-1}}{Q_{u,u-1}}}_{:=\tau_{u,k}}; \quad k = 1, \ldots, M, \quad (11)$$

where we used $Q_{u,u-1} \neq 0$ since $\mathrm{supp}(Q)$ is assumed to be connected. Next, we establish that $\tilde{f}_u$ is indeed consistent. First, the strong convexity of $p_{u,k}$ for $k = 1, \ldots, M$ directly follows from Lemma 1. Second, we observe that $p_{u,k}(\alpha) \neq p_{u,k+1}(\alpha)$ for some α since $q_{u-1,k}(\beta) \neq q_{u-1,k+1}(\beta)$ for some β. Third, we have

$$\tilde{f}_u(\alpha) = \frac{1}{2}\alpha^2 + c_u \alpha - f_{u-1}^*(-Q_{u,u-1}\alpha)$$

$$= \frac{1}{2}\alpha^2 + c_u \alpha - \max_{1 \leq k \leq M}\{q_{u-1,k}(-Q_{u,u-1}\alpha)\}$$

$$= \min_{1 \leq k \leq M}\left\{\frac{1}{2}\alpha^2 + c_u \alpha - q_{u-1,k}(-Q_{u,u-1}\alpha)\right\}$$

$$= \min_{1 \leq k \leq M}\{p_{u,k}(\alpha)\}.$$

Finally, due to Proposition 2, f_{u-1}^* can be obtained in $\mathcal{O}(N)$ time and memory. Combined with (11), this indicates that f_u can also be computed in $\mathcal{O}(N)$ time and memory. □

Due to (9), the parametric cost f_1 at the leaf node 1 is the sum of an indicator function and a consistent function with $N = 1$ piece. Therefore, Lemma 3 implies that f_2 is the sum of an indicator function and a consistent function with $N \leq 3$ pieces. Moreover, it can be computed in $O(1)$ time. Repeating this process until reaching the root node proves that f_n can be expressed as the sum of an indicator function and a consistent function with $N \leq 2n$ pieces, and it can be computed in $\mathcal{O}(1 + 3 + 5 + \cdots + 2n) = \mathcal{O}(n^2)$. Once f_n is determined, the optimal cost $f^\star$ can be derived by minimizing f_n over at most $2n$ strongly convex and quadratic pieces. The details of this procedure are delineated in Algorithm 1.

Theorem 1 *Algorithm 1 solves Problem (5) in $\mathcal{O}(n^2)$ time and memory.*

Proof Due to (9), the parametric cost at the leaf node 1 can be written as $f_1 = \tilde{f}_1 + \lambda_1 \mathbb{1}_\alpha$, where $\tilde{f}_1$ is consistent with $N = 1$ piece. Consequently, by inductively

---

Algorithm 1 Parametric algorithm for path graphs

Input: $c, \lambda \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, where Q is positive definite and $\mathrm{supp}(Q)$ is a path graph
Output: The optimal solution $x^\star$ and optimal cost $f^\star$
1: **for** $u = 1, \ldots, n$ **do**
2:   Obtain f_u based on f_{u-1}^* via Equation (9)
3:   Obtain f_u^* based on f_u via the breakpoint algorithm (Algorithm 4)
4:   $u \leftarrow \mathrm{child}(u)$
5: **end for**
6: Obtain $f^\star = \min_\alpha f_n(\alpha)$ and $x_n^\star = \mathrm{argmin}_\alpha f_n(\alpha)$
7: **for** $u = n-1, \ldots, 1$ **do**
8:   Set $x_u^\star = \mathrm{argmin}_\alpha \{ f_u(\alpha) + Q_{u+1,u} x_{u+1}^\star \cdot \alpha \}$
9: **end for**
10: **return** $f^\star$ and $x^\star$

---

applying Lemma 3 from the leaf to the root node, the correctness of Algorithm 1 is established.

To show its runtime, we consider the operations within the loops. Since f_{u-1}^* has at most $2n$ pieces, the first operation inside the loop (Line 3) can be executed in $O(n)$ time. Moreover, the second operation inside the first loop (Line 3) can be executed in $O(N) = O(n)$ time due to Proposition 2. Hence, the first loop can be executed in $O(n^2)$ time. On the other hand, according to Lemma 3, $f_n = \tilde{f}_n + \lambda_n \mathbb{1}_\alpha$, where $\tilde{f}_n$ is consistent with at most $2n$ pieces. Therefore, Line 6 can be executed in $O(n)$ time by minimizing at most $2n$ strongly convex and quadratic functions. Similarly, each operation inside the second loop can be executed in $O(n)$ time, resulting in $O(n^2)$ time and memory for the second loop. □

3.2 Tree graphs

In this section, we extend our parametric algorithm to the general tree structures. Toward this goal, we first revisit Example 1 to elucidate the key ideas behind this extension.

Example 1 (Continued) To obtain the optimal cost, akin to the path graphs, it suffices to derive the parametric cost f_n. This can be achieved by noting that:

$$
\begin{aligned}
f_n(\alpha) &= \frac{1}{2}\alpha^2 + c_n\alpha + \lambda_n \mathbb{1}_\alpha + \sum_{v \in \mathrm{par}(n)} \min_\xi \left\{ Q_{n,v}\alpha \cdot \xi + f_v(\xi) \right\} \\
&= \frac{1}{2}\alpha^2 + c_n\alpha + \lambda_n \mathbb{1}_\alpha - \sum_{v \in \mathrm{par}(n)} \max_\xi \left\{ -Q_{n,v}\alpha \cdot \xi - f_v(\xi) \right\} \\
&= \frac{1}{2}\alpha^2 + c_n\alpha + \lambda_n \mathbb{1}_\alpha - \sum_{v \in \mathrm{par}(n)} f_v^* \left(-Q_{n,v}\alpha \right).
\end{aligned}
$$

For every $v \in \mathrm{par}(n)$, $\mathrm{supp}_v(Q)$ is a path. Therefore, according to our discussion in the previous section, each f_v^* is consistent with at most $2L + 2$ pieces, and can

be obtained in $O(L^2)$ time via Algorithm 1. On the other hand, invoking Lemma 2 implies that $\sum_{v \in \text{par}(n)} f_v^* (-Q_{n,v}\alpha)$ is a piece-wise quadratic function with at most $B(2L+2)$ pieces, and can be obtained in $\mathcal{O}(B^2 L)$. Therefore, the optimal cost $f^\star$ can be obtained by minimizing different pieces of f_n in $\mathcal{O}(BL)$. This brings the complexity of the parametric algorithm to $\mathcal{O}(BL^2 + B^2 L)$. This is a significant improvement upon the direct DP approach, which runs in $\mathcal{O}\left((L+1)^B\right)$.

Motivated by the above example, we next present the analog of Lemma 3 for tree graphs.

Lemma 4 *Suppose that* $\text{supp}(Q)$ *is a tree graph. Moreover, given any node* u, *suppose that* $f_v = \tilde{f}_v + \lambda_v \, \mathbb{1}_\alpha$ *for every* $v \in \text{par}(u)$, *where* $\tilde{f}_v$ *is consistent with* N_v *pieces. Then, we can express* $f_u = \tilde{f}_u + \lambda_u \, \mathbb{1}_\alpha$, *where* $\tilde{f}_u$ *is consistent with at most* $\sum_{v \in \text{par}(u)} (N_v + 2)$ *pieces. Moreover, given* $\{f_v\}_{v \in \text{par}(u)}$, f_u *can be found in* $\mathcal{O}\left(\deg(u)\left(\sum_{v \in \text{par}(u)} N_v\right)\right)$ *time and memory.*

Proof Since for every $v \in \text{par}(u)$, $\tilde{f}_v$ is consistent with N_v pieces, Proposition 1 implies the existence of an integer $M_v \leq N_v + 2$ and scalars $-\infty = \tau_{v,0} < \tau_{v,1} < \cdots < \tau_{v,M_v} = +\infty$ such that f_v^* can be written as:

$$f_v^*(\beta) = q_{v,k}(\beta), \quad \text{for} \quad \tau_{v,k-1} < \beta \leq \tau_{v,k}; \ k = 1, \ldots, M_v,$$

where

1. $q_{v,1}, \ldots, q_{v,M}$ are quadratic and convex;
2. $q_{v,k}(\beta) \neq q_{v,k+1}(\beta)$ for some β, for $k = 1, \ldots, M_v - 1$;
3. $f_v^*(\beta) = \max_{1 \leq k \leq M_v} \{q_{v,k}(\beta)\}$ for all $\beta \in \mathbb{R}$.

Let Γ_v be the ordered list of the breakpoints of $f_v^*(-Q_{u,v}\alpha)$ defined as $\Gamma_v = \{-\tau_{v,k}/Q_{u,v}\}_{k=1}^{M_v}$. Consider $g_u(\alpha) = \sum_{v \in \text{par}(u)} f_v^*(-Q_{u,v}\alpha)$. According to Lemma 2, g_u is piece-wise quadratic with a set of breakpoints $\bigcup_{v \in \text{par}(u)} \Gamma_v$ that has a cardinality of $N_u \leq 1 + \sum_{v \in \text{par}(u)}(N_v + 2)$. Given the ordered lists $\{\Gamma_v\}_{v \in \text{par}(v)}$, $\bigcup_{v \in \text{par}(u)} \Gamma_v$ can be ordered and stored in $\mathcal{O}\left(\sum_{v \in \text{par}(u)} N_v\right)$ time and memory. Let $-\infty = \tau_{u,0} < \tau_{u,1} < \cdots < \tau_{u,N_u} = +\infty$ be the ordered elements of $\bigcup_{v \in \text{par}(u)} \Gamma_v$. One can write

$$g_u(\alpha) = \underbrace{\sum_{v \in \text{par}(u)} q_{v,i_v(k)}(-Q_{u,v}\alpha)}_{:=\tilde{q}_{u,k}(\alpha)}, \quad \text{for} \quad \tau_{u,k-1} < \alpha \leq \tau_{u,k}; \ k = 1, \ldots, N_u,$$

(12)

where $i_v(k)$ is the index for which $[\tau_{u,k-1}, \tau_{u,k}] \subseteq \left[-\frac{\tau_{v,i_v(k)}}{Q_{u,v}}, -\frac{\tau_{v,i_v(k)-1}}{Q_{u,v}}\right]$ if $Q_{u,v} > 0$ and $[\tau_{u,k-1}, \tau_{u,k}] \subseteq \left[-\frac{\tau_{v,i_v(k)-1}}{Q_{u,v}}, -\frac{\tau_{v,i_v(k)}}{Q_{u,v}}\right]$ if $Q_{u,v} < 0$. The above equation combined

with (9) implies that $f_u = \tilde{f}_u + \lambda_u \mathbb{1}_\alpha$, where

$$\tilde{f}_u(\alpha) = \underbrace{\frac{1}{2}\alpha^2 + c_u\alpha - \tilde{q}_{u,k}(\alpha)}_{:=p_{u,k}(\alpha)}, \quad \text{for} \quad \tau_{u,k-1} < \alpha \leq \tau_{u,k}; \ k = 1, \ldots, N_u. \quad (13)$$

Next, we prove that $\tilde{f}_u$ is consistent. First, if $p_{u,k}$ and $p_{u,k+1}$ are identical for some $1 \leq k \leq N_u$, one can remove the $(k+1)$-th piece and set $\tau_{u,k} \leftarrow \tau_{u,k+1}$ and $N_u \leftarrow N_u - 1$. This process can be repeated until $p_{u,k}$ and $p_{u,k+1}$ are not identical for all $1 \leq k \leq N_u - 1$. Second, the strong convexity of $p_{u,k}$ for $k = 1, \ldots, N_u$ directly follows from Lemma 1. Third, note that

$$\begin{aligned}
g_u(\alpha) &= \sum_{v \in \text{par}(u)} f_v^*(-Q_{u,v}\alpha) \\
&= \sum_{v \in \text{par}(u)} \max_{1 \leq k \leq N_v} \left\{ q_{v,k}(-Q_{u,v}\alpha) \right\} \\
&\geq \max_{1 \leq k \leq N_u} \left\{ \sum_{v \in \text{par}(u)} q_{v,i_v(k)}(-Q_{u,v}\alpha) \right\} \\
&= \max_{1 \leq k \leq N_u} \left\{ \tilde{q}_{u,k}(\alpha) \right\} \\
&\geq g_u(\alpha),
\end{aligned}$$

where the last inequality follows from (12). Therefore, we have $g_u(\alpha) = \max_{1 \leq k \leq N_u} \left\{ \tilde{q}_{u,k}(\alpha) \right\}$. This leads to

$$\begin{aligned}
\tilde{f}_u(\alpha) &= \frac{1}{2}\alpha^2 + c_u\alpha - g_u(\alpha) \\
&= \frac{1}{2}\alpha^2 + c_u\alpha - \max_{1 \leq k \leq N_u} \left\{ \tilde{q}_{u,k}(\alpha) \right\} \\
&= \min_{1 \leq k \leq N_u} \left\{ \frac{1}{2}\alpha^2 + c_u\alpha - \tilde{q}_{u,k}(\alpha) \right\} \\
&= \min_{1 \leq k \leq M} \left\{ p_{u,k}(\alpha) \right\}.
\end{aligned}$$

This completes the proof of the consistency of $\tilde{f}_u$. Finally, due to Proposition 2, each f_v^* can be obtained in $\mathcal{O}(N_v)$ time and memory. Therefore, $g_u(\alpha) = \sum_{v \in \text{par}(u)} f_v^*(-Q_{u,v}\alpha)$ can be obtained in $\mathcal{O}\left(\deg(u) \left(\sum_{v \in \text{par}(u)} N_v \right) \right)$ time and memory, according to Lemma 2. Combined with $f_u(\alpha) = (1/2)\alpha^2 + c_u\alpha + \lambda_u \mathbb{1}_\alpha - g_u(\alpha)$, this indicates that f_u can also be computed in $\mathcal{O}\left(\deg(u) \left(\sum_{v \in \text{par}(u)} N_v \right) \right)$ time and memory. $\qquad \square$

With Lemma 4 in place, we are prepared to present an overview of our parametric algorithm for general tree graphs. The algorithm starts with node 1. Since node 1 represents a leaf node, its parametric cost f_1 can be readily determined based on the recursion (9). Moreover, its conjugate f_1^* can be obtained in $\mathcal{O}(1)$ due to Proposition 2. Assuming that the parametric costs f_v and their conjugates f_v^* are available for every node $v < u$, the parametric cost f_u, can be obtained based on Lemma 4. Notably, due to the topological ordering of nodes, all $v \in \text{par}(u)$ satisfy $v < u$, ensuring that their conjugate parametric costs f_v^* needed to characterize f_u are known. By repeating this process iteratively, the algorithm efficiently computes the parametric costs following the increasing topological ordering.

Algorithm 2 formalizes the aforementioned intuition and presents the proposed parametric algorithm for trees with greater detail.

Algorithm 2 Parametric algorithm over general trees

Input: $c, \lambda \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, where Q is positive definite and $\text{supp}(Q)$ has a tree structure
Output: The optimal solution $x^\star$ and optimal cost $f^\star$
1: Label the nodes $\text{supp}(Q)$ according to their topological ordering
2: **for** $u = 1, \dots, n$ **do**
3:     Obtain f_u based on $\{f_v^*\}_{v \in \text{par}(u)}$ via Equation (9)
4:     Obtain f_u^* from f_u via Algorithm 4
5: **end for**
6: Obtain $f^\star = \min_\alpha f_n(\alpha)$ and $x_n^\star = \underset{\alpha}{\text{argmin}}\, f_n(\alpha)$
7: $J \leftarrow \text{par}(n)$
8: **while** $J \neq \{\}$ **do**
9:     Choose $u \in J$
10:     Set $x_u^\star = \underset{\alpha}{\text{argmin}}\{f_u(\alpha) + Q_{\text{child}(u),u} x_{\text{child}(u)}^\star \alpha\}$
11:     $J \leftarrow J \setminus \{u\}$
12:     $J \leftarrow J \cup \text{par}(u)$
13: **end while**
14: **return** $f^\star$ and $x^\star$

Theorem 2 *Under the assumption that* $\text{supp}(Q)$ *is a tree, Algorithm 2 solves Problem (1) in* $\mathcal{O}(n^2)$ *time and memory.*

Proof The proof is analogous to that of Theorem 1, and proceeds inductively using Equation (9) and Lemma 4. For brevity, we omit the specific details. □

3.3 Properties of consistent functions

In this section, we present the proof of Proposition 1. To this goal, we first introduce the fundamental properties of consistent functions and their conjugates.

For a piece-wise quadratic function g with N strongly convex pieces $p_1, \dots, p_N$, we define its *indexing function* $I_g : \mathbb{R} \to \{1, \dots, N\}$ as:

$$I_g(\beta) = \min\left\{k : \tau_{k-1} \leq \alpha^\star \leq \tau_k, \alpha^\star \in \underset{\alpha}{\text{argmax}}\,\{\beta\alpha - g(\alpha)\}\right\}, \qquad (14)$$

Fig. 6 A consistent function $f(\alpha)$ with three strongly convex quadratic pieces. In this example, p_1 and p_3 are defined from the same quadratic equation
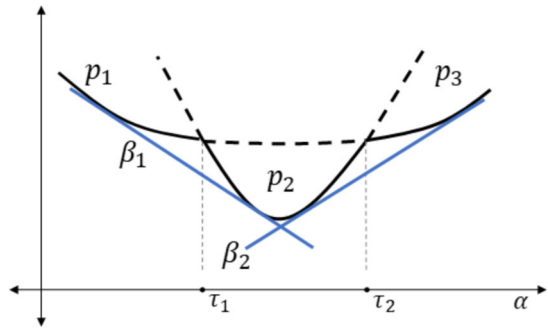
where $\{\tau_k\}_{k=0}^N$ are the breakpoints of g. Intuitively, the indexing function I_g returns the piece with the minimum index where a line with slope β is tangent to g. As an example, the indexing function for f depicted in Fig. 5 can be characterized as $I_f(\beta) = 1$ for all $\beta \leq \beta_1$, $I_f(\beta) = 2$ for all $\beta_1 < \beta \leq \beta_2$, and $I_f(\beta) = 4$ for all $\beta_2 < \beta$. As another example, consider the indexing function for f depicted in Fig. 6. Here, $I_f(\beta) = 1$ for all $\beta \leq \beta_1$, $I_f(\beta) = 2$ for all $\beta_1 < \beta \leq \beta_2$, and $I_f(\beta) = 3$ for all $\beta_2 < \beta$.

Due to the definition of the indexing function, there exists a solution $\alpha^\star \in \operatorname{argmax}_\alpha \{\beta\alpha - g(\alpha)\}$ such that $\tau_{I_g(\beta)-1} \leq \alpha^\star \leq \tau_{I_g(\beta)}$. Therefore, we have

$$g^*(\beta) = \beta\alpha^\star - g(\alpha^\star) = \beta\alpha^\star - p_{I_g(\beta)}(\alpha^\star) = \max_\alpha \{\beta\alpha - p_{I_g(\beta)}(\alpha)\} = p_{I_g(\beta)}^*(\beta). \tag{15}$$

Let the image of I_g be denoted as $\operatorname{Image}(I_g) = \{k : k = I_g(\beta) \text{ for some } \beta \in \mathbb{R}\}$. For every $k \in \operatorname{Image}(I_g)$, its inverse image is defined as $I_g^{-1}(k) = \{\beta : I_g(\beta) = k\}$. Revisiting Fig. 5, the indexing function of f satisfies $\operatorname{Image}(I_f) = \{1, 2, 4\}$ with inverse images $I_f^{-1}(1) = (-\infty, \beta_1]$, $I_f^{-1}(2) = (\beta_1, \beta_2]$, and $I_f^{-1}(4) = (\beta_2, +\infty)$.

Recall the intuition behind Proposition 1: In order to control the number of pieces of g^*, it suffices to control the number of changes in the indexing function I_g. This can be achieved by showing that I_g is non-decreasing. Our next lemma establishes this important property for consistent functions.

Lemma 5 *Any consistent function has a non-decreasing indexing function.*

To prove the above lemma, we first present the following intermediate result.

Lemma 6 *Suppose that g is consistent with pieces $p_1, \ldots, p_N$ and breakpoints $-\infty = \tau_0 < \tau_1 < \cdots < \tau_N = +\infty$. For any $\beta \in \mathbb{R}$ and $k \in \{1, \ldots, N\}$, define the linear function $\ell_{k;\beta}(\alpha) = \beta\alpha - p_k^*(\beta)$. Moreover, define $\alpha^\star(\beta) \in \operatorname{argmax}_\alpha \{\beta\alpha - g(\alpha)\}$. Let k^* be such that $\tau_{k^*-1} \leq \alpha^\star(\beta) \leq \tau_{k^*}$. The following statements hold:*

1. *We have $\alpha^\star(\beta) \notin \{\tau_{k^*-1}, \tau_{k^*}\}$.*
2. *We have $\ell_{k^*;\beta}(\alpha^\star(\beta)) = g(\alpha^\star(\beta))$, and $\ell_{k^*;\beta}(\alpha) \leq g(\alpha)$ for every $\alpha \in \mathbb{R}$.*

Proof To prove the first statement, suppose, by contradiction, that $\alpha^\star(\beta) = \tau_k$ for some $k \in \{k^* - 1, k^*\}$. Note that $g(\alpha) - \beta\alpha = p_k(\alpha) - \beta\alpha$ for every $\tau_{k-1} \leq \alpha \leq \tau_k$

and $g(\alpha) - \beta\alpha = p_{k+1}(\alpha) - \beta\alpha$ for every $\tau_k \leq \alpha \leq \tau_{k+1}$. Since $\alpha^\star(\beta) = \tau_k$, we must have $p'_k(\alpha^\star(\beta)) \leq 0$ and $p'_{k+1}(\alpha^\star(\beta)) \geq 0$. Since $\beta\alpha - g(\alpha)$ is a continuous function of α, we must have $p_k(\alpha^\star(\beta)) = p_{k+1}(\alpha^\star(\beta))$. We consider three cases:

1. Suppose $p'_k(\alpha^\star(\beta)) = p'_{k+1}(\alpha^\star(\beta)) = 0$. Since p_k and p_{k+1} are not identical, we must have $p''_k(\alpha^\star(\beta)) \neq p''_{k+1}(\alpha^\star(\beta))$. If $p''_k(\alpha^\star(\beta)) < p''_{k+1}(\alpha^\star(\beta))$, then $p_k(\alpha^\star(\beta) + \epsilon) < p_{k+1}(\alpha^\star(\beta) + \epsilon)$ for every $\epsilon > 0$, which is a contradiction. Similarly, if $p''_k(\alpha^\star(\beta)) > p''_{k+1}(\alpha^\star(\beta))$, then $p_k(\alpha^\star(\beta) - \epsilon) > p_{k+1}(\alpha^\star(\beta) - \epsilon)$ for every $\epsilon > 0$, which is again a contradiction.
2. Suppose $p'_k(\alpha^\star(\beta)) < 0$. Therefore, there exists $\bar\epsilon > 0$ such that, for every $\epsilon \in (0, \bar\epsilon]$, we have

$$p_k(\alpha^\star(\beta) + \epsilon) < p_k(\alpha^\star(\beta)) = p_{k+1}(\alpha^\star(\beta)) \leq p_{k+1}(\alpha^\star(\beta) + \epsilon),$$

   which is a contradiction.
3. Suppose $p'_{k+1}(\alpha^\star(\beta)) > 0$. Following the same reasoning as the previous case, we arrive at a contradiction.

To prove the second statement, recall that $\beta\alpha - g(\alpha) = \beta\alpha - p_{k^*}(\alpha)$ for every $\tau_{k^*-1} \leq \alpha \leq \tau_{k^*}$. Therefore, since $\alpha^\star(\beta) \in \text{argmax}_\alpha\{\beta\alpha - g(\alpha)\}$ and $\alpha^\star(\beta) \in (\tau_{k^*-1}, \tau_{k^*})$, we must have $\alpha^\star(\beta) \in \text{argmin}_{\alpha \in (\tau_{k^*-1}, \tau_{k^*})}\{p_{k^*}(\alpha) - \beta\alpha\}$. Since $p_{k^*}(\alpha) - \beta\alpha$ is a strongly convex function of α, this implies that $\alpha^\star(\beta) = \text{argmin}_\alpha\{p_{k^*}(\alpha) - \beta\alpha\} = \text{argmax}_\alpha\{\beta\alpha - p_{k^*}(\alpha)\}$. Therefore,

$$\beta\alpha^\star(\beta) - p_{k^*}(\alpha^\star(\beta)) = p^*_{k^*}(\beta)$$
$$\iff \beta\alpha^\star(\beta) - p^*_{k^*}(\beta) = p_{k^*}(\alpha^\star(\beta))$$
$$\iff \ell_{k^*;\beta}(\alpha^\star(\beta)) = g(\alpha^\star(\beta)).$$

Finally, since $\max_\alpha\{\beta\alpha - g(\alpha)\} = \max_\alpha\{\beta\alpha - p_{k^*}(\alpha)\}$, one can write

$$
\begin{aligned}
\ell_{k^*;\beta}(\alpha) &= \beta\alpha - p^*_{k^*}(\beta) \\
&= \beta\alpha - \max_\xi\{\beta\xi - p_{k^*}(\xi)\} \\
&= \beta\alpha + \min_\xi\{-\beta\xi + p_{k^*}(\xi)\} \\
&= \beta\alpha + \min_\xi\{-\beta\xi + g(\xi)\} \\
&= \min_\xi\{\beta\alpha - \beta\xi + g(\xi)\} \\
&\leq \beta\alpha - \beta\alpha + g(\alpha) \\
&= g(\alpha).
\end{aligned}
$$

This completes the proof. □

Proof of Lemma 5. Suppose that g is consistent with pieces $p_1, \ldots, p_N$ and breakpoints $-\infty = \tau_0 < \tau_1 < \cdots < \tau_N = +\infty$. To show $I_g(\beta)$ is non-decreasing, it suffices to show that if $k < I_g(\beta)$ for some $\beta \in \mathbb{R}$, then $k \neq I_g(\beta')$, for any $\beta' > \beta$.

By contradiction, suppose there exist $\beta < \beta'$ such that $k < I_g(\beta)$ and $k = I_g(\beta')$. Let $l = I_g(\beta)$. Due to the definition of the indexing function, there exist $\alpha_l^\star, \alpha_k^\star \in \mathbb{R}$ such that

$$\alpha_l^\star \in \text{argmax}_\alpha\{\beta\alpha - g(\alpha)\}, \text{ and } \tau_{l-1} \leq \alpha_l^\star \leq \tau_l,$$

$$\alpha_k^\star \in \text{argmax}_\alpha\{\beta'\alpha - g(\alpha)\}, \text{ and } \tau_{k-1} \leq \alpha_k^\star \leq \tau_k.$$

Due to the first statement of Lemma 6, we must have $\tau_{l-1} < \alpha_l^\star < \tau_l$ and $\tau_{k-1} < \alpha_k^\star < \tau_k$. This implies that

$$\alpha_k^\star < \tau_k \leq \tau_{l-1} < \alpha_l^\star \implies \alpha_k^\star < \alpha_l^\star. \tag{16}$$

On the other hand, the second statement of Lemma 6 implies that

$$\ell_{l;\beta}(\alpha_l^\star) = g(\alpha_l^\star) \text{ and } \ell_{l;\beta}(\alpha) \leq g(\alpha); \forall \alpha$$
$$\ell_{k;\beta'}(\alpha_k^\star) = g(\alpha_k^\star) \text{ and } \ell_{k;\beta'}(\alpha) \leq g(\alpha); \forall \alpha.$$

Combining the above two inequalities, we have

$$\ell_{l;\beta}(\alpha_k^\star) \leq g(\alpha_k^\star) = \ell_{k;\beta'}(\alpha_k^\star) \implies \beta\alpha_k^\star - p_l^*(\beta) \leq \beta'\alpha_k^\star - p_k^*(\beta')$$
$$\ell_{k;\beta'}(\alpha_l^\star) \leq g(\alpha_l^\star) = \ell_{l;\beta}(\alpha_l^\star) \implies \beta'\alpha_l^\star - p_k^*(\beta') \leq \beta\alpha_l^\star - p_l^*(\beta).$$

The above two inequalities yield

$$\beta\alpha_k^\star - p_l^*(\beta) + \beta'\alpha_l^\star - p_k^*(\beta') \leq \beta'\alpha_k^\star - p_k^*(\beta') + \beta\alpha_l^\star - p_l^*(\beta)$$
$$\Longleftrightarrow (\beta' - \beta)\alpha_l^\star \leq (\beta' - \beta)\alpha_k^\star$$
$$\Longleftrightarrow \alpha_l^\star \leq \alpha_k^\star,$$

which contradicts (16). This completes the proof. $\qquad\square$

Our next lemma provides a key property of the conjugate of a piece-wise quadratic function with a non-decreasing indexing function.

Lemma 7 *Suppose that g is a piece-wise quadratic function with N strongly convex pieces $p_1, \ldots, p_N$ and a non-decreasing indexing function I_g. There exist an integer $N' \leq N$, scalars $-\infty = \tau_0 < \tau_1 < \cdots < \tau_{N'} = +\infty$, and a strictly increasing function $\pi : \{1, \ldots, N'\} \to \{1, \ldots, N\}$ such that*

$$g^*(\beta) = p_{\pi(k)}^*(\beta), \quad \text{for } \tau_{k-1} \leq \beta \leq \tau_k; \ k = 1, \ldots, N', \tag{17}$$

where

1. *$p_1^*, \ldots, p_N^*$ are quadratic and strongly convex;*
2. *$p_{\pi(k)}^*(\beta) \neq p_{\pi(k+1)}^*(\beta)$ for some β, for $k = 1, \ldots, N' - 1$.*

Proof Let $j_1 < j_2 < \cdots < j_{N'}$ be the ordered elements of Image(I_g). We have $N' \leq N$ since Image(I_g) $\subseteq \{1, \ldots, N\}$. Moreover, we have $\bigcup_{k=1}^{N'} I_g^{-1}(j_k) = \mathbb{R}$. Since I_g is assumed to be non-decreasing, $I_g^{-1}(j_k)$ is a convex set for every $k = 1, \ldots, N'$. Therefore, there exist $-\infty = \tau_0 < \tau_1 \leq \cdots \leq \tau_{N'-1} < \tau_{N'} = +\infty$ such that, for every $k = 1, \ldots, N'$, $I_g^{-1}(j_k)$ can be characterized as:

$$I_g^{-1}(j_k) = [\tau_{k-1}, \tau_k], \ (\tau_{k-1}, \tau_k), \ [\tau_{k-1}, \tau_k), \ \text{or} \ (\tau_{k-1}, \tau_k]. \tag{18}$$

Upon defining $\pi(k) = j_k$ for every $1 \leq k \leq N'$, we have

$$g^*(\beta) = p_{j_k}^*(\beta) \qquad \text{if} \qquad \beta \in I_g^{-1}(j_k)$$
$$\Longleftrightarrow g^*(\beta) = p_{\pi(k)}^*(\beta) \qquad \text{if} \qquad \beta \in I_g^{-1}(j_k)$$
$$\Longleftrightarrow g^*(\beta) = p_{\pi(k)}^*(\beta) \qquad \text{if} \qquad \tau_{k-1} \leq \beta \leq \tau_k; k = 1, \ldots, N',$$

where the first equality is a direct consequence of (15), the second equality is due to the definition of the function π, and the third equality is due to (18) and the fact that g^* is continuous. This completes the proof of (17). Next, we proceed to prove the properties delineated in Lemma 7. To prove the first property, recall that p_k is strongly convex and quadratic for every $k = 1, \ldots, N$. Therefore, $p_k^*(\beta) = \max_\alpha \{\beta\alpha - p_k(\alpha)\}\}$ is also strongly convex and quadratic. Moreover, the second property follows since, if $p_{\pi(k)}^*$ and $p_{\pi(k+1)}^*$ are identical for some $1 \leq k \leq N'$, one can remove the $(k+1)$-th piece and set $\tau_k \leftarrow \tau_{k+1}$ and $N' \leftarrow N' - 1$. This process can be repeated until $p_{\pi(k)}^*(\beta)$ and $p_{\pi(k+1)}^*(\beta)$ are not identical for all $k = 1, \ldots, N' - 1$. □

We are now ready to present the proof of Proposition 1.

Proof of Proposition 1 One can write

$$f^*(\beta) = \max_\alpha \{\beta\alpha - f(\alpha)\}$$
$$= \max \left\{ -f(0), \max_{\alpha \neq 0} \left\{ \beta\alpha - \tilde{f}(\alpha) - \lambda \right\} \right\}$$
$$= \max \left\{ -f(0), \underbrace{\max_\alpha \left\{ \beta\alpha - \tilde{f}(\alpha) \right\}}_{:= \tilde{f}^*(\beta)} - \lambda \right\}. \tag{19}$$

Next, note that

$$\tilde{f}^*(\beta) = \max_\alpha \left\{ \beta\alpha - \tilde{f}(\alpha) \right\}$$
$$= \max_\alpha \left\{ \beta\alpha - \min_{1 \leq k \leq N} \{p_k(\alpha)\} \right\}$$
$$= \max_\alpha \left\{ \max_{1 \leq k \leq N} \{\beta\alpha - p_k(\alpha)\} \right\}$$

$$= \max_{1 \leq k \leq N} \left\{ \max_{\alpha} \{ \beta\alpha - p_k(\alpha) \} \right\}$$
$$= \max_{1 \leq k \leq N} \{ p_k^*(\beta) \}. \tag{20}$$

Since each p_k^* is strongly convex, $\tilde{f}^*$ is also strongly convex. Therefore, the equation $\tilde{f}^*(\beta) - \lambda = -f(0)$ can have at most two solutions. Moreover, $\tilde{f}^*(0) = -f(0)$ which implies $\tilde{f}^*(0) - \lambda < -f(0)$. Hence, $\tilde{f}^*(\beta) - \lambda = -f(0)$ has exactly two solutions. Let $\beta_1 < \beta_2$ be these solutions. Based on (19), f^* can be characterized as

$$f^*(\beta) = \begin{cases} -f(0) & \beta_1 \leq \beta \leq \beta_2 \\ \tilde{f}^*(\beta) - \lambda & \text{otherwise.} \end{cases} \tag{21}$$

Since $\tilde{f}$ is consistent, it must have a non-decreasing indexing function due to Lemma 5. Combined with Lemma 7, this implies that $\tilde{f}$ has at most $N' \leq N$ pieces. Therefore, f^* emerges as a piece-wise quadratic function with at most M pieces, where $M \leq N' + 2 \leq N + 2$. Let these pieces be denoted as $\{q_k\}_{k=1}^M$. For every $1 \leq k \leq M$, we either have $q_k(\beta) = p_{k'}^*(\beta) - \lambda$ for some $1 \leq k' \leq N$, or $q_k(\beta) = -f(0)$. Therefore, $q_1, \ldots, q_M$ are quadratic and convex. Moreover, it is easy to verify that q_k and q_{k+1} are not identical for all $k = 1, \ldots, M - 1$. Finally, note that

$$\tilde{f}^*(\beta) = \max_{1 \leq k \leq N} \{ p_k^*(\beta) \} \geq \max_{1 \leq k \leq N'} \{ p_{\pi(k)}^*(\beta) \} \geq p_{I_{\tilde{f}}(\beta)}^*(\beta) = \tilde{f}^*(\beta),$$

where the first equality follows from (20). The above inequality implies that $\tilde{f}^*(\beta) = \max_{1 \leq k \leq N'} \{ p_{\pi(k)}^*(\beta) \}$. Therefore, according to (19), we have

$$f^*(\beta) = \max\{ -f(0), \tilde{f}^*(\beta) - \lambda \} = \max_{1 \leq k \leq M} \{ q_k(\beta) \}.$$

This completes the proof. □

### 3.4 Breakpoint algorithm

Our next goal is to characterize f^* efficiently. Indeed, the function f^* can be expressed as

$$f^*(\beta) = \max\{ -f(0), \tilde{f}^*(\beta) - \lambda \} = \max \left\{ \underbrace{-f(0)}_{\tilde{p}_0(\beta)}, \max_{1 \leq k \leq N} \{ \underbrace{p_k^*(\beta) - \lambda}_{\tilde{p}_k(\beta)} \} \right\}$$
$$= \max_{0 \leq k \leq N} \{ \tilde{p}_k(\beta) \}.$$

A direct method for characterizing f^* is to identify the intersections of $\tilde{p}_k$ and $\tilde{p}_l$ for all possible pairs $0 \leq k < l \leq N$, sort these intersections, and then determine the minimum piece within every pair of adjacent intersections. This method correctly

Fig. 7 A semi-consistent function with two pieces. The function is not consistent since it violates the second property of Definition 3
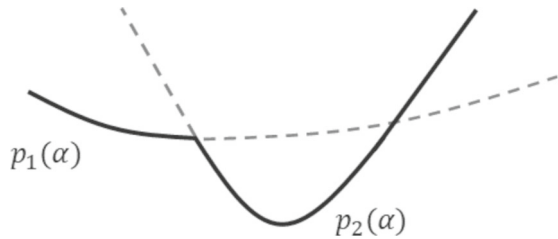
characterizes f^* and operates in $\mathcal{O}(N^2)$. However, we demonstrate that this complexity can be improved to $\mathcal{O}(N)$. To explain our method, we start by introducing the class of *semi-consistent* functions.

Definition 4 A piece-wise quadratic function g with breakpoints $-\infty = \tau_0 < \tau_1 < \cdots < \tau_N = +\infty$ and pieces $p_1, \ldots, p_N$ is called **semi-consistent** if it satisfies the following properties:

- $p_1, \ldots, p_N$ are strongly convex;
- We have $p_k(\alpha) \leq \min\{p_{k-1}(\alpha), p_{k+1}(\alpha)\}$ for all $\alpha \in [\tau_{k-1}, \tau_k]$ and $2 \leq k \leq N - 1$.
- For all $k \leq N$, the indexing function I_{g_k} is non-decreasing, where $g_k : \mathbb{R} \to \mathbb{R}$ is defined as:

$$g_k(\alpha) = \begin{cases} g(\alpha) & \alpha \leq \tau_k \\ p_k(\alpha) & \alpha > \tau_k. \end{cases} \tag{22}$$

The first property mirrors that of consistent functions. The second property is a local variant of the second property of the consistent functions: Within the local interval bounded by two adjacent breakpoints τ_{k-1} and τ_k, the function g is the minimum of the adjacent pieces p_{k-1}, p_k, and p_{k+1}. Moreover, the function g_k is obtained by restricting the function g to its first k pieces, with the final piece extended to $+\infty$. Indeed, g_k is piece-wise quadratic with k strongly convex pieces. However, it may not be consistent. It is also evident that $g_N = g$.

Not every semi-consistent function is consistent. An example is depicted in Fig. 7. However, our next lemma shows that every consistent function is semi-consistent.

Lemma 8 *Any consistent quadratic function is semi-consistent.*

Proof Suppose that g is consistent with N pieces. The first property of semi-consistent functions is trivially satisfied for g. Since $g(\alpha) = \min_{1 \leq k \leq N}\{p_k(\alpha)\}$, the function g also satisfies the second property. To prove the last property, we can follow the same steps as the proof of Lemma 5. The first step is to show that Lemma 6 holds for g_k. The second step is to prove the non-decreasing property of I_{g_k} based on the statements of Lemma 6. The details of the proof are omitted since they are identical to those of Lemma 5. □

Since every consistent function is semi-consistent, to prove Proposition 2, it suffices to provide an efficient algorithm for obtaining the conjugate of the functions expressed as $g + \lambda \mathbb{1}_\alpha$, where g is semi-consistent.

Recall the geometric interpretation of a conjugate function: Given any convex function p_k, the negative of its conjugate $-p_k^*$ is the intercept of a tangent to p_k with slope β.

Definition 5 For any $1 \leq k < l \leq N$, we define a feasible common tangent s_{kl} to pieces l and k as the slope of a line that is tangent to p_k and p_l at some points $\tau_{k-1} \leq \alpha_k \leq \tau_k$ and $\tau_{l-1} \leq \alpha_l \leq \tau_l$, respectively.

Observe that since functions p_k and p_l are strictly convex, any tangent line is an underestimator of the function. Moreover, any two different lines in $\mathbb{R}^2$ intersect in at most one point. If the intersection occurs in interval $[\tau_{k-1}, \tau_k]$, then one line is strictly "above" the other in interval $[\tau_{l-1}, \tau_l]$ and they cannot both be tangents of p_l. Cases where the intersection occurs in a different interval or where the lines are parallel can be handled identically. We formally prove this result next.

Lemma 9 *For any $1 \leq k < l \leq N$, the pieces k and l can have at most one feasible common tangent.*

Proof Since a feasible common tangent s_{kl} must satisfy $-p_l^*(s_{kl}) = -p_k^*(s_{kl})$ and both p_l^* and p_k^* are quadratic, the pieces p_l and p_k can have at most two feasible common tangents. By contradiction, suppose they have exactly two common tangents, given by $\ell_{kl}^1(\alpha) := s_{kl}^1\alpha + b_{kl}^1$ and $\ell_{kl}^2(\alpha) := s_{kl}^2\alpha + b_{kl}^2$. Without loss of generality, suppose $s_{kl}^2 > s_{kl}^1$. Let α_k^1 and α_k^2 be the points at which the lines $\ell_{kl}^1(\alpha)$ and $\ell_{kl}^2(\alpha)$ are tangent to p_k, respectively. Define α_l^1 and α_l^2 in a similar fashion. Since p_k and p_l are strongly convex and $s_{kl}^1 < s_{kl}^2$, we must have $\alpha_k^1 < \alpha_k^2$ and $\alpha_l^1 < \alpha_l^2$. Therefore,

$$\tau_{k-1} \leq \alpha_k^1 < \alpha_k^2 \leq \tau_k \leq \tau_{l-1} \leq \alpha_l^1 < \alpha_l^2 \leq \tau_l \implies \alpha_k^2 \leq \alpha_l^1. \tag{23}$$

On the other hand, due to the strong convexity of p_k, we have

$$\ell_{kl}^1(\alpha_k^1) = p_k(\alpha_k^1) \quad \text{and} \quad \ell_{kl}^1(\alpha) < p_k(\alpha); \forall \alpha \neq \alpha_k^1$$
$$\ell_{kl}^2(\alpha_k^2) = p_k(\alpha_k^2) \quad \text{and} \quad \ell_{kl}^2(\alpha) < p_k(\alpha); \forall \alpha \neq \alpha_k^2.$$

Combining the above two inequalities, we have $\ell_{kl}^1(\alpha_k^2) < \ell_{kl}^2(\alpha_k^2)$, which implies that $b_{kl}^1 - b_{kl}^2 < (s_{kl}^2 - s_{kl}^1)\alpha_k^2$. Similarly, one can show that $\ell_{kl}^2(\alpha_l^1) < \ell_{kl}^1(\alpha_l^1)$, which implies that $(s_{kl}^2 - s_{kl}^1)\alpha_l^1 < b_{kl}^1 - b_{kl}^2$. Therefore, we have

$$\begin{cases} b_{kl}^1 - b_{kl}^2 < (s_{kl}^2 - s_{kl}^1)\alpha_k^2 \\ (s_{kl}^2 - s_{kl}^1)\alpha_l^1 < b_{kl}^1 - b_{kl}^2 \end{cases} \implies \alpha_l^1 < \alpha_k^2.$$

This contradicts (23), thereby completing the proof. □

Our next algorithm (Algorithm 3) obtains the value of s_{kl}.

A few observations are in order regarding Algorithm 3. First, note that the conjugate functions p_k^* and p_l^* in Line 1 can be obtained in $\mathcal{O}(1)$ time and memory. Moreover, without loss of generality, we assume that $-p_k^*(\beta) = -p_l^*(\beta)$ has two roots β_{kl}^1 and

---

Algorithm 3 Feasible common tangent: $\text{SLOPE}(p_k, \tau_{k-1}, \tau_k, p_l, \tau_{l-1}, \tau_l)$

Input: $\{p_k, \tau_{k-1}, \tau_k\}$ and $\{p_l, \tau_{l-1}, \tau_l\}$
Output: The slope of the feasible common tangent s_{kl}
1: Obtain the conjugate functions p_k^* and p_l^*
2: Obtain the roots β_{kl}^1 and β_{kl}^2 of $-p_k^*(\beta) = -p_l^*(\beta)$
3: Obtain $\alpha_k^1 = \text{argmax}_\alpha \{\beta_{kl}^1 \alpha - p_k(\alpha)\}$ and $\alpha_l^1 = \text{argmax}_\alpha \{\beta_{kl}^1 \alpha - p_l(\alpha)\}$
4: Obtain $\alpha_k^2 = \text{argmax}_\alpha \{\beta_{kl}^2 \alpha - p_k(\alpha)\}$ and $\alpha_l^2 = \text{argmax}_\alpha \{\beta_{kl}^2 \alpha - p_l(\alpha)\}$
5: **if** $\alpha_k^1 \in [\tau_{k-1}, \tau_k]$ and $\alpha_l^1 \in [\tau_{l-1}, \tau_l]$ **then**
6:     **return** $s_{kl} = \beta_{kl}^1$
7: **else if** $\alpha_k^2 \in [\tau_{k-1}, \tau_k]$ and $\alpha_l^2 \in [\tau_{l-1}, \tau_l]$ **then**
8:     **return** $s_{kl} = \beta_{kl}^2$
9: **else if** $\alpha_l^1 \notin [\tau_{l-1}, \tau_l]$ and $\alpha_l^2 \notin [\tau_{l-1}, \tau_l]$ **then**
10:     **return** $s_{kl} = +\infty$
11: **else**
12:     **return** $s_{kl} = -\infty$
13: **end if**

β_{kl}^2; indeed, the later steps of the algorithm can be modified accordingly if $-p_k^*(\beta) = -p_l^*(\beta)$ has fewer than two roots. It is also easy to see that $\{\alpha_k^1, \alpha_l^1, \alpha_k^2, \alpha_l^2\}$ in Lines 3 and 4 can be obtained in $\mathcal{O}(1)$ time and memory. Finally, the algorithm assigns $+\infty$ or $-\infty$ to s_{kl} if a feasible common tangent does not exist.

We next show that the breakpoints of g^* coincide with certain feasible common tangents that satisfy a breakpoint condition.

**Definition 6** We say pieces $k < l$ satisfy the breakpoint condition if:

- $-\infty < s_{kl} < +\infty$;
- $I_g^-(s_{kl}) = \lim_{\epsilon \to 0^+} I_g(s_{kl} - \epsilon) = I_g(s_{kl}) = k$;
- $I_g^+(s_{kl}) = \lim_{\epsilon \to 0^+} I_g(s_{kl} + \epsilon) = l$.

We refer the reader back to Fig. 5 for intuition. Both lines with slopes β_1 and β_2 are tangent to pieces satisfying the breakpoint condition. Alternatively, imagine the line tangent to pieces p_2 and p_3. Such a line would cut into the epigraph of piece p_4. In this scenario, $I_g^+(s_{23}) = 4$, violating the last condition. Intuitively, tangent lines between pieces satisfying the breakpoint condition are the lines required to describe the convex envelope of the piece-wise quadratic function g. More formally, as we show next, the slopes of such lines are required to describe the conjugate function.

Lemma 10 *The pieces $k < l$ satisfy the breakpoint condition if and only if their feasible common tangent s_{kl} is a breakpoint for g^*.*

Proof Suppose that the pieces $k < l$ satisfy the breakpoint condition. Therefore, we have $I_g^-(s_{kl}) = I_g(s_{kl}) = k$, which implies that there exists some $\bar{\epsilon} > 0$ such that for all $\epsilon \in [0, \bar{\epsilon})$ we have $I_g(s_{kl} - \epsilon) = k$. From the definition of the indexing function, it follows that there exists $\alpha_k^\star \in \text{argmax}\{(s_{kl} - \epsilon)\alpha - g(\alpha)\}$ such that $\tau_{k-1} \leq \alpha_k^\star \leq \tau_k$. Therefore, we have

$$g^*(s_{kl} - \epsilon) = (s_{kl} - \epsilon)\alpha_k^\star - g(\alpha_k^\star)$$
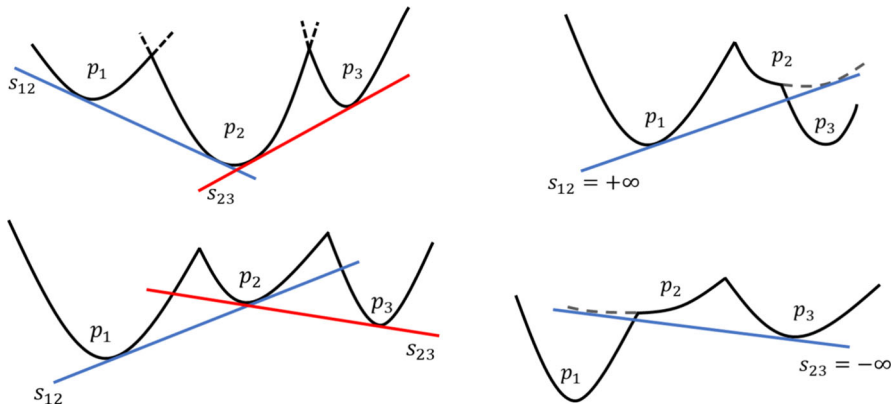
Fig. 8 The first row corresponds to the **ADD** step of Algorithm 4. The second row corresponds to the **DELETE** step, wherein piece p_2 is discarded by the algorithm

$$= (s_{kl} - \epsilon)\alpha_k^\star - p_k(\alpha_k^\star)$$
$$= \max_\alpha \{(s_{kl} - \epsilon)\alpha - p_k(\alpha)\}$$
$$= p_k^*(s_{kl} - \epsilon).$$

Similarly, since $I_g^+(s_{kl}) = l$, there exists some $\bar{\epsilon} > 0$ such that for all $\epsilon \in (0, \bar{\epsilon})$ we have $g^*(s_{kl} + \epsilon) = p_l^*(s_{kl} + \epsilon)$. The above two equations imply that s_{kl} is indeed a breakpoint of g^*.

Conversely, suppose that a point τ is a breakpoint for g^*. Since τ is a breakpoint, we must have $I_g^-(\tau) \neq I_g^+(\tau)$. This together with the non-decreasing property of I_g implies that $I_g^-(\tau) < I_g^+(\tau)$. Let $k = I_g^-(\tau)$ and $l = I_g^+(\tau)$ for some $k < l$. We proceed to prove that τ is indeed the feasible common tangent to the pieces k and l. First, it is easy to verify that p_k and p_l cannot be identical. Define $\alpha_k^\star = \mathrm{argmax}_\alpha\{\tau\alpha - p_k(\alpha)\}$ and $\alpha_l^\star = \mathrm{argmax}_\alpha\{\tau\alpha - p_l(\alpha)\}$. Due to the definition of the indexing function, we have $\tau_{k-1} \leq \alpha_k^\star \leq \tau_k$ and $\tau_{l-1} \leq \alpha_l^\star \leq \tau_l$. Consider the lines $\ell_{\tau,k}(\alpha) = \tau\alpha - p_k^*(\tau)$ and $\ell_{\tau,l}(\alpha) = \tau\alpha - p_l^*(\tau)$. Indeed, these two lines are tangent to pieces k and l at points $\alpha_k^\star$ and $\alpha_l^\star$, respectively. Moreover, they coincide since $p_k^*(\tau) = p_l^*(\tau)$ due to the continuity of g^*. Therefore, τ is the feasible common tangent to the pieces k and l. $\qquad\square$

According to Lemma 10, it suffices to identify every pair of pieces $k < l$ that satisfy the breakpoint condition. This can be naturally achieved by verifying the condition for all $\binom{n}{2}$ pairs of pieces. Our proposed Algorithm 4, which we call the *breakpoint algorithm*, achieves this goal in linear time. It keeps track of two ordered lists Γ and Π. The list Γ collects the set of candidate breakpoints, whereas the list Π records the pieces that satisfy the breakpoint condition. In other words, upon termination, the pieces $\Pi(j)$ and $\Pi(j+1)$ satisfy the breakpoint condition for any $j = 1, \ldots, |\Pi|-1$. The initial values of these lists are set as $\Gamma = [-\infty]$ and $\Pi = [1]$.

At every iteration, the algorithm takes one of the following steps:

Algorithm 4 Breakpoint algorithm

**Input:** $g + \lambda \mathbb{1}_\alpha$, where g is semi-consistent
Output: The conjugate of the input function
1: $\Gamma \leftarrow [-\infty]$ ▷ Ordered list of candidate breakpoints of g^*
2: $\Pi \leftarrow [1]$ ▷ Ordered indices satisfying the breakpoint condition
3: $j \leftarrow 2$
4: **while** $j \le N$ **do**
5:    $i \leftarrow \text{end}(\Pi)$ ▷ Return the last (maximum) element of Π
6:    $s_{ij} \leftarrow \text{SLOPE}(p_i, \tau_{i-1}, \tau_i, p_j, \tau_{j-1}, \tau_j)$ ▷ Obtain the feasible common tangent
7:    **if** $s_{ij} > \text{end}(\Gamma)$ **then**   ▷ **ADD**
8:       $\Gamma \leftarrow \text{append}(\Gamma, s_{ij})$ ▷ Append s_{ij} to Γ as a new breakpoint
9:       $\Pi \leftarrow \text{append}(\Pi, j)$ ▷ Append j to Π
10:       $j \leftarrow j + 1$
11:    **else if** $s_{ij} \le \text{end}(\Gamma)$ **then**   ▷ **DELETE**
12:       $\Gamma \leftarrow \text{delete}(\Gamma, \text{end}(\Gamma))$ ▷ Delete the last breakpoint from Γ
13:       $\Pi \leftarrow \text{delete}(\Pi, \text{end}(\Pi))$ ▷ Delete the last index from Π
14:    **end if**
15: **end while**
16: $\Gamma \leftarrow \text{append}(\Gamma, +\infty)$
17: Define $g^*(\beta) = p^*_{\Pi(k)}(\beta)$, for $\Gamma(k) \le \beta \le \Gamma(k+1)$; $k = 1, \ldots, M$.
18: Find the roots $\beta_1 < \beta_2$ of $-g(0) = g^*(\beta) - \lambda$
19: **return** the conjugate of $g(\alpha) + \lambda \mathbb{1}_\alpha$ as

$$\begin{cases} -g(0) & \beta_1 \le \beta \le \beta_2 \\ g^*(\beta) - \lambda & \text{otherwise} \end{cases}$$

Fig. 9 The auxiliary function $\tilde{g}$ defined by removing piece $N - 1$ from g, and extending the pieces $N - 2$ and N to substitute piece $N - 1$
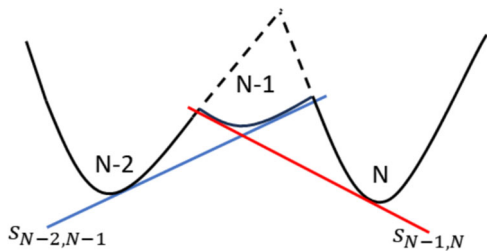
– *ADD (Line 7 of Algorithm 4):* When a common tangent between the piece j and the highest index i in Π is greater than the largest discovered breakpoint in Γ, the algorithm adds the index j and the common tangent s_{ij} to the lists Π and Γ, respectively. This scenario is depicted in the first row of Fig. 8. Note that, at this step, it is possible for the algorithm to add an infeasible common tangent with $s_{ij} = +\infty$ to Γ (see Fig. 8, top right figure). However, both s_{ij} and j will be discarded in the DELETE step, as we explain next.
– *DELETE (Line 11 of Algorithm 4):* When the slope of a common tangent between the piece j and the piece with the highest index i in Π is less than or equal to the largest discovered breakpoint τ in Γ, the algorithm deletes the last elements of the lists Γ and Π. Intuitively, this condition implies that the last piece of Π cannot satisfy the breakpoint condition when paired with any other piece. As another interpretation, this piece does not play a role in characterizing the convex

envelope of g since it lies in the interior of its epigraph. This scenario is also depicted in the second row of Fig. 8.

Our next theorem shows that the breakpoint algorithm returns the conjugate of any function $g(\alpha) + \lambda \mathbb{1}_\alpha$, provided that g is semi-consistent.

Theorem 3 *Let g be semi-consistent with N pieces. The breakpoint algorithm (Algorithm 4) correctly computes the conjugate of $g + \lambda \mathbb{1}_\alpha$ for any $\lambda > 0$ in $\mathcal{O}(N)$ time and memory.*

Before presenting the proof of the above theorem, we show how it can be used to complete the proof of Proposition 2.

Proof of Proposition 2 According to Lemma 8, $\tilde{f}$ is semi-consistent. Therefore, the proof readily follows upon choosing $g = \tilde{f}$ in Theorem 3. □

Next, we present the main idea behind the correctness proof of the breakpoint algorithm. Our proof is based on induction on the number of pieces in g. Suppose the breakpoint algorithm returns the conjugate of any semi-consistent function with at most $N - 1$ pieces. Our goal is to use this assumption to prove that the algorithm returns the conjugate of g_N with N pieces. Note that, when running the breakpoint algorithm on g_N, the algorithm first processes the first $N - 1$ pieces of g_N, which are identical to g_{N-1}. Due to Definition 4, g_{N-1} is semi-consistent with $N - 1$. Therefore, relying on our induction hypothesis, the breakpoint algorithm correctly identifies the breakpoints and pieces of g_{N-1}^*. Let $s_{i,N-1}$ and $N - 1$ denote the last breakpoint and piece added to Γ and Π respectively until the algorithm reaches piece N. Upon processing piece N, two potential scenarios emerge:

- Case 1: $s_{i,N-1} < s_{N-1,N}$. In this case, the algorithm "adds" the breakpoint $s_{N-1,N}$ and the piece N to Γ and Π, then returns these sets as the set of breakpoints and pieces of g_N^*. We prove that these sets coincide with the true sets of breakpoints and pieces of g_N^*.
- Case 2: $s_{i,N-1} \geq s_{N-1,N}$. In this scenario, the algorithm "deletes" the breakpoint $s_{i,N-1}$ and the piece $N - 1$ from Γ and Π respectively. Here, we establish that the piece $N - 1$ does not contribute to the characterization of g_N^*. In this scenario, g_N^* is the same as the conjugate of an auxiliary function $\tilde{g}_{N-1}$, obtained by removing piece $N - 1$ from g_N, and subsequently, extending pieces $N - 2$ and N to substitute piece $N - 1$. Figure 9 illustrates this function. We show that the constructed $\tilde{g}_{N-1}$ is semi-consistent and has $N - 1$ pieces. Therefore, by induction hypothesis, the algorithm correctly recovers its conjugate.

The rest of this section is devoted to formalizing the above intuition.

Proof of Theorem 3 We begin by presenting the proof of correctness, followed by the proof of its runtime. Suppose that Line 17 correctly recovers g^*. Upon finding the roots $\beta_1 < \beta_2$ of $-g(0) = g^*(\beta) - \lambda$, Equation (21) can be invoked to show that Line 19 returns the conjugate of $g + \lambda \mathbb{1}_\alpha$. Therefore, to prove the correctness of the algorithm, it suffices to show that Line 17 correctly recovers g^*. To this goal, we prove that the ordered lists Γ and Π coincide with the correct breakpoints and pieces

of g^*, respectively. Our proof is by induction on the number of pieces N of g. Recall that $g = g_N$ as defined Definition 4. To streamline the presentation, we keep the dependency of g on N explicit throughout the proof.

Base case Suppose $N = 1$. Indeed, both g_N and g_N^* have one piece with no breakpoints. Since the While loop in Line 4 starts only when g has more than one piece, the algorithm correctly returns the initial values of $\Pi = [1]$ and $\Gamma = [-\infty, +\infty]$. Thus, the base case of the induction hypothesis is true.

Induction step Suppose that the breakpoint algorithm correctly recovers Γ and Π for any semi-consistent function g_{N-1} with at most $N - 1$ pieces. Our goal is to prove that the algorithm correctly recovers the correct breakpoints and pieces for any semi-consistent function g_N with N pieces.

We use $(\Gamma_N^\star, \Pi_N^\star)$ and (Γ_N, Π_N) to denote the true set of breakpoints and pieces of g_N^*, and those returned by the algorithm, respectively. Similarly, $(\Gamma_{N-1}^\star, \Pi_{N-1}^\star)$ and $(\Gamma_{N-1}, \Pi_{N-1})$ are the true breakpoints and pieces, and those returned by the algorithm for g_{N-1}^*, respectively. From our induction hypothesis, we have $\Gamma_{N-1}^\star = \Gamma_{N-1}$ and $\Pi_{N-1}^\star = \Pi_{N-1}$. When we apply the algorithm to g_N, the algorithm first processes the first $N - 1$ pieces of g_N. Let $(\tilde{\Gamma}_{N-1}, \tilde{\Pi}_{N-1})$ denote the set of breakpoints and pieces returned by the algorithm at this point. For g_{N-1}, the piece $N - 1$ is defined over the domain $[\tau_{N-2}, \infty)$. Therefore, we have $\lim_{\beta \to \infty} I_{g_{N-1}}(\beta) = N - 1$. Suppose i is the piece for which the pair i and $N - 1$ satisfies the breakpoint condition for g_{N-1}. This implies that

1. $I_{g_{N-1}}^-(s_{i,N-1}) = I_{g_{N-1}}(s_{i,N-1}) = i$,
2. $I_{g_{N-1}}^+(s_{i,N-1}) = N - 1$.

Due to the non-decreasing property of $I_{g_{N-1}}$, we have $s_{i,N-1} = \max\{\Gamma_{N-1}^\star\}$. We consider two cases:

Case 1: $s_{i,N-1} < s_{N-1,N}$ In this case, the algorithm proceeds with the ADD step and returns $\Gamma_N = \tilde{\Gamma}_{N-1} \cup \{s_{N-1,N}\}$ and $\Pi_N = \tilde{\Pi}_{N-1} \cup \{N\}$. We show that these sets coincide with $(\Gamma_N^\star, \Pi_N^\star)$.

Claim 1 $\tilde{\Gamma}_{N-1} = \Gamma_{N-1}^\star$ *and* $\tilde{\Pi}_{N-1} = \Pi_{N-1}^\star$.

To prove this claim, we first observe that the algorithm runs identically over the first $N - 2$ pieces of g_{N-1} and g, since these functions are identical over $(-\infty, \tau_{N-1}]$. Therefore, it follows that $\tilde{\Gamma}_{N-1}$ matches $\Gamma_{N-1}^\star$ entirely, except for a potential distinction in their final elements. This distinction occurs only if $s_{i,N-1} = -\infty$ or $s_{i,N-1} = +\infty$. Since $s_{i,N-1} \in \Gamma_{N-1}^\star$, we have $-\infty < s_{i,N-1}$. Moreover, since $\tau_N = +\infty$, we have $s_{N-1,N} < +\infty$ according to Algorithm 3. This implies that $-\infty < s_{i,N-1} < s_{N-1,N} < +\infty$. Therefore, both $s_{i,N-1}$ and $s_{N-1,N}$ are finite and $\tilde{\Gamma}_{N-1} = \Gamma_{N-1}^\star$. The proof of $\tilde{\Pi}_{N-1} = \Pi_{N-1}^\star$ follows similarly.

Based on the above claim, it suffices to show that $\Gamma_N^\star = \Gamma_{N-1}^\star \cup \{s_{N-1,N}\}$ and $\Pi_N^\star = \Pi_{N-1}^\star \cup \{N\}$. To this goal, we rely on two crucial claims.

Claim 2 $g_{N-1}^\star(\beta) = \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_{N-1}(\alpha)\}$ *for every* $\beta < s_{N-1,N}$.

To prove the above claim, it suffices to show that, for every $\beta < s_{N-1,N}$, there exists some $\alpha^\star(\beta) \in \{\text{argmax}_\alpha\{\alpha\beta - g_{N-1}(\alpha)\}\}$ such that $\alpha^\star(\beta) < \tau_{N-1}$. First consider the case $\beta \leq s_{i,N-1}$. In this case, $I_{g_{N-1}}(\beta) \leq i$, which in turn implies $\alpha^\star(\beta) < \tau_i < \tau_{N-1}$. When $s_{i,N-1} < \beta \leq s_{N-1,N}$, from the non-decreasing property of the indexing function, we have $I_{g_{N-1}}(\beta) = N-1$. Thus, $\max_\alpha\{\alpha\beta - g_{N-1}(\alpha)\} = \max_\alpha\{\alpha\beta - p_{N-1}(\alpha)\}$ for every $s_{i,N-1} < \beta \leq s_{N-1,N}$. Since p_{N-1} is strongly convex, $\alpha^\star(\beta)$ is an increasing function of β for every $s_{i,N-1} < \beta \leq s_{N-1,N}$. On the other hand, $\alpha^\star(s_{N-1,N}) = \text{argmax}_\alpha\{\alpha s_{N-1,N} - p_{N-1}(\alpha)\} < \tau_{N-1}$, where the last inequality follows from the fact that $s_{N-1,N}$ is finite and is the feasible common tangent to pieces $N-1$ and N. Therefore, we have $\alpha^\star(\beta) < \tau_{N-1}$ for every $\beta < s_{N-1,N}$.

Claim 3 $g^*_{N-1}(\beta) > \alpha\beta - p_N(\alpha)$ *for every* $\beta < s_{N-1,N}$ *and* $\alpha > \tau_{N-1}$.

To prove this claim, define the line $\ell_\beta(\alpha) = \alpha\beta - g^*_{N-1}(\beta)$. It is easy to see that

$$\ell_{s_{N-1,N}}(\alpha) > \ell_\beta(\alpha), \qquad \text{for every } \beta < s_{N-1,N} \text{ and } \alpha > \tau_{N-1}. \qquad (24)$$

Since $I_{g_{N-1}}(s_{N-1,N}) = N-1$, it follows that $g^*_{N-1}(s_{N-1,N}) = p^*_{N-1}(s_{N-1,N}) = p^*_N(s_{N-1,N})$. Thus,

$$\ell_{s_{N-1,N}}(\alpha) = s_{N-1,N}\alpha - g^*_{N-1}(s_{N-1,N}) = s_{N-1,N}\alpha - p^*_N(s_{N-1,N}).$$

On the other hand, due to the property of conjugate functions, for all $\alpha \in \mathbb{R}$,

$$p_N(\alpha) \geq s_{N-1,N}\alpha - p^*_N(s_{N-1,N}) = \ell_{s_{N-1,N}}(\alpha).$$

The above inequality together with (24) implies that

$$
\begin{aligned}
&p_N(\alpha) > \ell_\beta(\alpha), && \text{for every } \beta < s_{N-1,N} \text{ and } \alpha > \tau_{N-1} \\
\Longleftrightarrow\ &p_N(\alpha) > \beta\alpha - g^*_{N-1}(\beta), && \text{for every } \beta < s_{N-1,N} \text{ and } \alpha > \tau_{N-1} \\
\Longleftrightarrow\ &g^*_{N-1}(\beta) > \beta\alpha - p_N(\alpha), && \text{for every } \beta < s_{N-1,N} \text{ and } \alpha > \tau_{N-1},
\end{aligned}
$$

which completes the proof of this claim.

Equipped with Claims 2 and 3, we are ready to prove $\Gamma^\star_N = \Gamma^\star_{N-1} \cup \{s_{N-1,N}\}$ and $\Pi^\star_N = \Pi^\star_{N-1} \cup \{N\}$. To this goal, it suffices to show that

$$g^*_N(\beta) = \begin{cases} g^*_{N-1}(\beta) & \beta < s_{N-1,N} \\ p^*_N(\beta) & \beta \geq s_{N-1,N}. \end{cases}$$

Consider the case $\beta < s_{N-1,N}$. From the definition of the conjugate function, we have

$$
\begin{aligned}
g^\star_N(\beta) &= \max_{\alpha \in \mathbb{R}} \{\alpha\beta - g_N(\alpha)\} \\
&= \max\left\{\max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_N(\alpha)\},\ \max_{\alpha > \tau_{N-1}} \{\alpha\beta - g_N(\alpha)\}\right\}
\end{aligned}
$$

$$= \max \left\{ \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_{N-1}(\alpha)\}, \max_{\alpha > \tau_{N-1}} \{\alpha\beta - p_N(\alpha)\} \right\}$$

$$= \max \left\{ g^*_{N-1}(\beta), \max_{\alpha > \tau_{N-1}} \{\alpha\beta - p_N(\alpha)\} \right\}$$

$$= g^*_{N-1}(\beta),$$

where the second to last equality follows from Claim 2 and the last equality is due to Claim 3. Using the fact that $g^*_N(\beta) = g^*_{N-1}(\beta)$ for $\beta < s_{N-1,N}$, we obtain $I_{g_N}(\beta) = I_{g_{N-1}}(\beta) = N - 1$ for $s_{i,N-1} < \beta < s_{N-1,N}$. On the other hand, $\lim_{\beta \to +\infty} I_g(\beta) = N$. Therefore, $s_{N-1,N} \in \Gamma^\star_N$, which implies $g^\star_N(\beta) = p^*_N(\beta)$ for $\beta \geq s_{N-1,N}$. This completes the proof of the first case.

Case 2: $s_{i,N-1} \geq s_{N-1,N}$ In this case, the algorithm proceeds with the DELETE step and discards $s_{i,N-1}$ and $N - 1$ from $\tilde{\Gamma}_{N-1}$ and $\tilde{\Pi}_{N-1}$, respectively. Our next claim shows that both $s_{i,N-1}$ and $N - 1$ are correctly deleted, as piece $N - 1$ does not belong to Image(I_{g_N}).

Claim 4 $N - 1 \notin$ Image(I_{g_N}).

To prove this claim, suppose, by contradiction, that $N - 1 \in$ Image(I_{g_N}). This implies that there exists a piece k such that both pairs $k, N - 1$ and $N - 1, N$ satisfy the breakpoint condition for g_N. Therefore, $s_{k,N-1} < s_{N-1,N} \leq s_{i,N-1}$. Due to the non-decreasing property of I_{g_N}, we have $I_{g_N}(\beta) \leq N - 1$ for every $\beta \leq s_{N-1,N}$. This implies the existence of $\alpha^\star(\beta) \leq \tau_{N-1}$ such that $\alpha^\star(\beta) \in \text{argmax}_\alpha \{\alpha\beta - g_{N-1}(\alpha)\}$ for every $\beta \leq s_{N-1,N}$. Therefore, we have $g^*_{N-1}(\beta) = \max_\alpha \{\alpha\beta - g_{N-1}(\alpha)\} = \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_{N-1}(\alpha)\}$ for every $\beta \leq s_{N-1,N}$. Similarly, we have $I_{g_{N-1}}(\beta) \leq i$ for every $\beta \leq s_{i,N-1}$, which leads to $g^*_N(\beta) = \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_N(\alpha)\}$ for every $\beta \leq s_{i,N-1}$. Combining these two equalities, for every $\beta \leq s_{N-1,N} \leq s_{i,N-1}$, we have

$$g^*_{N-1}(\beta) = \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_{N-1}(\alpha)\} = \max_{\alpha \leq \tau_{N-1}} \{\alpha\beta - g_N(\alpha)\} = g^*_N(\beta).$$

The above equality implies that $I_{g_{N-1}}(\beta) = I_{g_N}(\beta) = N - 1$ for every $s_{k,N-1} < \beta \leq s_{N-1,N}$. On the other hand, recall that $I_{g_{N-1}}(\beta) \leq i$ for every $\beta \leq s_{i,N-1}$, which leads to $I_{g_{N-1}}(\beta) < N - 1$ for every $s_{k,N-1} < \beta \leq s_{N-1,N}$. This leads to a contradiction, thereby proving the claim.

As the last step of the proof, we consider the following function:

$$\tilde{g}_{N-1}(\alpha) = \begin{cases} \min\{p_{N-2}(\alpha), p_N(\alpha)\} & \tau_{N-2} < \alpha < \tau_{N-1}, \\ g(\alpha) & \text{otherwise.} \end{cases} \tag{25}$$

The function $\tilde{g}_{N-1}$ is obtained by removing piece $N - 1$ from g, and subsequently, extending p_{N-2} and p_N to substitute piece $N - 1$. Our final claim shows that $\tilde{g}_{N-1}$ and g_N have the same conjugates.

Claim 5 $\tilde{g}^*_{N-1} = g^*_N$.

To prove this claim, note that $\tilde{g}_{N-1}(\alpha)$ and $g(\alpha)$ are identical except within the interval $[\tau_{N-2}, \tau_{N-1}]$. Due to the second property of semi-consistent functions (Definition 4), we have $\tilde{g}_{N-1}(\alpha) \geq g(\alpha)$ within the interval $[\tau_{N-2}, \tau_{N-1}]$. This implies that

$$\beta\alpha - g(\alpha) \geq \beta\alpha - \tilde{g}_{N-1}(\alpha); \qquad \forall \alpha, \beta \in \mathbb{R}$$
$$\implies \max_{\alpha}\{\beta\alpha - g(\alpha)\} \geq \max_{\alpha}\{\beta\alpha - \tilde{g}_{N-1}(\alpha)\}; \qquad \forall \beta \in \mathbb{R}$$
$$\implies g^*(\beta) \geq \tilde{g}^*_{N-1}(\beta); \qquad \forall \beta \in \mathbb{R}.$$

On the other hand, due to Claim 4, we have $N-1 \notin \text{Image}(I_{g_N})$. Therefore, for every $\beta \in \mathbb{R}$, there exists $\alpha^\star(\beta) \notin [\tau_{k-1}, \tau_k]$ such that $\alpha^\star(\beta) \in \text{argmax}_\alpha\{\beta\alpha - g(\alpha)\}$. This implies that, for every $\beta \in \mathbb{R}$:

$$\begin{aligned}
g^*(\beta) &= \max_\alpha\{\beta\alpha - g(\alpha)\} \\
&= \beta\alpha^\star(\beta) - g(\alpha^\star(\beta)) \\
&= \beta\alpha^\star(\beta) - \tilde{g}_{N-1}(\alpha^\star(\beta)) \\
&\leq \max_\alpha\{\beta\alpha - \tilde{g}_{N-1}(\alpha)\} = \tilde{g}^*_{N-1}(\beta).
\end{aligned}$$

Combining the above two inequalities implies that $\tilde{g}^*_{N-1}(\beta) = g^*_N(\beta)$, thereby proving the claim.

After discarding piece $N-1$, the algorithm operates identically on $\tilde{g}_{N-1}$ and g_N. Indeed, $\tilde{g}_{N-1}$ is semi-consistent since it satisfies the properties outlined in Definition 4. Given that $\tilde{g}_{N-1}$ contains $N-1$ pieces, by our induction hypothesis, the breakpoint algorithm correctly identifies the breakpoints and pieces of $\tilde{g}^*_{N-1}$, which coincide with those of g^*_N as asserted in the above claim. This completes the correctness proof of the algorithm.

Finally, we analyze the runtime of the algorithm. We consider the operations within the While loop of Algorithm 4. Every execution of Algorithm 3 can be completed in $\mathcal{O}(1)$. To see this, note that the If conditions in Lines 5, 7, 9, and 11 of Algorithm 3 can be checked in $\mathcal{O}(1)$ time. The remaining operations of the While loop either add or delete an element to a list, each taking $\mathcal{O}(1)$ time and memory. Thus a single round of the While loop can be executed in $\mathcal{O}(1)$ time and memory. Next, we show that the While loop executes at most $\mathcal{O}(N)$ rounds. Once a piece is deleted, it will never be revisited. Since at most N pieces can be added and at most N pieces can be deleted, the While loop can execute at most $\mathcal{O}(N)$ rounds. Finally, note that, since Π and Γ have $\mathcal{O}(N)$ elements, computing g^*_N in Line 17 requires $\mathcal{O}(N)$ time and memory. Similarly, it follows that Line 18 can be computed in $\mathcal{O}(N)$. Consequently, we conclude that Algorithm 4 operates in $\mathcal{O}(N)$ time and memory. $\qquad \square$

4 Numerical stability of the parametric algorithm

The breakpoint algorithm (Algorithm 4) is prone to numerical instabilities for trees with a large number of nodes. In this section, we explain the root cause of this numerical

issue and describe a correction step that averts this without any compromises to the performance and accuracy of the algorithm.

Consider an arbitrary pair of nodes u, v where $v = \text{child}(u)$ and v is not a branch. Since $f_u(\alpha)$ is consistent, it can be written as

$$f_u(\alpha) = \min_{1 \leq k \leq N_u} \{p_{u,k}(\alpha)\} + \lambda_u \, \mathbb{1}_\alpha,$$

where $\{p_{u,k}(\alpha)\}_{k=1}^{N_u}$ are strongly convex and quadratic. For every $k = 1, \ldots, N_u$, let $p_{u,k}(\alpha) = \gamma_{u,k,1}\alpha^2 + \gamma_{u,k,2}\alpha + \gamma_{u,k,3}$. Lemma 3 and Equation (9) imply that

$$f_v(\alpha) = \min_{1 \leq k \leq N_v} \{p_{v,k}(\alpha)\} + \lambda_v \, \mathbb{1}_\alpha,$$

where

$$p_{v,k}(\alpha) = \frac{1}{2}\alpha^2 + c_v\alpha - p_{u,k}^*(-Q_{u,v}\alpha) \tag{26}$$

$$= \underbrace{\left(\frac{1}{2} - \frac{Q_{u,v}^2}{4\gamma_{u,k,1}}\right)}_{:=\gamma_{v,k,1}}\alpha^2 + \underbrace{\left(c_v - \frac{\gamma_{u,k,2}Q_{u,v}}{2\gamma_{u,k,1}}\right)}_{:=\gamma_{v,k,2}}\alpha + \underbrace{\left(\gamma_{u,k,3} - \frac{\gamma_{u,k,2}^2}{4\gamma_{u,k,1}}\right)}_{:=\gamma_{v,k,3}}. \tag{27}$$

Suppose, for some arbitrary indices k, l, we obtain $\gamma_{v,k,1}, \gamma_{v,l,1}$ from $\gamma_{u,k,1}, \gamma_{u,l,1}$ based on the equation above. Taking $\epsilon = |\gamma_{u,k,1} - \gamma_{u,l,1}|$, we obtain

$$|\gamma_{v,k,1} - \gamma_{v,l,1}| = \underbrace{\left|\frac{Q_{u,v}^2}{4\gamma_{u,k,1}\gamma_{u,l,1}}\right|}_{\rho} \cdot \epsilon.$$

When $\gamma_{u,k,1}, \gamma_{u,l,1} > Q_{u,v}/2$, we observe that $\rho < 1$, resulting in a decrease in the discrepancy of the quadratic terms. This scenario is likely to occur in practice, as $|Q_{u,v}| < 1$ due to the positive definiteness of Q, and the quadratic coefficients remain close to $1/2$ due to (26). The shrinking effect of the update rule is exacerbated in situations where multiple neighboring nodes satisfy $\rho < 1$, thereby leading to fast decay in ϵ. As ϵ approaches machine precision, the breakpoint algorithm would suffer from numerical instabilities.

To address this challenge, we note that, since the slope of the common tangent s_{kl} is proportional to ϵ^{-1}, such errors arise only at breakpoints with significantly large absolute values. Our subsequent lemma demonstrates that these breakpoints correspond to suboptimal pieces, and thus can be easily excluded from consideration.

Lemma 11 *Let $x^\star$ be the optimal solution of Problem (1). We have $\|x^\star\|_\infty \leq \frac{\|c\|_2}{\lambda_{\min}(Q)}$, where $\lambda_{\min}(Q)$ denotes the smallest eigenvalue of Q.*

Proof Suppose J corresponds to the set of row indices over which $x^\star$ is non-zero. We have $x^\star = -Q_{J,J}^{-1} c_J$, which implies

$$\|x^\star\|_\infty \leq \|x^\star\|_2 = \left\| (Q_{J,J})^{-1} c_J \right\|_2 \leq \|(Q_{J,J})^{-1}\|_2 \|c\|_2 \leq \frac{\|c\|_2}{\lambda_{\min}(Q_{J,J})}.$$

Since $J \subset N$, we have

$$\lambda_{\min}(Q_{J,J}) = \min_{\|x\|_2=1} x^\top Q_{J,J} x = \min_{\substack{\|x\|_2=1, \\ x_J=0}} x^\top Q x \geq \min_{\|x\|_2=1} x^\top Q x \geq \lambda_{\min}(Q).$$

This completes the proof. □

According to the above lemma, it suffices to characterize the parametric cost at any node u within the range $\left[-\frac{\|c\|_2}{\lambda_{\min}(Q)}, \frac{\|c\|_2}{\lambda_{\min}(Q)} \right]$. Therefore, the aforementioned numerical issue can be mitigated by first obtaining $\frac{\|c\|_2}{\lambda_{\min}(Q)}$ and then discarding the breakpoints falling outside the range $\left[-\frac{\|c\|_2}{\lambda_{\min}(Q)}, \frac{\|c\|_2}{\lambda_{\min}(Q)} \right]$.

5 Experiments

In this section, we assess the performance of our algorithm across various synthetic and real-world case studies. All experiments were run on a computer with 16 cores of 3.0 GHz Xeon Gold 6154 processors and 8 GB memory per core. Specifically, we compare the proposed parametric algorithm with Gurobi v10.0.2. For Gurobi, a time limit of 1 h was set, and the algorithm was terminated whenever the optimality gap fell below 0.01%. If Gurobi failed to achieve an optimality gap of 0.01% or less within this time limit, we reported the best optimality gap attained. We also note that Gurobi, from version 10 onwards, uses a branch-and-bound method based on a perspective reformulation to solve Problem (1); these reformulations are known to outperform the classical *big-M* reformulations (see, e.g., [52]) and are considered state-of-the-art.[3] The Python implementation of our algorithm as well as the presented case studies are available at https://github.com/aareshfb/Tree-Parametric-Algorithm.git.

5.1 Case study on synthetic dataset

For our first set of experiments, we construct supp(Q) as a randomly generated connected tree. The nonzero off-diagonal elements are selected from a uniform distribution within the range $[-1, 0]$. Each diagonal element $Q_{i,i}$ is set to $1 + \sum_{j \neq i} |Q_{i,j}|$. This ensures that Q is positive definite. Similarly, elements of vector c were generated

---

[3] We also attempted to manually hard-code the perspective reformulation in Gurobi. However, our observations indicate that Gurobi's default setting implements the perspective reformulation more effectively than our manual approach. While we did not include this comparison in the paper, the results are available in our GitHub repository.
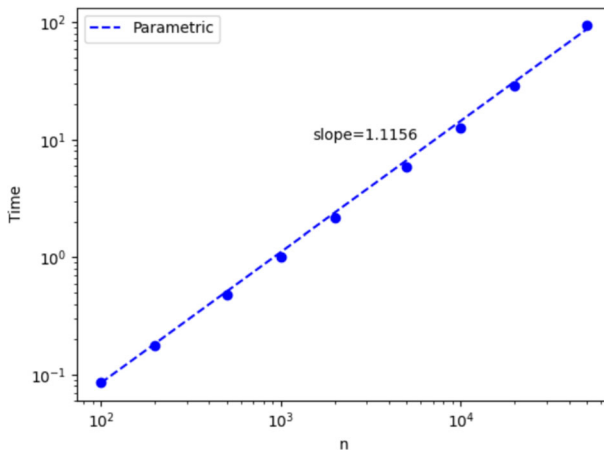
Fig. 10 The runtime of the parametric algorithm (Algorithm 2) for different values of n. The reported results are averaged over 5 trials

from a uniform distribution within the interval $(-10, 10)$. Unless explicitly stated otherwise, the default regularizing parameter was set to $\lambda_i = 7.5$ for all i. This value approximately corresponds to 50% non-zero elements in the optimal solution for the selected Q and c.

First, we examine the performance of the parametric algorithm for problems with varying size n. The results are presented in Table 1.

It can be seen that Gurobi is unable to solve instances with sizes exceeding $n = 200$ within 1 h. In contrast, our proposed parametric algorithm can solve instances with $n = 5000$ in less than 6 s, significantly outperforming Gurobi. As a sanity check, we also report the optimal objective values of both methods. To provide further insight into the efficiency of the parametric algorithm, we plot its runtime across a broader range of n in Fig. 10. Notably, the parametric algorithm can solve instances of size $n = 50,000$ within 2 min.

Moreover, while the theoretical complexity of the parametric algorithm can be as high as $\mathcal{O}(n^2)$, in practice, we observe a complexity that is closer to linear $\mathcal{O}(n^{1.1156})$. This improved complexity can be attributed to the fact that, while the parametric cost at the root node $f_n(\alpha)$ may have up to $2n$ pieces, in practice, the number of pieces is expected to be significantly smaller. More specifically, recall that N_u denotes the number of pieces in the parametric cost $f_u(x)$. We have shown that the runtime of Algorithm 2 is $\mathcal{O}\left(\sum_{u=1}^{n} N_u\right) = \mathcal{O}\left(n\bar{N}\right)$, where $\bar{N}$ denotes the average number of pieces. While this leads to a quadratic runtime when $\bar{N} = \mathcal{O}(n)$, it becomes linear if $\bar{N} = \mathcal{O}(1)$.

Figure 11 illustrates the average number of pieces generated by the parametric algorithm for different values of n. It is evident that as n increases from 1000 to 20,000, the average number of pieces ranges from 20 to 35. This observation supports our hypothesis that, in practice, the average number of pieces grows only sublinearly with respect to n.

Table 1 Performance for varying sizes

Metric	Method	$n = 200$	$n = 500$	$n = 1000$	$n = 2000$	$n = 5000$
Time (s)	Parametric	0.17	0.47	0.99	2.14	5.99
	Gurobi	68.50	TL	TL	TL	TL
Objective value	Parametric	−970.75	−2568.81	−5162.32	−10,437.16	−25614.37
	Gurobi	−970.75	−2568.77	−5162.20	−10,435.72	−25,606.82
B&B nodes	Gurobi	643,413	12,883,889	6,329,913	2,865,993	697,020
Opt. gap (%)	Gurobi	0.00	1.02	1.55	1.91	2.19
Bound	–	79.88	128.44	182.77	259.63	408.44

TL: Time Limit (1 h). The reported results are averaged over 5 trials. "Parametric" refers to the parametric algorithm (Algorithm 2). "Bound" refers to the value of the term $\|c\|_2/\lambda_{\min}(Q)$ derived for the numerical stability of Algorithm 4 in Lemma 11
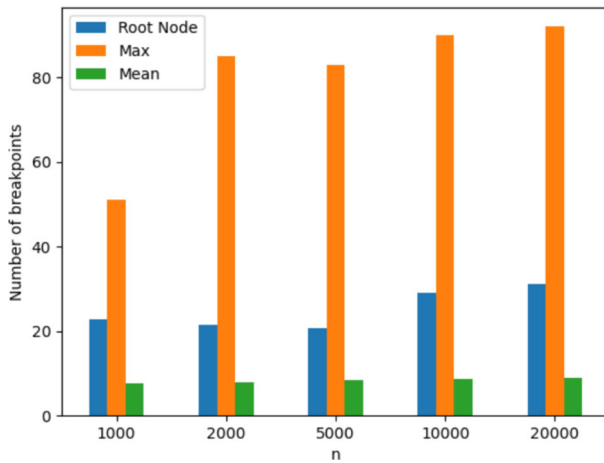
Fig. 11 The values of $\bar{N}$ (denoted as "Mean"), $\max_u\{N_u\}$ (denoted as "Max"), and N_1 (denoted as "Root Node") for different values of n. Note that the $\max_u\{N_u\}$ does not necessarily coincide with N_1. The reported results are averaged over 10 trials

Next, we let $n = 1000$ and compare the performance of the parametric algorithm and Gurobi for different regularization parameters λ. Given any index i, the parameter λ_i is sampled randomly from a uniform distribution within the range $[0.75\bar{\lambda}, 1.25\bar{\lambda}]$. The results are summarized in Table 2. It is observed that while the performance of the parametric algorithm remains almost independent of $\bar{\lambda}$, the optimality gap obtained by Gurobi remains large, except for the extreme values of $\bar{\lambda}$ that correspond to nearly fully dense or fully sparse solutions.

Finally, we focus on the special case of path graphs. Specifically, we compare our parametric algorithm (Algorithm 1) to the direct DP approach proposed in [40]. As discussed in Sect. 2, the direct DP approach solves instances with path structure in $\mathcal{O}(n^2)$ time complexity. While this runtime matches the theoretical complexity of our parametric algorithm, Fig. 12 illustrates that their practical performance differs. In particular, while the direct DP approach outperforms the parametric algorithm for $n \leq 2000$, its runtime scales almost quadratically with n. On the other hand, the practical performance of the parametric algorithm scales almost linearly with n, enabling it to outperform the direct DP approach for larger instances $n > 2000$.

5.2 Case study on accelerometer dataset

In this case study, we highlight the performance of the parametric algorithm for solving the robust inference of GHMM, as detailed in Sect. 1.1. Specifically, we focus on the task of recognizing physical activities for a participant using data collected from a single chest-mounted accelerometer. We consider the dataset from [11, 12]. To enhance the representation of these activities, [3] proposed using the mean absolute value of 10 successive signal differences from this dataset. The pre-processed data can be accessed online at https://sites.google.com/usc.edu/gomez/data.

Table 2 Performance comparison for varying regularization

Metric	Method	$\bar{\lambda} = 0.2$ NZ $\approx 91\%$	$\bar{\lambda} = 2$ NZ $\approx 73\%$	$\bar{\lambda} = 5$ NZ $\approx 59\%$	$\bar{\lambda} = 13$ NZ $\approx 34\%$	$\bar{\lambda} = 25$ NZ $\approx 10\%$	$\bar{\lambda} = 50$ NZ $\approx 0\%$
Time(s)	Parametric	1.14	1.06	1.04	0.98	0.95	0.91
	Gurobi	16.12	TL	TL	TL	TL	7.63
B&B nodes	Gurobi	2778.8	5,471,547	5,655,580	6,577,235	8,350,992	99.4
Opt. gap (%)		0.01	0.10	0.60	7.11	42.84	0.00

TL: Time Limit (1 h), NZ: percentage of non-zero elements in the optimal solution $x^{\star}$. The reported results are averaged over 5 trials
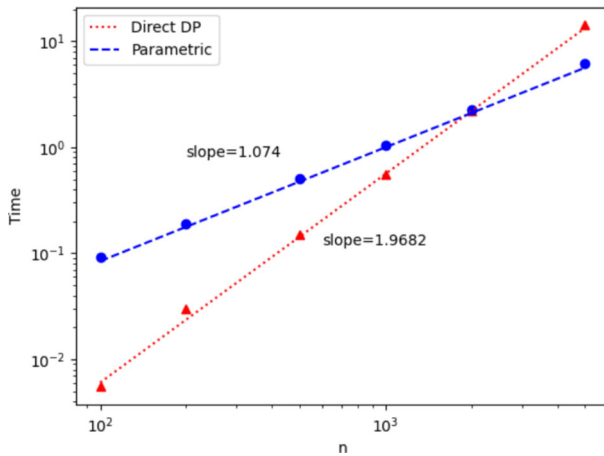
Fig. 12 The runtime of the parametric algorithm (Algorithm 1) and the direct DP approach of [40] for instances with path structure. The reported results are averaged over 5 trials

We utilize the same dataset in our study. The signal comprises 13,800 readings indicating changes in "x acceleration" for a participant. The participant's activity sequence is as follows: they were "working at a computer" until timestamp 4,415; then engaged in "standing up, walking, and going upstairs" until timestamp 4,735; followed by "standing" from timestamp 4,735 to 5,854, from 8,072 to 9,044, and again from 9,045 to 9,720. Subsequently, they were "walking" from timestamp 5,854 to 8,072; involved in "going up or down stairs" from timestamp 9,044 to 9,435; "walking and talking with someone" from timestamp 9,720 to 10,430; and "talking while standing" from timestamp 10,457 to 13,800 (with the status between timestamps 10,430 and 10,457 being unknown).

This problem can be formulated as an instance of Problem (3), where the hidden state x_t represents the activity level of the participant. Specifically, intervals characterized by minimal or absent physical activity naturally correspond to time stamps t where $x_t = 0$. Furthermore, we segment the signal into windows of magnitude K and regard each segment t as the observation set for the hidden state x_t. More precisely, we treat $y_{(t-1)K+1}, \ldots, y_{tK}$ as the observations corresponding to the hidden state x_t.

Additionally, we assume that a subset of the observations is corrupted with outlier noise. As discussed in Sect. 1.1, the inference of a GHMM with outliers (referred to as robust inference hereafter) can be addressed by solving Problem (3). Since this problem has a tree structure, it can be solved via the parametric algorithm. In this context, the scale of the problems being addressed exceeds $n = 30,000$. At such scales, Gurobi fails to yield a reliable solution. Alternatively, in scenarios where the observations are assumed to be free of outliers, the variables w and z in Problem (3) can be set to zero. This transformation simplifies the problem into one defined over a path graph, which can be solved using the parametric algorithm over path graphs (Algorithm 1) or the direct DP approach proposed in [40].

Figure 13 depicts the robust and non-robust inference of the hidden signal for $K = 10$. It is evident that the original signal is corrupted with outlier noise, with the most
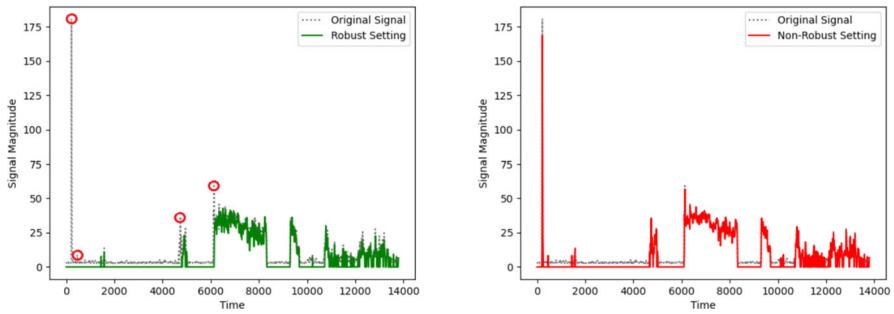
Fig. 13 Robust and non-robust inference of the hidden signal. In the figure on the left, the outliers removed from the signal are circled in red. The parameters in this experiment are set to $\gamma_t = 400$, $\lambda_{k,t} = 100$, $\sigma_t^2 = 2$, and $\nu_t = 1$ (color figure online)
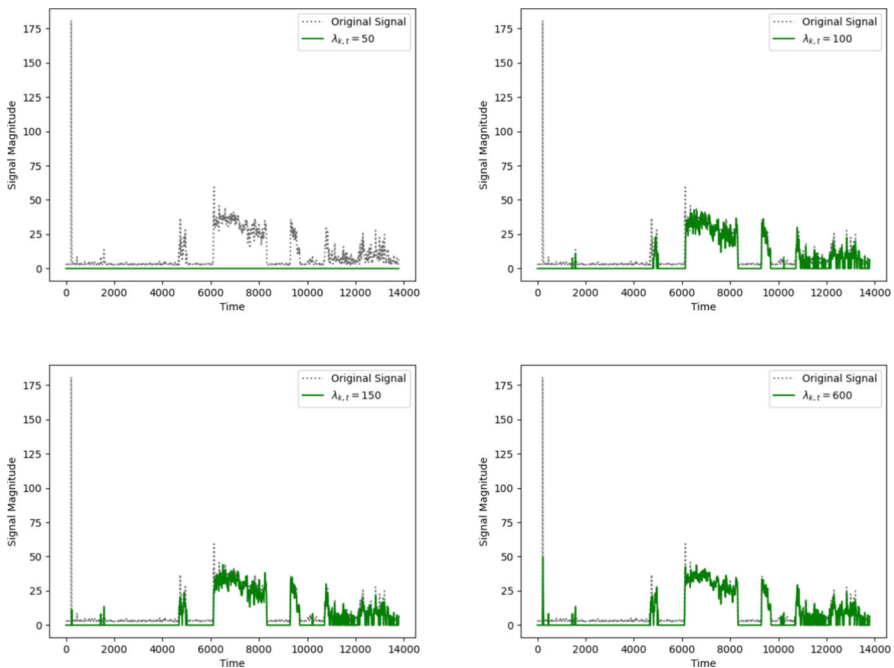

Fig. 14 The recovered signal for $\lambda_{k,t} \in \{50, 100, 150, 600\}$. The other parameters are set to $\gamma_t = 400$, $\sigma_t^2 = 2$, $\nu_t = 1$, and $K = 10$

significant outlier appearing at timestamp 250. While the robustly recovered signal successfully filters out the outliers, its non-robust counterpart fails to remove them. In these experiments, our parametric algorithm solves the robust inference problem within 46.4 s, whereas the non-robust inference is solved within 1.2 s. This disparity in runtimes is not surprising, given that the robust inference problem is nearly 11 times larger.

Figure 14 depicts the impact of the regularization parameter $\lambda_{k,t}$ on the recovered signal. A small value of $\lambda_{k,t}$ results in a fully dense w, effectively treating the entire
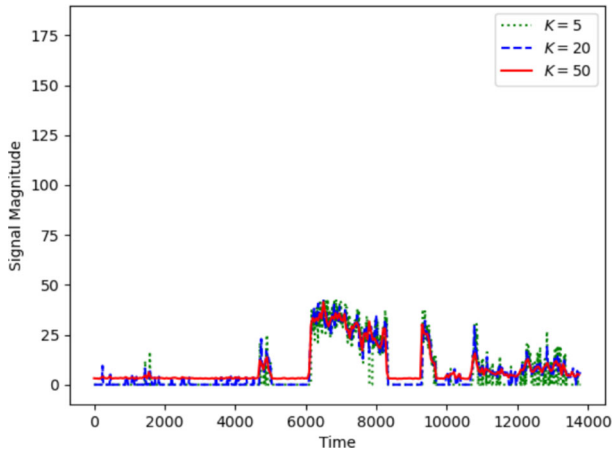
Fig. 15 The recovered signal for three values of K. The parameters are set to $\gamma_t = 250$, $\lambda_{k,t} = 100$, $\sigma_t^2 = 2$, and $\nu_t = 1$

(a) L_1 relaxation   (b) Wavelet denoising   (c) Low-pass filter
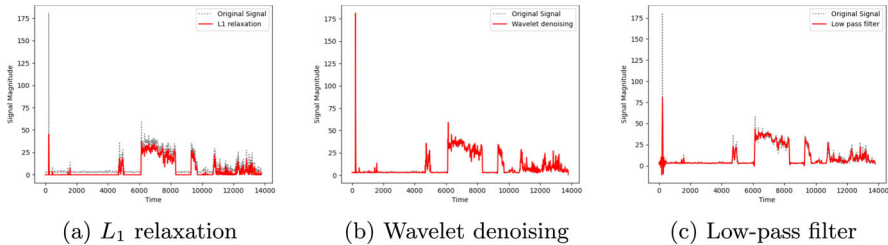
Fig. 16 The recovered signal obtained using three methods: L_1 relaxation, wavelet denoising, and the low-pass filter

observations as corrupted by outlier noise. Conversely, a larger $\lambda_{k,t}$ enforces sparser w, indicating that most observations are assumed to be free of outlier noise.

Finally, Fig. 15 illustrates the impact of varying values of the partition size K on the recovered signal. Recall that K represents the number of observations for each hidden state. As a result, a larger K typically improves the smoothness of the recovered signal but could potentially obscure finer changes. This phenomenon is shown in Fig. 15.

As previously mentioned, the optimization problem for inferring GHMMs in (3) is nonconvex due to the combinatorial nature of the sparsity-promoting regularizer, which is commonly referred to as the L_0 penalty. To handle this nonconvexity, a standard approach is to relax the L_0 penalty to its convex L_1 surrogate [54]. Figure 16a demonstrates the performance of the L_1 relaxation, with regularization parameters determined through a grid search over $\lambda_{k,t}$ and γ_t. As shown, while the L_1 relaxation imposes sparsity, it does not effectively eliminate outliers and results in substantial shrinkage of the non-zero values.

Additionally, we compare our method to two commonly used algorithms for human activity recognition: signal denoising using the wavelet transform and low-pass filters [26, 38]. These methods were implemented using the *Signal Processing Toolbox* [45]
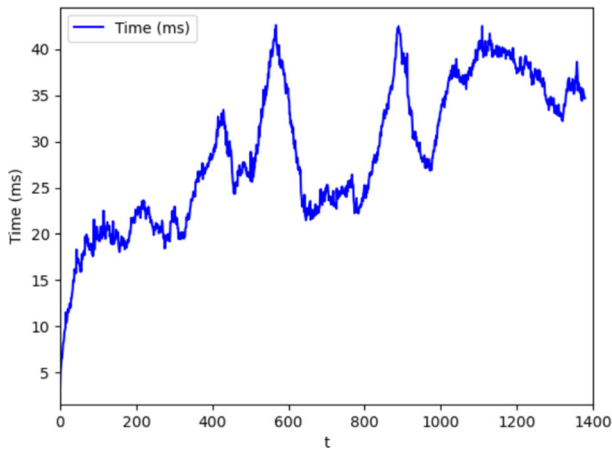
Fig. 17 The update time of the 5 most recent hidden states after the arrival of $K = 10$ observations. The other parameters are set to $\gamma_t = 250, \lambda_{k,t} = 100, \sigma_t^2 = 2, \nu_t = 1$

and *Wavelet Toolbox* [42] in MATLAB 2020b. As shown in Fig. 16b, c, the results indicate that these techniques are ineffective at removing outliers.

Online setting Finally, we consider the online setting, where the goal is to infer the values of the hidden state x_t, as the new collected data from the accelerometer arrives "on-the-go". More specifically, at each new timestep $t = 1, \ldots, T$, K new observations are revealed, and the goal is to infer the value of x_t, and possibly update the values of S most recent values $x_{t-1}, \ldots, x_{t-S}$ based on the newly observed data. Note that new observations at current time t not only help with the inference of the current hidden state x_t, but also can potentially change the optimal value of the past hidden states $x_{t-1}, \ldots, x_1$. Consequently, the optimal inference of the hidden state necessitates resolving a sequence of optimization problems with the new incoming data.

Thanks to our parametric approach, we achieve this goal in *milliseconds*. To see this, note that our parametric algorithm performs inference by sequentially obtaining $f_{x_1}, \ldots, f_{x_{t-1}}$ corresponding to the parametric costs at the hidden states $x_1, \ldots, x_{t-1}$, along with their conjugates (refer to Fig. 1 for an illustration of the associated graph). Therefore, according to the recursive equation (9), the parametric cost f_{x_t} at the new time t can be efficiently characterized merely based on the conjugate functions $f^*_{x_{t-1}}$ (which is already computed and available) and $\{f^*_{y_{k,t}}\}_{k=1}^K$, thus circumventing the need to resolve Problem (3) from scratch. Once the parametric cost f_{x_t} is obtained, the hidden states $x_t, \ldots x_{t-S}$ can be updated in $\mathcal{O}(S)$, according to Algorithm 2. Figure 17 illustrates the runtime of this online version of our algorithm. At any given time t, the optimal cost, along with the updated values of $x_t, \ldots, x_{t-4}$ are obtained based on $K = 10$ new observations within at most 45 ms.

6 Conclusions

In this paper, we consider mixed-integer quadratic programs with indicators where the matrix defining the quadratic term, Q, has a tree structure. While for general Q the problem is NP-hard, we propose a highly efficient algorithm for the tree-structured Q. Our algorithm has a time and memory complexity of $\mathcal{O}(n^2)$ that maintains the same complexity as the simpler path-structured problem studied earlier. Our computational results show that the practical complexity of the algorithm on our test instances is almost linear. Our algorithm can be leveraged in problems where the Q matrix can be decomposed into trees in a similar procedure proposed in [40].

Appendix A: Proof of Lemma 1

Let J be the set of nodes in $\mathrm{supp}_u(Q)$, excluding u. Let us define $p_{u,s}(\alpha)$:

$$p_{u,s}(\alpha) = \min_{x \in \mathbb{R}^{n_u-1}} \frac{1}{2}\alpha^2 + c_u\alpha + \left(\frac{1}{2}x^\top Q_{J,J}x + \alpha Q_{u,J}^\top x + c_J^\top x + \lambda_J^\top s\right)$$
$$\text{s.t. } x_i(1 - s_i) = 0 \quad i = 1, 2 \ldots n_u - 1.$$

It is easy to verify that $f_u(\alpha) = \min_{s \in \{0,1\}^{n_u-1}}\{p_{u,s}(\alpha)\} + \lambda_u \mathbb{1}_\alpha$. Therefore, it remains to show that for every $s \in \{0, 1\}^{n_u-1}$, $p_{u,s}(\alpha)$ is strongly convex and quadratic. To establish this, we use the Karush-Kuhn-Tucker (KKT) conditions to calculate the solution of the optimization problem. First, let $J_s = \{i : i \in J \text{ and } s_i = 1\}$. Since the constraints in the definition of $p_{u,s}(\alpha)$ sets some of the coordinates of x to 0, the optimization problem can be simplified as follows

$$p_{u,s}(\alpha) = \min_{x \in \mathbb{R}^{|J_s|}} \frac{1}{2}\alpha^2 + c_u\alpha + \left(\frac{1}{2}x^\top Q_{J_s,J_s}x + \alpha Q_{u,J_s}^\top x + c_{J_s}^\top x + \sum_{i \in J_s}\lambda_i\right).$$

From KKT conditions it follows that

$$p_{u,s}(\alpha) = \frac{1}{2}\left(1 - Q_{u,J_s}\left(Q_{J_s,J_s}\right)^{-1}Q_{u,J_s}^\top\right)\alpha^2 + \left(c_u - c_{J_s}^T\left(Q_{J_s,J_s}\right)^{-1}Q_{u,J_s}^\top\right)\alpha$$

$$+ \left(-\frac{1}{2}c_{J_s}^\top(Q_{J_s,J_s})^{-1}c_{J_s} + \sum_{i \in J_s}\lambda_i\right).$$

Note that $\left(1 - Q_{u,J_s}\left(Q_{J_s,J_s}\right)^{-1}Q_{u,J_s}^\top\right)$ is the Schur complement of $Q_{J_s \cup \{u\}, J_s \cup \{u\}}$, which, owing to the positive definiteness of Q, is positive definite.

Therefore, $\left(1 - Q_{u,J_s}\left(Q_{J_s,J_s}\right)^{-1} Q_{u,J_s}^{\top}\right) > 0$, implying that $p_{u,s}(\alpha)$ is strongly convex. This completes the proof. □

Appendix B: Proof of Lemma 2

Let $-\infty = \tau_0 < \tau_1 < \cdots < \tau_N = +\infty$ be the ordered elements of $\bigcup_{l=1}^{L} \Gamma_l$. The proof follows by noting that none of $\{f_l\}_{l=1}^{L}$ contain any breakpoints within the interval (τ_{k-1}, τ_k); $k = 1, \ldots, N$. Therefore, the set of breakpoints of g can only belong to $\{\tau_0, \ldots, \tau_N\}$. It follows that g is a piece-wise quadratic function with at most N pieces.

Next, we discuss the complexity of obtaining g. First, we combine the sorted lists $\{\Gamma_l\}_{l=1}^{L}$ and represent them in a single sorted list, which can be done in $\mathcal{O}(N \log L)$ using L-way Merge algorithm [25, Theorem 1]. Second, for each interval, g can be constructed by summing the corresponding pieces of the functions $\{f_l\}_{l=1}^{L}$. This leads to the overall complexity of $\mathcal{O}(NL)$, where $N = |\bigcup_{l=1}^{L} \Gamma_l|$. □

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows (1988)
2. Atamtürk, A., Gómez, A.: Strong formulations for quadratic optimization with M-matrices and indicator variables. Math. Program. **170**(1), 141–176 (2018)
3. Atamtürk, A., Gómez, A., Han, S.: Sparse and smooth signal estimation: convexification of ℓ_0-formulations. J. Mach. Learn. Res. **22**(1), 2370–2412 (2021)
4. Bertsekas, D.P., Yu, H.: A unifying polyhedral approximation framework for convex optimization. SIAM J. Optim. **21**(1), 333–360 (2011)
5. Bertsimas, D., Cory-Wright, R., Pauphilet, J.: A new perspective on low-rank optimization. Math. Program. **202**, 47–92 (2023)
6. Bertsimas, D., King, A., Mazumder, R.: Best subset selection via a modern optimization lens. Ann. Stat. **44**(2), 813–852 (2016)
7. Bertsimas, D., Parys, B.V.: Sparse high-dimensional regression: exact scalable algorithms and phase transitions. Ann. Stat. **48**(1), 300–323 (2020)
8. Besag, J.: Spatial interaction and the statistical analysis of lattice systems. J. R. Stat. Soc.: Ser. B (Methodol.) **36**(2), 192–225 (1974)
9. Besag, J., Kooperberg, C.: On conditional and intrinsic autoregressions. Biometrika **82**(4), 733–746 (1995)
10. Brown, R.G., Hwang, P.Y.: Introduction to Random Signals and Applied Kalman Filtering, vol. 3. Wiley, New York (1992)
11. Casale, P., Pujol, O., Radeva, P.: Human activity recognition from accelerometer data using a wearable device. In: Pattern Recognition and Image Analysis: 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings 5. pp. 289–296. Springer (2011)

12. Casale, P., Pujol, O., Radeva, P.: Personalization and user verification in wearable systems using biometric walking patterns. Pers. Ubiquit. Comput. **16**, 563–580 (2012)
13. Ceria, S., Soares, J.: Convex programming for disjunctive convex optimization. Math. Program. **86**, 595–614 (1999)
14. Chakrabarty, D., Lee, Y.T., Sidford, A., Wong, S.C.W.: Subquadratic submodular function minimization. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 1220–1231 (2017)
15. Chang, I., Tiao, G.C., Chen, C.: Estimation of time series parameters in the presence of outliers. Technometrics **30**(2), 193–204 (1988)
16. Chen, X., Ge, D., Wang, Z., Ye, Y.: Complexity of unconstrained minimization. Math. Program. **143**(1–2), 371–383 (2014)
17. Das, A., Kempe, D.: Algorithms for subset selection in linear regression. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 45–54 (2008)
18. Dedieu, A., Hazimeh, H., Mazumder, R.: Learning sparse classifiers: continuous and mixed integer optimization perspectives. J. Mach. Learn. Res. **22**(1), 6008–6054 (2021)
19. Del Pia, A., Dey, S.S., Weismantel, R.: Subset selection in sparse matrices. SIAM J. Optim. **30**(2), 1173–1190 (2020)
20. Fattahi, S., Gomez, A.: Scalable inference of sparsely-changing gaussian Markov random fields. Adv. Neural. Inf. Process. Syst. **34**, 6529–6541 (2021)
21. Fattahi, S., Gómez, A.: Solution path of time-varying Markov random fields with discrete regularization. arXiv preprint arXiv:2307.13750 (2023)
22. Gómez, A.: Outlier detection in time series via mixed-integer conic quadratic optimization. SIAM J. Optim. **31**(3), 1897–1925 (2021)
23. Gómez, A., Neto, J.: Outlier detection in regression: conic quadratic formulations. arXiv preprint arXiv:2307.05975 (2023)
24. Gómez, A., Xie, W.: A note on quadratic constraints with indicator variables: convex hull description and perspective relaxation. Oper. Res. Lett. **52**, 107059 (2024)
25. Greene, W.A.: k-way merging and k-ary sorts. In: ACM Southeast Conference, pp. 127–135 (1993)
26. Gu, F., Chung, M.H., Chignell, M., Valaee, S., Zhou, B., Liu, X.: A survey on deep learning for human activity recognition. ACM Comput. Surv. **54**(8), 1–34 (2021)
27. Günlük, O., Linderoth, J.: Perspective reformulations of mixed integer nonlinear programs with indicator variables. Math. Program. **124**, 183–205 (2010)
28. Günlük, O., Linderoth, J.: Perspective reformulation and applications. In: Mixed Integer Nonlinear Programming, pp. 61–89. Springer (2011)
29. Han, S., Gómez, A.: Compact extended formulations for low-rank functions with indicator variables. Math. Oper. Res. (2024), to appear. arXiv preprint arXiv:2110.14884
30. Han, S., Gómez, A., Pang, J.S.: On polynomial-time solvability of combinatorial Markov random fields. arXiv preprint arXiv:2209.13161 (2022)
31. Hastie, T., Tibshirani, R., Tibshirani, R.J.: Extended comparisons of best subset selection, forward stepwise selection, and the lasso. arXiv preprint arXiv:1707.08692 (2017)
32. Hazimeh, H., Mazumder, R., Saab, A.: Sparse regression at scale: branch-and-bound rooted in first-order optimization. Math. Program. **196**(1–2), 347–388 (2022)
33. Huo, X., Chen, J.: Complexity of penalized likelihood estimation. J. Stat. Comput. Simul. **80**(7), 747–759 (2010)
34. Insolia, L., Kenney, A., Chiaromonte, F., Felici, G.: Simultaneous feature selection and outlier detection with optimality guarantees. Biometrics **78**(4), 1592–1603 (2022)
35. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
36. Kim, Y.J., Kang, B.N., Kim, D.: Hidden Markov model ensemble for activity recognition using tri-axis accelerometer. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics, pp. 3036–3041. IEEE (2015)
37. Küçükyavuz, S., Shojaie, A., Manzour, H., Wei, L., Wu, H.H.: Consistent second-order conic integer programming for learning Bayesian networks. J. Mach. Learn. Res. **24**(322), 1–38 (2023)
38. Kumar, P., Chauhan, S., Awasthi, L.K.: Human activity recognition (har) using deep learning: review, methodologies, progress and future research directions. Arch. Comput. Methods Eng. **31**(1), 179–219 (2024)

39. Lee, Y.T., Sidford, A., Wong, S.C.W.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 1049–1065. IEEE (2015)

40. Liu, P., Fattahi, S., Gómez, A., Küçükyavuz, S.: A graph-based decomposition method for convex quadratic optimization with indicators. Math. Program. **200**(2), 669–701 (2023)

41. Manzour, H., Küçükyavuz, S., Wu, H.H., Shojaie, A.: Integer programming for learning directed acyclic graphs from continuous data. INFORMS J. Optimiz. **3**(1), 46–73 (2021)

42. Misiti, M., Misiti, Y., Oppenheim, G., Poggi, J.M.: Wavelet Toolbox. The MathWorks Inc., Natick, MA **15**, 21 (1996)

43. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Math. Program. **118**(2), 237–251 (2009)

44. Ravikumar, V., Xu, T., Al-Holou, W.N., Fattahi, S., Rao, A.: Efficient inference of spatially-varying Gaussian Markov random fields with applications in gene regulatory networks. IEEE/ACM Trans. Comput. Biol. Bioinform. (2023)

45. Stearns, S.D., David, R.A.: Signal Processing Algorithms in Matlab. Prentice-Hall, Inc (1996)

46. Stubbs, R.A.: Branch-and-Cut Methods for Mixed 0–1 Convex Programming. Northwestern University (1996)

47. Trabelsi, D., Mohammed, S., Chamroukhi, F., Oukhellou, L., Amirat, Y.: An unsupervised approach for automatic activity recognition based on hidden Markov model regression. IEEE Trans. Autom. Sci. Eng. **10**(3), 829–835 (2013)

48. Tsay, R.S.: Time series model specification in the presence of outliers. J. Am. Stat. Assoc. **81**(393), 132–141 (1986)

49. Wei, L., Atamtürk, A., Gómez, A., Küçükyavuz, S.: On the convex hull of convex quadratic optimization problems with indicators. Math. Program. **204**(1–2), 703–737 (2024)

50. Wei, L., Gómez, A., Küçükyavuz, S.: Ideal formulations for constrained convex optimization problems with indicator variables. Math. Program. **192**(1–2), 57–88 (2022)

51. Wei, L., Gómez, A., Küçükyavuz, S.: On the convexification of constrained quadratic optimization problems with indicator variables. In: International Conference on Integer Programming and Combinatorial Optimization. pp. 433–447. Springer (2020)

52. Xie, W., Deng, X.: Scalable algorithms for the sparse ridge regression. SIAM J. Optim. **30**, 3359–3386 (2020)

53. Yan, H., Grasso, M., Paynabar, K., Colosimo, B.M.: Real-time detection of clustered events in video-imaging data with applications to additive manufacturing. IISE Trans. **54**(5), 464–480 (2022)

54. Zhang, C.H., Huang, J.: The sparsity and bias of the lasso selection in high-dimensional linear regression. Ann. Stat. **36**(4), 1567–1594 (2008)