A polynomial-time solvable class of sparse box-constrained polynomial optimization problems *

Aida Khajavirad †

August 19, 2026

Abstract

We study the problem of minimizing a multivariate polynomial function over the unit hypercube. Exploiting sparsity in the interaction graph or hypergraph, we identify variables that can be restricted to binary values at optimality and eliminate the remaining continuous variables component-wise, reducing the problem to structured binary polynomial optimization. For quadratic objectives, we obtain exact polynomial-time solvability in the standard Turing bit model under conditions involving treewidth and the nonconvexity of the continuous components. For higher-degree objectives, we obtain an exact polynomial-time algorithm in the unit-cost algebraic RAM model under logarithmic incidence treewidth and small, weakly coupled continuous components.

Keywords: box-constrained polynomial optimization, binary polynomial optimization, sparsity, hypergraph, treewidth, polynomial-time algorithm.

1 Introduction

We consider the problem of minimizing a multivariate polynomial function of degree $d \geq 2$ over the unit hypercube. Let $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{Z}^n_{\geq 0}$. Define $x^\alpha := x_1^{\alpha_1} \dots x_n^{\alpha_n}$ and $|\alpha| := \sum_{i=1}^n \alpha_i$. We define a *box-constrained polynomial optimization problem* of degree d as follows:

$$\begin{aligned} \min \ & \sum_{|\alpha| \leq d} c_\alpha x^\alpha \\ \text{s.t.} \ & x \in [0,1]^n, \end{aligned} \tag{1}$$

where $c_\alpha \in \mathbb{Q}$ for all $\alpha \in \mathbb{Z}^n_{\geq 0}$ with $|\alpha| \leq d$, and we assume that $c_\alpha \neq 0$ for at least one α with $|\alpha| = d$. Throughout the paper, we assume that the input polynomial is given explicitly in the standard monomial basis, that is, as a list of pairs (α, c_α) with $c_\alpha \neq 0$. The input length $\mathcal{L}$ denotes the total number of bits required to encode the nonzero coefficients and their corresponding exponent vectors. We assume that the degree d is polynomially bounded in $\mathcal{L}$; that is, $d \in O(\mathcal{L}^c)$ for some fixed constant $c > 0$. In particular, d is not fixed and may grow with the instance. Problem (1) is $\mathcal{NP}$-hard even for $d = 2$, as it contains the maximum cut problem as a special case. If $d = 2$ and the problem is convex, i.e., if the Hessian of the quadratic function is positive semidefinite, then thanks to the ellipsoid method, Problem (1) can be solved in time polynomial in $\mathcal{L}$ [21]. If $d \geq 3$ and the polynomial function is convex over the unit hypercube, then an ϵ-optimal

*The author was supported in part by AFOSR grant FA9550-23-1-0123 and ONR grant N00014-25-1-2491.

†Department of Industrial and Systems Engineering, Lehigh University. E-mail: aida@lehigh.edu.

solution can be computed in time polynomial in $\mathcal{L}$ and $\log(1/\epsilon)$ using the ellipsoid method [14] or interior point methods [18]. However, if Problem (1) is nonconvex, then its tractability has only been established in fixed dimension. Namely, if n is fixed, then Problem (1) is solvable in time polynomial in $\mathcal{L}$ via quantifier-elimination techniques [19].

Binary polynomial optimization. In contrast to the continuous setting, the complexity of binary polynomial optimization, i.e., the problem of minimizing a multivariate polynomial over the set of binary points, has been extensively studied in the literature [8, 4, 12, 9, 11, 6]. To formally define this problem, we use the hypergraph representation scheme of [10]. A hypergraph G is a pair (V, E), where V is a finite set of nodes and E is a set of subsets of V of cardinality at least two, called the edges of G. Define $d := \max_{e \in E} |e|$. With any hypergraph $G = (V, E)$ and the cost vector $c \in \mathbb{Q}^{V \cup E}$, we associate the following *binary polynomial optimization problem of degree* d:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \prod_{i \in e} z_i + \sum_{i \in V} c_i z_i \\ \text{s.t.} \quad & z \in \{0,1\}^V, \end{aligned} \tag{2}$$

where, without loss of generality, we assume that each node is contained in at least one edge, and $c_e \neq 0$ for all $e \in E$. Problem (2) is $\mathcal{NP}$-hard even for $d = 2$. If the objective function of Problem (2) is submodular, then this problem can be solved in strongly polynomial time [20]. By strongly polynomial time, we mean time that is polynomial in $|V|, |E|$. Given a hypergraph $G = (V, E)$, the *intersection graph of* G is the graph with node set V, and where two nodes $i, j \in V$ are adjacent if $i, j \in e$ for some $e \in E$. Henceforth, we refer to the treewidth of the intersection graph of a hypergraph G as the *primal treewidth* of G and denote it by $\text{ptw}(G)$. In [8], the authors prove that if $\text{ptw}(G)$ is bounded, then Problem (2) can be solved in strongly polynomial time. Given a hypergraph $G = (V, E)$, the *incidence graph of* G is a bipartite graph whose vertex set is $V \cup E$ and the edge set is $\{\{i, e\} : i \in V,\ e \in E,\ i \in e\}$. Henceforth, we refer to the treewidth of the incidence graph of a hypergraph G as the *incidence treewidth* of G and denote it by $\text{itw}(G)$. For any hypergraph G, we have $\text{itw}(G) \leq \text{ptw}(G)$. In [6], the authors prove that if $\text{itw}(G)$ is bounded, then Problem (2) can be solved in strongly polynomial time. Finally, in [9], the authors prove that if the hypergraph G is β-acyclic, then Problem (2) can be solved in strongly polynomial time. Note that the incidence treewidth of a β-acyclic hypergraph may be unbounded, in general.

Let us consider the important special case of Problem (2) with $d = 2$. With any graph $G = (V, E)$ and cost vector $c \in \mathbb{Q}^{V \cup E}$, we associate the *binary quadratic optimization problem*:

$$\begin{aligned} \min \quad & \sum_{\{i,j\} \in E} c_{ij} z_i z_j + \sum_{i \in V} c_i z_i \\ \text{s.t.} \quad & z \in \{0,1\}^V. \end{aligned} \tag{3}$$

First, for a graph G we have $\text{ptw}(G) = \text{itw}(G)$. Second, for a graph, the notion of β-acyclicity in hypergraphs coincides with the usual notion of graph acyclicity. Therefore, for Problem (3) the most general sufficient condition for polynomial-time solvability in terms of treewidth is as follows: if the graph G has bounded treewidth, then Problem (3) can be solved in strongly polynomial time. In fact, the results of [7, 11] imply that under certain complexity assumptions, bounded treewidth is essentially a necessary and sufficient condition for polynomial-time solvability of Problem (3).

Our contributions. In this paper, by building on existing algorithms for binary polynomial optimization and box-constrained polynomial optimization, we obtain new sufficient conditions for

the exact solution of Problem (1). For quadratic objectives, our algorithm runs in time polynomial in $\mathcal{L}$ in the standard Turing bit model. For higher-degree objectives, we obtain a polynomial-time algorithm in the unit-cost algebraic RAM model.

First, we focus on the important special case of Problem (1) with $d = 2$; namely, the *box-constrained quadratic optimization problem*:

$$\begin{aligned} \min \quad & x^\top Q x + c^\top x \\ \text{s.t.} \quad & x \in [0,1]^n, \end{aligned} \tag{4}$$

where $c \in \mathbb{Q}^n$ and $Q \in \mathbb{Q}^{n \times n}$ is a symmetric matrix. In [15], the authors prove that if the rank of Q is fixed, then Problem (4) can be solved in polynomial time. We partition the variables according to the sign of the diagonal entries of Q: variables x_i with $q_{ii} \leq 0$ can be restricted to binary values in an optimal solution. Henceforth, we refer to these variables as *hidden binary variables*, while the remaining variables form the continuous part. We decompose the continuous variables into connected components of the interaction graph and eliminate these components independently. We show that the complexity of this elimination can be controlled by a parameter measuring the nonconvexity of each continuous component, and obtain sufficient conditions involving this parameter and the treewidth of the resulting interaction graph under which Problem (4) can be solved in polynomial time in the standard Turing bit model. The elimination produces a binary polynomial optimization problem with logarithmically bounded treewidth, which can then be solved efficiently by dynamic programming (see Theorem 1 and Corollary 1).

Next, we extend this approach to higher-degree box-constrained polynomial optimization problems. We again separate variables into a hidden binary part and a continuous part and decompose the continuous variables into connected components of the "interaction" hypergraph. When each continuous component has constant size and directly interacts with only a logarithmic number of binary variables, and when the incidence treewidth of the interaction hypergraph is logarithmically bounded, we can eliminate each continuous component locally using polynomial-time algorithms for fixed-dimensional polynomial optimization from real algebraic geometry. Unlike in the quadratic case, the optimal values obtained from these local continuous problems need not be rational; in general, they are real algebraic numbers. We therefore formulate the higher-degree result in the unit-cost algebraic RAM model. Using existing algorithms for binary polynomial optimization and exact sign determination of real algebraic expressions, we show that the resulting problem can be solved exactly in polynomial time in this model (see Theorem 2).

Related to our use of sparsity, Bienstock and Muñoz [2] show that box-constrained polynomial optimization problems with bounded primal treewidth admit polynomial-size linear programming approximations that, for any prescribed tolerance, yield a feasible point whose objective value is additively close to the global optimum, whereas our focus is on sufficient conditions for exact polynomial-time solvability.

Organization. The remainder of the paper is structured as follows. In Section 2 we obtain sufficient conditions under which box-constrained quadratic optimization problems can be solved in polynomial time in the standard Turing bit model. In Section 3 we obtain sufficient conditions under which higher-degree box-constrained polynomial optimization problems can be solved in polynomial time in the unit-cost algebraic RAM model.

2 Box-constrained quadratic optimization

In this section, we consider the box-constrained quadratic optimization problem defined by (4). To exploit the sparsity of the objective function, we define an *interaction graph* $G = (V, E)$ for this problem as follows. For each variable x_i, $i \in [n] := \{1, \cdots, n\}$, we define a node i. Two distinct nodes i and j are adjacent if the coefficient q_{ij} is nonzero. Henceforth, given a graph G, we denote by $\mathrm{tw}(G)$ the treewidth of G.

The computational model. Throughout this section, running times are measured in the standard Turing bit model. All numerical input data are rational and encoded in binary, and $\mathcal{L}$ denotes their total binary encoding length. Thus, whenever we refer to polynomial time in this section, we mean time polynomial in $\mathcal{L}$, accounting for the bit complexity of all arithmetic operations.

2.1 Statement of main results

We are now ready to state our sufficient conditions for polynomial-time solvability of Problem (4).

Theorem 1. *Let $G = (V, E)$ be the interaction graph of Problem (4). Define*

$$V^+ := \{i \in V : q_{ii} > 0\}. \tag{5}$$

Denote by G_{V^+} the subgraph of G induced by V^+, and denote by $\mathcal{C}$ the set of connected components of G_{V^+}. For each connected component $C \in \mathcal{C}$, define its neighborhood as:

$$N(C) := \{i \in V \setminus C : \{i, j\} \in E, \text{ for some } j \in C\}. \tag{6}$$

Let Q_C denote the principal submatrix of Q indexed by C, and let κ_C be the smallest integer in $\{0, \ldots, |C| - 1\}$ such that every $(|C| - \kappa_C) \times (|C| - \kappa_C)$ principal submatrix of Q_C is positive semidefinite. Denote by $G_{V \setminus V^+}$ the subgraph of G induced by $V \setminus V^+$, and denote by $\bar{G}$ the graph on $V \setminus V^+$ obtained from $G_{V \setminus V^+}$ by making each $N(C)$ a clique. Suppose that the following conditions are satisfied:

(i) $\mathrm{tw}(\bar{G}) \in O(\log |V|)$;
(ii) $\kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) \in O(\log |V|)$ *for all* $C \in \mathcal{C}$, *where we define* $0 \log(1 + \frac{|C|}{0}) := 0$.

Then Problem (4) can be solved in time polynomial in the input length $\mathcal{L}$ in the Turing bit model.

Remark 1. Let us consider the two assumptions in Theorem 1; these assumptions control two different sources of complexity in Problem (4). Namely, assumption (i) is concerned with the difficulty of the combinatorial component of Problem (4), while assumption (ii) is concerned with the difficulty of its continuous component. Since for each $C \in \mathcal{C}$, $N(C)$ is a clique of the graph $\bar{G}$, assumption (i) implies that $|N(C)| \in O(\log |V|)$ for all $C \in \mathcal{C}$. On the other hand, assumption (ii) has a natural interpretation depending on the size of κ_C relative to $|C|$. In the worst case where $\kappa_C = \Theta(|C|)$, we have $\kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) = \Theta(|C|)$, and hence assumption (ii) becomes $|C| \in O(\log |V|)$. On the other hand, if $\kappa_C = \Theta(1)$, then $\kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) = \Theta(\log |C|)$. Since $|C| \leq |V|$, this term is automatically in $O(\log |V|)$. Therefore, in this regime, assumption (ii) places no restriction on the size of C and is in fact redundant.

As we will detail in the next section, one can obtain the following more restrictive sufficient condition whose treewidth assumption is stated directly in terms of the original graph G.

Corollary 1. *Let $G = (V, E)$ be the interaction graph of Problem* (4)*. Denote by G_{V^+} the subgraph of G induced by V^+, where V^+ is defined by* (5)*, and denote by $\mathcal{C}$ the set of connected components of G_{V^+}. Let Q_C denote the principal submatrix of Q indexed by C, and let κ_C be the smallest integer in $\{0, \dots, |C| - 1\}$ such that every $(|C| - \kappa_C) \times (|C| - \kappa_C)$ principal submatrix of Q_C is positive semidefinite. Suppose that the following conditions are satisfied:*

(i) $\operatorname{tw}(G) \in O(\log |V|)$*;*

(ii) $|N(C)| + \kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) \in O(\log |V|)$ *for all $C \in \mathcal{C}$, where $N(C)$ is defined by* (6) *and where we define* $0 \log(1 + \frac{|C|}{0}) := 0$*.*

Then Problem (4) *can be solved in time polynomial in the input length $\mathcal{L}$ in the Turing bit model.*

Remark 2. The structural quantities appearing in the assumptions of Theorem 1 can be computed in time polynomial in $\mathcal{L}$ for instances satisfying these assumptions. The interaction graph $G = (V, E)$ can be constructed in polynomial time by inspecting the nonzero entries of the matrix Q. The set V^+ can be identified by checking the signs of the diagonal entries q_{ii}. The connected components of the induced subgraph G_{V^+} and the corresponding neighborhoods $N(C)$ can be computed using standard graph traversal algorithms in time $O(|V| + |E|)$. For each $C \in \mathcal{C}$, the parameter κ_C can be determined by considering, for $r = 0, 1, \dots, |C| - 1$, all principal submatrices of Q_C of order $|C| - r$ and finding the smallest r for which all such submatrices are positive semidefinite. If $\kappa_C = 0$, only the matrix Q_C needs to be checked. Now suppose that $\kappa_C \geq 1$. The total number of principal submatrices that need to be considered is at most

$$\sum_{r=0}^{\kappa_C} \binom{|C|}{r} \leq (\kappa_C + 1) \left(\frac{e|C|}{\kappa_C}\right)^{\kappa_C}.$$

Since $\kappa_C \leq |C|$, we have $\kappa_C + 1 \leq \left(1 + \frac{|C|}{\kappa_C}\right)^{\kappa_C}$ and $\frac{e|C|}{\kappa_C} \leq \left(1 + \frac{|C|}{\kappa_C}\right)^2$. Therefore,

$$\sum_{r=0}^{\kappa_C} \binom{|C|}{r} \leq \left(1 + \frac{|C|}{\kappa_C}\right)^{3\kappa_C} = |V|^{O(1)},$$

where the last equality follows from assumption (ii). Positive semidefiniteness of each principal submatrix can be checked in polynomial time. Hence, under assumption (ii), all parameters κ_C can be computed in polynomial time. The graph $\bar{G}$ can be constructed in polynomial time from G and the sets $N(C)$. Moreover, although computing treewidth exactly is $\mathcal{NP}$-hard, there exists an algorithm that, given a graph $\bar{G}$ and an integer k, runs in time $2^{O(k)}|V|$ and either certifies that $\operatorname{tw}(\bar{G}) > k$ or constructs a tree decomposition of width $O(k)$ [3]. Therefore, under assumption (i), a tree decomposition of $\bar{G}$ of width $O(\log |V|)$ can be constructed in polynomial time.

Remark 3. Assumptions (i)-(ii) of Theorem 1 do not imply any nontrivial upper bound on the number of variables corresponding to V^+. In fact, it is possible to have both $|V^+| = \Theta(|V|)$ and $|V^-| = \Theta(|V|)$. To see this, for each integer $m \geq 1$, define a graph $G = (V, E)$ as follows. Let $V^+ := \{p_1, \dots, p_m\}$, $V^- := \{z_1, \dots, z_{m+1}\}$, and let $V := V^+ \cup V^-$. The nodes in V^- form a path, and each node $p_i \in V^+$ is adjacent to two consecutive nodes $z_i, z_{i+1} \in V^-$. More precisely, $E := \big\{\{z_i, z_{i+1}\} : i \in [m]\big\} \cup \big\{\{p_i, z_i\}, \{p_i, z_{i+1}\} : i \in [m]\big\}$. Choose the diagonal coefficients of Q so that $q_{p_i p_i} > 0$ for every $i \in [m]$ and $q_{z_j z_j} \leq 0$ for every $j \in [m + 1]$, and choose the off-diagonal coefficients so that the interaction graph of Q is G. Then the set defined in (5) is precisely V^+. Since there are no edges between distinct nodes in V^+, the connected components of G_{V^+} are the singletons $C_i := \{p_i\}$ for all $i \in [m]$. For each $i \in [m]$, we have $N(C_i) = \{z_i, z_{i+1}\}$. Moreover, since

C_i is a singleton and $q_{p_ip_i} > 0$, the matrix Q_{C_i} is positive definite, and hence $\kappa_{C_i} = 0$. Therefore, assumption (ii) is satisfied. The subgraph G_{V^-} induced by V^- is the path $z_1 - z_2 - \cdots - z_{m+1}$. Furthermore, each set $N(C_i) = \{z_i, z_{i+1}\}$ already forms a clique in G_{V^-}. Consequently, making each $N(C_i)$ a clique adds no new edges, and hence $\bar{G} = G_{V^-}$. It follows that $\text{tw}(\bar{G}) = 1$, so assumption (i) is also satisfied. Finally, $|V| = 2m + 1$, $|V^+| = m$, $|V^-| = m + 1$, and therefore $|V^+| = \Theta(|V|)$ and $|V^-| = \Theta(|V|)$. See Figure 1.

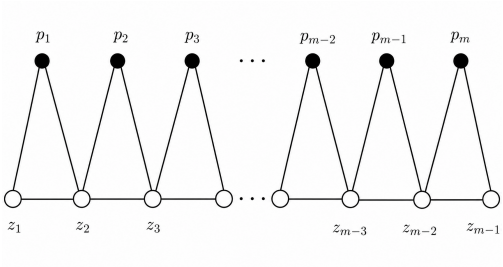

Figure 1: The example of Remark 3 illustrating the application of Theorem 1. The black nodes represent V^+, while the white nodes represent V^-

Remark 4. In [13, 17], the authors propose new convex relaxations for Problem (4). To this end, they associate a graph $G = (V, E, L)$ with this problem, where V and E are node set and edge set, respectively, of G as defined for our interaction graph, and L denotes the loop set, where $\{i, i\} \in L$, if $q_{ii} \neq 0$. They further define $L = L^- \cup L^+$, where L^- (resp. L^+) contain all loops with $q_{ii} < 0$ (resp. $q_{ii} > 0$). Subsequently, they study the facial structure of the set:

$$\text{QP}(G) := \text{conv}\Big\{z \in \mathbb{R}^{V \cup E \cup L} : z_{ii} \geq z_i^2,\ \forall \{i,i\} \in L^+,\ z_{ii} \leq z_i^2,\ \forall \{i,i\} \in L^-,\ z_{ij} = z_i z_j,$$
$$\forall \{i,j\} \in E,\ z_i \in [0,1], \forall i \in V\Big\},$$

where $\text{conv}(\cdot)$ denotes the convex hull. They obtain sufficient conditions under which $\text{QP}(G)$ admits a polynomial-size second-order cone (SOC) or semidefinite programming (SDP) representable formulation that can be constructed in polynomial time. We next summarize these sufficient conditions. Suppose that $\text{tw}(G) \in O(\log |V|)$. We have the following cases:

1. if $|C| = 1$ and $|N(C)| \in O(\log |V|)$ for all $C \in \mathcal{C}$, then $\text{QP}(G)$ admits a polynomial-size SOC-representable formulation that can be constructed in polynomial time (see theorem 3 in [13] and corollary 2 in [17]).
2. if $|C| = 2$ and $|N(C)| \in O(\log |V|)$ for all $C \in \mathcal{C}$, then $\text{QP}(G)$ admits a polynomial-size SDP-representable formulation that can be constructed in polynomial time (see theorem 6 in [17]).

Conditions (1) and (2) are special cases of Corollary 1 because assumption (ii) can be equivalently stated as $|N(C)| \in O(\log |V|)$ and $\kappa_C \log(1 + \frac{|C|}{\kappa_C}) \in O(\log |V|)$ for all $C \in \mathcal{C}$. Moreover, since by construction $\kappa_C \leq |C|$, we have $\kappa_C \log(1 + \frac{|C|}{\kappa_C}) \leq \kappa_C \frac{|C|}{\kappa_C} = |C|$. Second, the tightness of a polynomial-size SOCP or SDP relaxation of Problem (4) does not imply its polynomial-time solvability because when the optimal solution of Problem (4) is not unique, the solution returned by the SOCP or SDP solver may not be feasible for the nonconvex problem (4). It is also important to note that Corollary 1 does not imply the results of [13, 17] either, because polynomial-time solvability of Problem (4) does not imply that $\text{QP}(G)$ admits a polynomial-size extended formulation.

2.2 Proof of Theorem 1

To prove Theorem 1, we first present a number of intermediate results that will be used in the proof. The first proposition is a celebrated result in discrete optimization. Recall that given a hypergraph $G = (V, E)$, the primal treewidth of G, denoted by $\mathrm{ptw}(G)$ is the treewidth of its intersection graph. In the following, by $\mathrm{poly}(|V|, |E|)$, we denote a polynomial function in $|V|, |E|$.

Proposition 1 ([8])**.** *Let $G = (V, E)$ be a hypergraph with the primal treewidth $\mathrm{ptw}(G) = \kappa$. Then Problem (2) can be solved by dynamic programming in $O(2^\kappa |V|)$ operations. In particular, if $\kappa \in O(\log \mathrm{poly}(|V|, |E|))$, then Problem (2) is solvable in strongly polynomial time.*

The next result is concerned with Problem (4) and essentially states that the variables x_i with $q_{ii} \leq 0$ can be treated as binary variables. We refer to such variables as *hidden binary variables*.

Lemma 1. *Let $G = (V, E)$ be the interaction graph of Problem (4). Then there exists an optimal solution x^* of Problem (4) such that $x_i^* \in \{0, 1\}$ for all $i \in V \setminus V^+$, where V^+ is defined by (5).*

Proof. Let x^* be a minimizer of Problem (4). Consider some $i \in V \setminus V^+$ and fix all other coordinates at their current values. The resulting univariate function is concave and therefore has an endpoint with objective value no greater than the current value. Since the current point is optimal, equality must hold. Replace x_i^* by that endpoint and proceed to the next coordinate. This preserves optimality throughout and eventually makes every coordinate in $V \setminus V^+$ binary. □

We make use of the following result [23] to obtain a polynomial-time face enumeration algorithm to solve certain box-constrained quadratic programs.

Proposition 2. *Let x^* be a minimizer of Problem (4) satisfying the following properties:*

(i) x^* has the maximum number of elements that are equal to 0 or 1.
(ii) If there are two or more minima satisfying condition (i), then x^* has the maximum number of elements that are equal to zero.

Denote by $\bar{x}^\top \bar{Q} \bar{x} + \bar{c}^\top \bar{x}$ the reduced quadratic function obtained by substituting for binary elements of x^* into the objective function of Problem (4). Then the matrix $\bar{Q}$ is positive definite.

Thanks to Proposition 2 we next obtain a simple algorithm for solving Problem (4), which in particular runs in polynomial time when the number of variables is fixed.

Lemma 2. *Consider Problem (4) and let $\kappa \in \{0, \cdots, n-1\}$ be the smallest integer such that every $(n-\kappa) \times (n-\kappa)$ principal submatrix of Q is positive semidefinite. If no such integer exists, set $\kappa := n$. Then Problem (4) can be solved in time*

$$O\Big((\kappa + 1)\Big(\frac{2en}{\kappa}\Big)^\kappa \mathrm{poly}(\mathcal{L})\Big) \tag{7}$$

if $\kappa \geq 1$, and in time $\mathrm{poly}(\mathcal{L})$ if $\kappa = 0$.

Proof. For any $B \subseteq [n]$, denote by $Q_{[n]\setminus B}$ the principal submatrix of Q indexed by $[n] \setminus B$. For each subset $B \subseteq [n]$ with $|B| \leq \kappa$ and each $z \in \{0, 1\}^B$, consider the problem obtained from Problem (4) by fixing $x_B = z$. The resulting problem is a box-constrained quadratic optimization problem in the variables $x_{[n]\setminus B}$, whose quadratic matrix is $Q_{[n]\setminus B}$. If $Q_{[n]\setminus B}$ is positive semidefinite, then the resulting problem is a convex quadratic optimization problem and can be solved in time polynomial in $\mathcal{L}$ [21]. Consider the following algorithm. For every $B \subseteq [n]$ with $|B| \leq \kappa$ and every $z \in \{0, 1\}^B$, check whether $Q_{[n]\setminus B}$ is positive semidefinite. If so, solve the corresponding convex

quadratic optimization problem obtained by fixing $x_B = z$. Among all solutions obtained in this way, return one having minimum objective value.

We show that at least one of the convex quadratic optimization problems considered by the algorithm has the same optimal value as Problem (4). Let x^* be a global minimizer of Problem (4) satisfying the properties of Proposition 2, and define $B^* := \{i \in [n] : x_i^* \in \{0,1\}\}$ and $F^* := [n] \backslash B^*$. By Proposition 2, the quadratic matrix of the reduced problem obtained by substituting the binary components of x^* into the objective function is positive definite. Since fixing variables changes only the linear and constant terms involving the remaining variables, this quadratic matrix is precisely Q_{F^*}. Hence, $Q_{F^*} \succ 0$. Suppose first that $|B^*| \leq \kappa$. Consider the choice $B = B^*$ and $z = x^*_{B^*}$. Since $Q_{F^*} \succ 0$, the corresponding restricted problem is convex and is therefore solved by the algorithm. Moreover, x^* is feasible for this restricted problem. Since the feasible region of the restricted problem is contained in the feasible region of Problem (4), and x^* is a global minimizer of Problem (4), it follows that x^* is also a global minimizer of the restricted problem. Thus, this restricted problem has the same optimal value as Problem (4).

Now suppose that $|B^*| > \kappa$. Choose any subset $B \subseteq B^*$ with $|B| = \kappa$, and set $z = x^*_B$. By the definition of κ, every $(n-\kappa) \times (n-\kappa)$ principal submatrix of Q is positive semidefinite. Since $|[n] \setminus B| = n - \kappa$, we have $Q_{[n]\backslash B} \succeq 0$. Hence, the problem obtained by fixing $x_B = z$ is convex and is considered by the algorithm. Again, x^* is feasible for this restricted problem, and since x^* is globally optimal for Problem (4), it is also optimal for the restricted problem. Therefore, this restricted problem has the same optimal value as Problem (4). Thus, in either case, at least one of the convex quadratic optimization problems solved by the algorithm has the same optimal value as Problem (4). On the other hand, the feasible region of every restricted problem considered by the algorithm is contained in $[0,1]^n$, and hence its optimal value cannot be smaller than the optimal value of Problem (4). It follows that the best solution returned by the algorithm is a minimizer of Problem (4). It remains to bound the running time. For each $r \in \{0, \ldots, \kappa\}$, there are $\binom{n}{r}$ subsets $B \subseteq [n]$ with $|B| = r$, and for each such subset there are 2^r vectors $z \in \{0,1\}^B$. Hence, the total number of restricted problems considered is $\sum_{r=0}^{\kappa} 2^r \binom{n}{r}$. For $\kappa = 0$, this quantity is equal to one. Suppose that $\kappa \geq 1$. Using $\binom{n}{r} \leq \left(\frac{en}{r}\right)^r$ for $1 \leq r \leq n$, we obtain $2^r \binom{n}{r} \leq \left(\frac{2en}{r}\right)^r$. The function $g(t) = t \log(2en/t)$ is increasing on $[1, n]$, since its derivative is $\log(2n/t) > 0$ on this interval. Hence, for every $1 \leq r \leq \kappa$, we have $\left(\frac{2en}{r}\right)^r \leq \left(\frac{2en}{\kappa}\right)^\kappa$, implying that,

$$\sum_{r=0}^{\kappa} 2^r \binom{n}{r} \leq (\kappa + 1) \left(\frac{2en}{\kappa}\right)^\kappa.$$

For every pair (B, z), positive semidefiniteness of $Q_{[n]\backslash B}$ can be checked in time polynomial in $\mathcal{L}$. Moreover, whenever $Q_{[n]\backslash B}$ is positive semidefinite, the corresponding box-constrained convex quadratic optimization problem can be solved in time polynomial in $\mathcal{L}$. Therefore, the total running time is given by (7) for $\kappa \geq 1$, while for $\kappa = 0$ it is $\text{poly}(\mathcal{L})$, and this completes the proof. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. Define $V^- := V \setminus V^+$, where V^+ is defined by (5). Let x_{V^-} (resp. x_{V^+}) contain the components x_i of x with $i \in V^-$ (resp. $i \in V^+$). Denote by $f(x)$ the objective function of Problem (4). Then by Lemma 1, the optimal value of Problem (4) is equal to the optimal value of the following problem:

$$\begin{aligned} \min \quad & f(x_{V^-}, x_{V^+}) \\ \text{s.t.} \quad & x_{V^-} \in \{0,1\}^{V^-},\ x_{V^+} \in [0,1]^{V^+}. \end{aligned}$$

Let G_{V^+} be the subgraph of G induced by V^+, and let $\mathcal{C}$ denote the set of connected components of G_{V^+}. Define

$$f_{V^-}(x_{V^-}) := \sum_{i \in V^-} q_{ii} x_i^2 + 2 \sum_{\substack{\{i,j\} \in E: \\ i,j \in V^-}} q_{ij} x_i x_j + \sum_{i \in V^-} c_i x_i,$$

and, for each $C \in \mathcal{C}$, define

$$f_C(x_C, x_{N(C)}) := \sum_{i \in C} q_{ii} x_i^2 + 2 \sum_{\substack{\{i,j\} \in E: \\ i,j \in C}} q_{ij} x_i x_j + 2 \sum_{\substack{\{i,j\} \in E \\ i \in C,\ j \in N(C)}} q_{ij} x_i x_j + \sum_{i \in C} c_i x_i,$$

where $N(C)$ is defined by (6). Since the sets $C \in \mathcal{C}$ are the connected components of G_{V^+} and $N(C) \subseteq V^-$ for all $C \in \mathcal{C}$, we deduce that:

$$f(x_{V^-}, x_{V^+}) = f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} f_C(x_C, x_{N(C)}).$$

Since $C \cap C' = \emptyset$ for any $C \neq C' \in \mathcal{C}$, for any fixed x_{V^-} we have:

$$\min_{x_{V^+} \in [0,1]^{V^+}} f(x_{V^-}, x_{V^+}) = f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} \min_{x_C \in [0,1]^C} f_C(x_C, x_{N(C)}). \tag{8}$$

Now, for each $C \in \mathcal{C}$, and for each $z \in \{0,1\}^{N(C)}$, define

$$\psi_C(z) := \min_{x_C \in [0,1]^C} f_C(x_C, z).$$

From (8) we deduce that:

$$\min_{\substack{x_{V^-} \in \{0,1\}^{V^-} \\ x_{V^+} \in [0,1]^{V^+}}} f(x_{V^-}, x_{V^+}) = \min_{x_{V^-} \in \{0,1\}^{V^-}} \left(f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} \psi_C(x_{N(C)}) \right).$$

Thus, once we compute $\psi_C(z)$ for all $z \in \{0,1\}^{N(C)}$ and for all $C \in \mathcal{C}$, the original continuous problem is reduced to a binary optimization problem. Let us now consider the complexity of computing the function values $\psi_C(z)$, $z \in \{0,1\}^{N(C)}$ for all $C \in \mathcal{C}$. For each fixed $z \in \{0,1\}^{N(C)}$, computing $\psi_C(z)$ amounts to minimizing a quadratic function over $[0,1]^C$. From assumption (ii) in the statement of the theorem it follows that $\kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) \in O(\log |V|)$ for all $C \in \mathcal{C}$, where the expression is defined to be zero when $\kappa_C = 0$. If $\kappa_C = 0$, then by Lemma 2, $\psi_C(z)$ can be computed in time $\text{poly}(\mathcal{L})$. Now suppose that $\kappa_C \geq 1$. By Lemma 2, for a fixed z, $\psi_C(z)$ can be computed in time

$$O\Big((\kappa_C + 1) \left(\frac{2e|C|}{\kappa_C}\right)^{\kappa_C} \text{poly}(\mathcal{L})\Big).$$

Since $\kappa_C \leq |C|$, we have $\kappa_C + 1 \leq 2^{\kappa_C} \leq \left(1 + \frac{|C|}{\kappa_C}\right)^{\kappa_C}$. Moreover, setting $t := \frac{|C|}{\kappa_C} \geq 1$, we have $2et \leq 8t \leq (1+t)^3$, and hence $\left(\frac{2e|C|}{\kappa_C}\right)^{\kappa_C} \leq \left(1 + \frac{|C|}{\kappa_C}\right)^{3\kappa_C}$. It follows that

$$(\kappa_C + 1) \left(\frac{2e|C|}{\kappa_C}\right)^{\kappa_C} \leq \left(1 + \frac{|C|}{\kappa_C}\right)^{4\kappa_C}.$$

From assumption (ii) it follows that $\left(1+\frac{|C|}{\kappa_C}\right)^{4\kappa_C} \in |V|^{O(1)}$. Hence, for each fixed $z \in \{0,1\}^{N(C)}$, the value $\psi_C(z)$ can be computed in $O(|V|^{O(1)}\text{poly}(\mathcal{L}))$ time. Moreover, since $N(C)$ is a clique of $\bar{G}$, from assumption (i) it follows that $|N(C)| \in O(\log|V|)$. Therefore, $\psi_C(z)$ for all $z \in \{0,1\}^{N(C)}$ can be computed in $O(|V|^{O(1)}\text{poly}(\mathcal{L}))$ time. Finally, we have $|\mathcal{C}| \leq |V|$. Therefore, all such functional computations can be done in time polynomial in the input length $\mathcal{L}$. Moreover, the algorithm in Lemma 2 computes $\psi_C(z)$ by solving a polynomial number of convex quadratic optimization problems with rational data of encoding length polynomial in $\mathcal{L}$. The optimal value of each such problem can be computed exactly and has encoding length polynomial in $\mathcal{L}$. Consequently, $\psi_C(z)$ can be computed exactly and also has encoding length polynomial in $\mathcal{L}$. Hence, all values defining the functions ψ_C can be represented using a polynomial number of bits.

Finally, let us consider the cost of solving the binary optimization problem:

$$\begin{aligned} \min \quad & f_{V^-}(x_{V^-}) + \sum_{C\in\mathcal{C}} \psi_C(x_{N(C)}) \\ \text{s.t.} \quad & x_{V^-} \in \{0,1\}^{V^-}. \end{aligned} \tag{9}$$

Since each term $\psi_C(x_{N(C)})$ depends only on the binary variables x_i, $i \in N(C)$, it admits a unique multilinear representation in these variables. Moreover, since assumption (i) implies $|N(C)| \in O(\log|V|)$, this representation can be constructed from the values of ψ_C in polynomial time. It contains at most $2^{|N(C)|} = |V|^{O(1)}$ monomials, and each of its coefficients is a sum of at most $2^{|N(C)|}$ values $\psi_C(z)$. Hence, each coefficient has encoding length polynomial in $\mathcal{L}$. Since $|\mathcal{C}| \leq |V|$, Problem (9) admits a binary polynomial representation of total encoding length polynomial in $\mathcal{L}$. Therefore, Problem (9) can be represented as a binary polynomial optimization problem of polynomial encoding length. Denote by G_{V^-} the subgraph of G induced by V^-. Denote by $\bar{G}$ the graph obtained from G_{V^-} by making each $N(C)$ a clique for all $C \in \mathcal{C}$. It then follows that the intersection graph of the hypergraph of the objective function of Problem (9), denoted by H, is a subgraph of $\bar{G}$. From assumption (i) we deduce that $\text{tw}(H) \leq \text{tw}(\bar{G}) \in O(\log|V|)$. Therefore, by Proposition 1, Problem (9) can be solved in time polynomial in $|V|2^{O(\log|V|)}\text{poly}(\mathcal{L}) = |V|^{O(1)} \cdot \text{poly}(\mathcal{L})$. Let $x^*_{V^-}$ be an optimal solution of Problem (9). For each $C \in \mathcal{C}$, let x^*_C be an optimal solution of the problem defining $\psi_C(x^*_{N(C)})$, that is,

$$x^*_C \in \arg\min_{x_C\in[0,1]^C} f_C(x_C, x^*_{N(C)}).$$

Since the components in $\mathcal{C}$ are pairwise disjoint, x^*_C, $C \in \mathcal{C}$, together with $x^*_{V^-}$ define a vector $x^* \in [0,1]^V$. By (8) and the definition of ψ_C, this vector attains the optimal value of Problem (4). Moreover, by Lemma 2, each x^*_C can be computed in polynomial time. Hence, an optimal solution of Problem (4) can also be recovered in polynomial time and this completes the proof. □

2.3 Proof of Corollary 1

To prove Corollary 1, it suffices to show that if the interaction graph G has treewidth $O(\log|V|)$ and the neighborhoods $N(C)$ have logarithmic size, then the graph obtained from G by removing the components of G_{V^+} and making their neighborhoods cliques also has treewidth $O(\log|V|)$. The next two lemmas establish this fact. In the following, given a graph $G = (V, E)$ and a node $v \in V$, we say that the graph $G - v := (V', E')$ is obtained from G by *removing* v if $V' = V \setminus \{v\}$ and $E' = \{\{i,j\} \in E : i \neq v, j \neq v\}$.

Lemma 3 ([17]). *Let $G = (V, E)$ be a graph. Let $C \subseteq V$ be such that the subgraph of G induced by C is a connected graph. Denote by G' a graph obtained from G by removing all nodes in C and then making $N(C)$ a clique, i.e., adding all missing edges between pairs of nodes in $N(C)$, where $N(C)$ is defined by (6). Then the treewidth of G' is upper bounded by*

$$\mathrm{tw}(G') \leq \max(\mathrm{tw}(G),\ |N(C)| - 1). \tag{10}$$

The next result is obtained by a repeated application of the above lemma.

Lemma 4 ([17]). *Consider a graph $G = (V, E)$ and let $S \subseteq V$ be nonempty. Denote by G_S the subgraph of G induced by S. Denote by $C_1, \cdots, C_p$ the connected components of G_S, for some $p \geq 1$. Define $G_0 := G$. For each $i \in [p]$, let G_i be the graph obtained from G_{i-1} by removing the nodes in C_i and adding edges between the pairs of nodes in $G_{i-1} - C_i$ that are both adjacent to some node in C_i in G_{i-1}. Define*

$$N(C_i) := \{\, u \in V \setminus C_i : \exists v \in C_i \text{ with } \{u, v\} \in E\}, \quad \forall i \in [p],$$

and $d_{\max} := \max_{i\in[p]} |N(C_i)|$. We then have

$$\mathrm{tw}(G_p) \leq \max\big(\mathrm{tw}(G),\ d_{\max} - 1\big),$$

where G_p is the graph obtained from G after p component removal and clique addition operations.

We are now ready to prove Corollary 1.

Proof of Corollary 1. Let $V^- := V \setminus V^+$, and let $\bar{G}$ be the graph on V^- obtained from the subgraph of G induced by V^- by making $N(C)$ a clique for every $C \in \mathcal{C}$, as in Theorem 1. We show that assumptions (i)–(ii) of the Corollary 1 imply assumptions (i)–(ii) of Theorem 1.

Since both terms in assumption (ii) of Corollary 1 are nonnegative, we have, for every $C \in \mathcal{C}$, $|N(C)| \in O(\log |V|)$ and $\kappa_C \log\left(1 + \frac{|C|}{\kappa_C}\right) \in O(\log |V|)$, where the second expression is defined to be zero when $\kappa_C = 0$. Thus, assumption (ii) of Theorem 1 is satisfied. It remains to verify assumption (i) of Theorem 1, namely that $\mathrm{tw}(\bar{G}) \in O(\log |V|)$. Suppose first that $V^+ = \emptyset$. Then $\mathcal{C} = \emptyset$ and $\bar{G} = G$. Hence, by assumption (i) of Corollary 1 we have $\mathrm{tw}(\bar{G}) = \mathrm{tw}(G) \in O(\log |V|)$. Now suppose that $V^+ \neq \emptyset$. Let $C_1, \ldots, C_p$ be the connected components of the subgraph of G induced by V^+. Since the C_i's are the connected components of G_{V^+}, there are no edges of G between two distinct components C_i and C_j. Consequently, $N(C_i) \subseteq V^-$ for every $i \in [p]$. Applying Lemma 4 with $S = V^+$, the graph obtained by removing all components $C_1, \ldots, C_p$ and making each $N(C_i)$ a clique satisfies

$$\mathrm{tw}(\bar{G}) \leq \max\{\mathrm{tw}(G), \max_{C\in\mathcal{C}} |N(C)| - 1\}.$$

The graph produced by these component-removal and clique-addition operations is precisely $\bar{G}$: after all nodes of V^+ are removed, the remaining node set is V^-, and for each component $C \in \mathcal{C}$, all pairs of nodes in $N(C)$ have been made adjacent. By assumption (i) of the corollary, $\mathrm{tw}(G) \in O(\log |V|)$, while, as observed above, assumption (ii) implies $\max_{C\in\mathcal{C}} |N(C)| \in O(\log |V|)$. Therefore, $\mathrm{tw}(\bar{G}) \leq \max\{O(\log |V|), O(\log |V|)\} = O(\log |V|)$. Hence, assumption (i) of Theorem 1 is also satisfied. Therefore, Problem (4) can be solved in time polynomial in the input length $\mathcal{L}$. □

3 Higher-degree polynomial optimization

In this section, we consider a box-constrained polynomial optimization problem of degree $d \geq 3$. We obtain a sufficient condition under which this problem can be solved exactly in time polynomial in the input length in the unit-cost algebraic RAM model, which we formally define next.

The computational model. Throughout this section, running times are measured in the *unit-cost algebraic RAM model* [22]. The machine has ordinary index registers and numerical registers. Numerical registers contain exact rational numbers. Each arithmetic operation $+$, $-$, $\times$, $\div$, each comparison between numerical registers, and each memory operation is assigned unit cost, independently of the bit lengths of the rational numbers involved. The input length $\mathcal{L}$ retains its usual bit-model meaning and denotes the number of bits required to encode the input polynomial. Thus, a running time polynomial in $\mathcal{L}$ in this section means that the number of unit-cost algebraic RAM operations is polynomial in $\mathcal{L}$; the bit lengths of intermediate rational numbers are not included in the running time. Moreover, real algebraic numbers are represented symbolically. A real algebraic number α is represented by a pair (p, I), where p is a univariate square-free polynomial with integer coefficients, and I is a rational interval containing one real root of p, namely α. Whenever real algebraic representations are manipulated below, we use explicit symbolic algorithms over $\mathbb{Q}$, and measure their running time by the number of underlying unit-cost rational arithmetic operations and comparisons. A real algebraic vector is represented by a real univariate representation in the standard sense of [1]. Rational linear combinations of real algebraic numbers are stored formally, by keeping the algebraic numbers separately and recording their rational coefficients. An exact optimizer of Problem (1) is represented by its binary coordinates together with one real univariate representation for the continuous coordinates in each component. The optimal value is represented as a formal rational linear combination of the algebraic local optimal values.

As in Section 2, we first identify a subset of variables that can be treated as binary variables because there is an optimal solution of Problem (1) at which these variables take binary values. As before, we refer to such variables as hidden binary variables. For a vector $x \in \mathbb{R}^n$ and an index $i \in [n]$, we denote by $x_{-i} := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \mathbb{R}^{n-1}$ the vector obtained from x by removing its i-th coordinate. Accordingly, for $t \in \mathbb{R}$ we write $(x_{-i}, t) := (x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_n) \in \mathbb{R}^n$. The *standard monomial basis* of the space of polynomials in n variables of degree at most d is the set $\{x^\alpha : \alpha \in \mathbb{Z}^n_{\geq 0}, |\alpha| \leq d\}$; a polynomial is said to be written in this basis if it is expressed as a linear combination of these monomials. The following lemma provides easily verifiable sufficient conditions for identifying hidden binary variables:

Lemma 5. *Let $f(x)$, $x \in \mathbb{R}^n$ be a polynomial and fix some $i \in [n]$. Then we have the following:*

1. *Write $f(x)$ as a polynomial in x_i:*

$$f(x) = \sum_{m=0}^{d} a_m(x_{-i})\, x_i^m,$$

where each $a_m(x_{-i})$ is a polynomial in the remaining variables x_{-i}. Assume that for every $m \geq 2$, the polynomial $a_m(x_{-i})$ has only nonpositive coefficients when written in the standard monomial basis. Then there exists a minimizer x^* of $f(x)$ over $[0,1]^n$ such that $x_i^* \in \{0,1\}$.

2. *Define*

$$\Delta_i(x) := f(x) - (1 - x_i) f(x_{-i}, 0) - x_i f(x_{-i}, 1),$$

and write $\Delta_i(x) = x_i(1 - x_i)H_i(x)$. Assume that $H_i(x)$ has only nonnegative coefficients when written in the standard monomial basis. Then there exists a minimizer x^* of $f(x)$ over $[0,1]^n$ such that $x_i^* \in \{0,1\}$.

Proof. First consider part (1). Fix $x_{-i} \in [0,1]^{n-1}$ and consider

$$\phi(t) = f(x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_n) = \sum_{m=0}^{d} a_m(x_{-i})\, t^m.$$

Since each monomial in the variables x_{-i} is nonnegative on $[0,1]^{n-1}$, and since each coefficient of $a_m(x_{-i})$ is nonpositive for every $m \geq 2$, it follows that $\phi(t)$ is concave on $[0,1]$. Therefore, there exists a minimizer of $f(x)$, $x \in [0,1]^n$ at $x_i^* \in \{0,1\}$.

Next, consider part (2). Since every monomial is nonnegative on $[0,1]^n$ and H_i has only nonnegative coefficients in the standard monomial basis, we have $H_i(x) \geq 0$ for all $x \in [0,1]^n$. Therefore $\Delta_i(x) \geq 0$ for all $x \in [0,1]^n$. Thus, the slice of $f(x)$ in x_i lies above its chord for every fixing of the other variables, implying that there exists a minimizer x^* of $f(x)$ with $x_i^* \in \{0,1\}$. □

Remark 5. In Lemma 5, while condition 1 is simpler to verify, condition 2 is strictly more general. To see this, consider $f(x_1, x_2) = x_1(1 - x_1)(x_1 + x_2)$. Expanding in powers of x_1, we obtain $f(x_1, x_2) = -x_1^3 + (1 - x_2)x_1^2 + x_1x_2$. The coefficient of x_1^2 is the polynomial $1 - x_2$, which has a positive coefficient in the standard monomial basis. Therefore, condition 1 of Lemma 5 does not apply. On the other hand, $f(0, x_2) = f(1, x_2) = 0$, implying that $\Delta_1(x_1, x_2) = f(x_1, x_2) = x_1(1 - x_1)H_1(x_1, x_2)$, with $H_1(x_1, x_2) = x_1 + x_2$. Since H_1 has only nonnegative coefficients in the standard monomial basis, condition 2 of Lemma 5 applies. Hence there exists a minimizer of f with $x_1 \in \{0,1\}$. For $d = 2$, conditions (1) and (2) of Lemma 5 are equivalent and both reduce to the condition of Lemma 1; i.e., $q_{ii} \leq 0$. To see this consider $f(x) = x^\top Qx + c^\top x$. Viewing f as a polynomial in x_i, we obtain

$$f(x) = q_{ii}x_i^2 + \Big(2\sum_{j \neq i} q_{ij}x_j + c_i\Big)x_i + \text{terms independent of } x_i.$$

Thus $a_2(x_{-i}) = q_{ii}$, and condition (1) is equivalent to $q_{ii} \leq 0$. Moreover, a direct computation gives $\Delta_i(x) = f(x) - (1 - x_i)f(x_{-i}, 0) - x_i f(x_{-i}, 1) = -q_{ii}\, x_i(1 - x_i)$. Hence in condition (2) we have $H_i(x) = -q_{ii}$, and requiring H_i to have nonnegative coefficients is again equivalent to $q_{ii} \leq 0$.

We should remark that a more general sufficient condition than those Lemma 5 is $H_i(x) \geq 0$ for all $x \in [0,1]^n$. However, this condition cannot be verified in polynomial time, in general.

Consider the polynomial $f(x) = \sum_{|\alpha| \leq d} c_\alpha x^\alpha$, $x \in \mathbb{R}^n$. We associate with f the *interaction hypergraph* $G = (V, E)$ with $V = [n]$ and for each monomial $c_\alpha x^\alpha$ with $c_\alpha \neq 0$ and $|\operatorname{supp}(\alpha)| \geq 2$, we include the edge $\operatorname{supp}(\alpha) := \{i \in V : \alpha_i \neq 0\}$ in E. Let $G = (V, E)$ be a hypergraph and let $C \subseteq V$. We define the *subhypergraph* of G induced by C as $G_C = (C, E_C)$ where $E_C := \{e \cap C : e \in E,\ |e \cap C| \geq 2\}$. A hypergraph $G = (V, E)$ is *connected* if for every $u, v \in V$ there exists a sequence of edges $e_1, \ldots, e_k \in E$ such that $u \in e_1$, $v \in e_k$, $e_j \cap e_{j+1} \neq \emptyset$ for all $j \in [k-1]$. The *connected components* of a hypergraph are the maximal subsets of nodes that induce connected subhypergraphs. Recall that the incidence graph $I(G)$ of a hypergraph G is the bipartite graph with node set $V \cup E$, in which $v \in V$ is adjacent to $e \in E$ if and only if $v \in e$. We then define the *incidence treewidth* of G, denoted by $\operatorname{itw}(G)$, as the treewidth of $I(G)$.

We are now ready to state the main result of this section.

Theorem 2. *Consider Problem* (1) *with the interaction hypergraph $G = (V, E)$. Let V^- be the set of hidden binary variables as defined by Lemma 5, and let $V^+ = V \setminus V^-$. Denote by G_{V^+} the subhypergraph of G induced by V^+ and denote by $\mathcal{C}$ the set of connected components of G_{V^+}. For each component $C \in \mathcal{C}$, define its neighborhood:*

$$N(C) := \big\{u \in V \setminus C : \exists\, e \in E \text{ such that } e \cap C \neq \emptyset,\ u \in e\big\}.$$

Suppose that the following conditions are satisfied:

(i) $\operatorname{itw}(G) \in O(\log |V|)$;

(ii) $|C| \in O(1)$ *for all* $C \in \mathcal{C}$*;*
(iii) $|N(C)| \in O(\log |V|)$ *for all* $C \in \mathcal{C}$*.*

Then Problem (1) *can be solved exactly in time polynomial in the input length* $\mathcal{L}$ *in the unit-cost algebraic RAM model. The algorithm returns an exact symbolic representation of an optimal solution and of the optimal value.*

We note that the structural quantities appearing in the assumptions of Theorem 2 can be computed in time polynomial in $\mathcal{L}$. Indeed, $G = (V, E)$ can be constructed in polynomial time from the monomial representation of the polynomial. The set V^- can be identified by checking the sufficient conditions of Lemma 5 for each variable. Condition (1) can be checked directly from the coefficients of the input polynomial. For condition (2), writing $f(x) = \sum_{m=0}^{d} a_m(x_{-i})x_i^m$, we have $H_i(x) = -\sum_{m=2}^{d} a_m(x_{-i})\left(1 + x_i + \cdots + x_i^{m-2}\right)$. Thus, each input monomial generates at most $d-1$ monomials in the explicit representation of H_i. Since d is polynomially bounded in $\mathcal{L}$, H_i can be constructed explicitly and its coefficient signs can be checked in time polynomial in $\mathcal{L}$. The connected components of G_{V^+}, as well as $N(C)$ for each component C, can be computed by performing a graph traversal on the incidence graph of G in time linear in the size of the incidence representation, namely $O\big(|V| + |E| + \sum_{e \in E} |e|\big)$. Once these sets are obtained, verifying that $|C| \in O(1)$ and $|N(C)| \in O(\log |V|)$ is immediate. Finally, although computing treewidth exactly is $\mathcal{NP}$-hard, there exists an algorithm that, given a graph with N nodes and an integer k, runs in time $2^{O(k)}N$ and either certifies that the treewidth is greater than k or constructs a tree decomposition of width $O(k)$ [3].

To prove Theorem 2 we first present a number of intermediate results that will be used in the proof. The first result serves as the strongest sufficient condition for polynomial-time solvability of binary polynomial optimization.

Proposition 3 ([6])**.** *There is a strongly polynomial time algorithm to solve Problem* (2) *if the interaction hypergraph* G *is* β*-acyclic or if* $\mathrm{itw}(G) \in O(\log \mathrm{poly}(|V|, |E|))$*. Moreover, an optimal solution of Problem* (2) *can be found using* $2^{O(\mathrm{itw}(G))}\mathrm{poly}(|V|, |E|)$ *arithmetic operations and comparisons involving the objective function coefficients.*

Now let us turn our attention to a continuous polynomial optimization problem. The problem of minimizing a polynomial over a compact semialgebraic set can be reduced to deciding the feasibility of semialgebraic systems. In particular, deciding whether there exists $x \in [0,1]^k$ such that $f(x) \leq \lambda$ is an instance of the existential theory of the reals. It is a classical result that such problems can be solved in time polynomial in the encoding length of the input when the number of variables is fixed; see Renegar [19] and Basu–Pollack–Roy [1]. More precisely, from Algorithm 14.1, Algorithm 14.9, and Theorem 13.11 of [1], we obtain the following result:

Proposition 4 ([1])**.** *Given a polynomial* $f(x)$*,* $x \in \mathbb{R}^n$*, of degree at most* d *and with rational coefficients, the problem of minimizing* f *over* $[0,1]^n$ *can be solved exactly using* $(nd)^{O(n)}$ *arithmetic operations over* $\mathbb{Q}$*. The algorithm returns an exact real algebraic representation of the optimal value together with a real univariate representation of an optimal solution. For fixed* n*, the degrees and encoding lengths of these representations are polynomially bounded in* d *and* $\mathcal{L}$*.*

Recall that throughout this paper we assume that d is polynomially bounded in $\mathcal{L}$. Therefore, for fixed n, Proposition 4 implies that Problem (1) can be solved using a number of unit-cost algebraic RAM operations polynomial in $\mathcal{L}$. We should remark that even in fixed dimension, the optimal value of a polynomial optimization problem over $[0,1]^n$ may not be rational for $d \geq 3$. In general, optimal solutions and optimal values are algebraic numbers, i.e., roots of polynomials

with rational coefficients. An exact representation of a real algebraic optimal value consists of a square-free defining polynomial together with a rational isolating interval containing the root.

Finally, to prove Theorem 2 we need to show that the incidence treewidth of the interaction hypergraph remains bounded even if we add certain complete hypergraphs with small node sets to it. The next two lemmas establish this fact. To state these results, we first need to introduce some notation and terminology. Given a graph $G = (V, E)$, a *tree decomposition* of G is a pair $(\mathcal{X}, T)$, where $\mathcal{X} = \{X_1, \cdots, X_m\}$ is a family of subsets of V, called *bags*, and T is a tree with m nodes, where each node of T corresponds to a bag such that:

1. $V = \bigcup_{i \in [m]} X_i$.
2. For every edge $\{v_j, v_k\} \in E$, there is a bag X_i for some $i \in [m]$ such that $X_i \ni v_j, v_k$.
3. For each node $v \in V$, the set of all bags containing v induces a connected subtree of T.

The *width* $\omega(\mathcal{X})$ of a tree decomposition $(\mathcal{X}, T)$ is the size of its largest bag X_i minus one. The *treewidth* $\mathrm{tw}(G)$ of a graph G is the minimum width among all possible tree decompositions of G. Let $G = (V, E)$ be a hypergraph and let $C \subseteq V$. We define the hypergraph obtained from G by *removing* C as $G - C := (V \setminus C, E')$, where $E' := \{e \setminus C : e \in E,\ |e \setminus C| \geq 2\}$.

Lemma 6. *Let $G = (V, E)$ be a hypergraph, and let $C \subseteq V$ be such that the subhypergraph of G induced by C is connected. Define the neighborhood of C as*

$$N(C) := \{u \in V \setminus C : \exists e \in E \text{ with } u \in e \text{ and } e \cap C \neq \emptyset\}. \tag{11}$$

Let G' be the hypergraph obtained from G by removing C and subsequently adding the complete hypergraph on $N(C)$, i.e., adding as edges all subsets $f \subseteq N(C)$ with $|f| \geq 2$. Then we have

$$\mathrm{itw}(G') \leq \max\big(\mathrm{itw}(G),\ |N(C)|\big).$$

Proof. Let $I(G) = (\bar{V}, \bar{E})$ denote the incidence graph of the hypergraph G. Define

$$S := C \cup \{e \in E : e \cap C \neq \emptyset\}.$$

Note that $S \subseteq \bar{V}$. We first show that the subgraph of $I(G)$ induced by S is connected. Denote by $G_C = (C, E_C)$ the subhypergraph of G induced by C. Take any two nodes $u, v \in C$. Since by assumption G_C is connected, there exist edges $f_1, \ldots, f_k \in E_C$ for some $k \geq 1$ such that $u \in f_1$, $v \in f_k$, and $f_j \cap f_{j+1} \neq \emptyset$ for all $j = 1, \ldots, k-1$. For each $j \in [k]$, choose an edge $e_j \in E$ such that $f_j = e_j \cap C$. Moreover, for each $j \in [k-1]$, choose a node $w_j \in f_j \cap f_{j+1} \subseteq C$. Define the sequence of nodes of $I(G)$: $u, e_1, w_1, e_2, w_2, \ldots, w_{k-1}, e_k, v$. This is a path in $I(G)$ because $u \in e_1$, $v \in e_k$ and for each $j \in [k-1]$, we have $w_j \in e_j \cap e_{j+1}$. Moreover, every node on this path belongs to S, so u and v are connected in the subgraph $I(G)$ induced by S. Now let $e \in E$ be any edge with $e \cap C \neq \emptyset$ implying that $e \in S$. Pick $u \in e \cap C$. Then e is adjacent to u in $I(G)$. Since every node in C lies in the same connected component of the subgraph $I(G)$ induced by S, it follows that every such e also lies in that same connected component. Hence, the subgraph $I(G)$ induced by S is connected.

Denote by $N_{I(G)}(S)$ the neighborhood of S in the graph $I(G)$. We claim that

$$N_{I(G)}(S) = N(C), \tag{12}$$

where $N(C)$ is defined by (11). To see this, let $v \in V \setminus C$. Then v is adjacent in $I(G)$ to a node of S if and only if there exists an edge $e \in E$ such that $v \in e$ and $e \cap C \neq \emptyset$. This is precisely the definition of $v \in N(C)$. Therefore, $N_{I(G)}(S) \cap V = N(C)$. On the other hand, consider some $e \in E \setminus S$. By definition of S, this means that $e \cap C = \emptyset$. If e were adjacent to some node of S in

$I(G)$, then there would exist a node $v \in C$ such that $v \in e$, which contradicts $e \cap C = \emptyset$. Hence, no $e \in E \setminus S$ is adjacent to S. Therefore, equality (12) holds.

Let H be the graph obtained from $I(G)$ by removing S and making its neighborhood a clique. Since the subgraph of $I(G)$ induced by S is connected, by Lemma 3 and equality (12) we have:

$$\mathrm{tw}(H) \leq \max\big(\mathrm{itw}(G),\, |N(C)| - 1\big). \tag{13}$$

Let $(\mathcal{Y}, T')$ be a tree decomposition of H of width at most the right-hand side of (13). Since $N(C)$ is a clique in H, there exists a node $t^* \in V(T')$ whose bag Y_{t^*} contains $N(C)$; i.e.,

$$N(C) \subseteq Y_{t^*}. \tag{14}$$

We now construct a tree decomposition of $I(G')$ from $(\mathcal{Y}, T')$. From the definition of the hypergraph G' it follows that the incidence graph $I(G')$ is obtained from H by deleting the edges having both incident nodes in $N(C)$ and, for each subset $f \subseteq N(C)$ with $|f| \geq 2$, adding one new node w_f adjacent to all nodes in f. First, observe that deleting edges does not affect the validity of a tree decomposition, so $(\mathcal{Y}, T')$ is also a tree decomposition of the graph obtained from H after removing the edges with incident nodes in $N(C)$. For each subset $f \subseteq N(C)$ with $|f| \geq 2$, add a new leaf node t_f adjacent to t^*, and define its bag by

$$Y_{t_f} := f \cup \{w_f\}. \tag{15}$$

Keep all old bags unchanged. Let $(\mathcal{Y}', T'')$ be the resulting family of bags and tree. We next verify that $(\mathcal{Y}', T'')$ is a tree decomposition of $I(G')$.

1. Every old node of $I(G')$ already belongs to some bag of $(\mathcal{Y}, T')$. Each new node w_f belongs to the new bag Y_{t_f} defined by (15). Hence condition 1 of a tree decomposition holds.
2. Every old edge of $I(G')$ is an edge of the graph obtained from H by deleting the edges inside $N(C)$, hence it is contained in some old bag. Now let (u, w_f) be a new edge of $I(G')$. By construction, $u \in f$, hence both u, w_f belong to the bag Y_{t_f} defined by (15). Hence condition (2) of a tree decomposition holds.
3. Let u be a node of $I(G')$. Then the following cases arise:
   - If u is an old node not in $N(C)$, then no new bag contains u, implying that the set of bags containing u is unchanged and remains connected.
   - If u is an old node and $u \in N(C)$, then by (14) we have $u \in Y_{t^*}$. Moreover, u belongs to a new bag Y_{t_f} exactly when $u \in f$, and every such new bag is attached as a leaf to t^*. Therefore, the set of bags containing u is obtained from the old connected subtree containing u by attaching some leaves to the node t^*, and hence remains connected.
   - If u is a new node, i.e., $u = w_f$, then it belongs only to the single bag Y_{t_f} defined by (15), so the set of bags containing u is connected.

   Hence condition (3) of a tree decomposition holds.

Thus, $(\mathcal{Y}', T'')$ is a tree decomposition of $I(G')$. By (15), we can upper bound the size of the new bags as $|Y_{t_f}| = |f| + 1 \leq |N(C)| + 1$. Therefore, the width of the tree decomposition $(\mathcal{Y}', T'')$ is at most $\max\big(\mathrm{itw}(G),\ |N(C)|\big)$, and this completes the proof. □

Lemma 7. *Let $G = (V, E)$ be a hypergraph, and let $S \subseteq V$ be nonempty. Denote by G_S the subhypergraph of G induced by S and let $C_1, \ldots, C_p$ denote the connected components of G_S. Define $G_0 := G$. For each $r \in [p]$, let G_r be the hypergraph obtained from G_{r-1} by first removing C_r and then adding the complete hypergraph on $N(C_r)$. Set $d_{\max} := \max_{r \in [p]} |N(C_r)|$. Then*

$$\mathrm{itw}(G_p) \leq \max\big(\mathrm{itw}(G),\, d_{\max}\big).$$

Proof. We prove the statement by induction on p. In the base case, we have $p = 1$ and the proof follows directly from Lemma 6. Now assume that $p \geq 2$ and that the statement holds for the first k components, where $1 \leq k < p$. That is, assume that

$$\operatorname{itw}(G_k) \leq \max\Big(\operatorname{itw}(G), \max_{r\in[k]} |N(C_r)|\Big). \tag{16}$$

We must prove that

$$\operatorname{itw}(G_{k+1}) \leq \max\Big(\operatorname{itw}(G), \max_{r\in[k+1]} |N(C_r)|\Big).$$

To do so, we will apply Lemma 6 to the hypergraph G_k and the set C_{k+1}. To this end, we need to verify two facts:

1. The subhypergraph of G_k induced by C_{k+1} is connected. Let $G_k = (V_k, E_k)$. Define $E[C_{k+1}] := \{e \cap C_{k+1} : e \in E, |e \cap C_{k+1}| \geq 2\}$ and $E_k[C_{k+1}] := \{e \cap C_{k+1} : e \in E_k, |e \cap C_{k+1}| \geq 2\}$. To prove the statement, it suffices to show that $E_k[C_{k+1}] = E[C_{k+1}]$. First we show that $E_k[C_{k+1}] \subseteq E[C_{k+1}]$. Let $f \in E_k[C_{k+1}]$. Then there exists $e \in E_k$ such that $f = e \cap C_{k+1}$ and $|e \cap C_{k+1}| \geq 2$. If $e \notin E$, then e was added during the elimination of some C_r with $r \in [k]$, implying that $e \subseteq N(C_r)$. Pick $v \in e \cap C_{k+1}$. Then $v \in N(C_r)$, so there exists $e' \in E$ such that $v \in e'$ and $e' \cap C_r \neq \emptyset$. Hence $e' \cap S$ intersects both C_r and C_{k+1}, contradicting that they are distinct connected components of G_S. Therefore $e \in E$, and $f \in E[C_{k+1}]$.
   Next we show that $E[C_{k+1}] \subseteq E_k[C_{k+1}]$. Let $f \in E[C_{k+1}]$. Then there exists $e \in E$ such that $f = e \cap C_{k+1}$ and $|e \cap C_{k+1}| \geq 2$. If $e \notin E_k$, then e was removed during the elimination of some C_r with $r \in [k]$, so $e \cap C_r \neq \emptyset$. Then $e \cap S$ intersects both C_r and C_{k+1}, again contradicting the definition of the components of G_S. Hence $e \in E_k$, and $f \in E_k[C_{k+1}]$.
2. The neighborhood of C_{k+1} in G_k is $N(C_{k+1})$; i.e., $N_{G_k}(C_{k+1}) = N(C_{k+1})$. We first show that $N_{G_k}(C_{k+1}) \subseteq N(C_{k+1})$. Let $u \in N_{G_k}(C_{k+1})$. Then there exists $e \in E_k$ such that $u \in e$ and $e \cap C_{k+1} \neq \emptyset$. If $e \notin E$, then $e \subseteq N(C_r)$ for some $r \in [k]$. Pick $v \in e \cap C_{k+1}$. Then $v \in N(C_r)$, so there exists $e' \in E$ such that $v \in e'$ and $e' \cap C_r \neq \emptyset$, yielding a contradiction as above. Hence $e \in E$, and $u \in N(C_{k+1})$. Next we show that $N(C_{k+1}) \subseteq N_{G_k}(C_{k+1})$. Let $u \in N(C_{k+1})$. Then there exists $e \in E$ such that $u \in e$, $e \cap C_{k+1} \neq \emptyset$. If $e \notin E_k$, then $e \cap C_r \neq \emptyset$ for some $r \in [k]$, which again contradicts the definition of the connected components of G_S. Hence $e \in E_k$, and $u \in N_{G_k}(C_{k+1})$.

Therefore we can apply Lemma 6 to the hypergraph G_k and the set C_{k+1} to obtain

$$\operatorname{itw}(G_{k+1}) \leq \max\big(\operatorname{itw}(G_k), |N(C_{k+1})|\big).$$

Using the induction hypothesis (16), we obtain

$$\operatorname{itw}(G_{k+1}) \leq \max\Big(\operatorname{itw}(G), \max_{r\in[k]} |N(C_r)|, |N(C_{k+1})|\Big),$$

which by definition of $d_{\max}$ completes the proof. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. For any $U \subseteq V$, denote by x_U the vector containing all components x_i of x with $i \in U$. From Lemma 5 it follows that there exists a minimizer of Problem (1) at which all variables indexed by V^- are binary. Consequently,

$$\min_{x\in[0,1]^V} f(x) = \min_{\substack{x_{V^-}\in\{0,1\}^{V^-} \\ x_{V^+}\in[0,1]^{V^+}}} f(x_{V^-}, x_{V^+}). \tag{17}$$

We next decompose $f(x)$ according to the connected components of G_{V^+}. First observe that $N(C) \subseteq V^-$ for every $C \in \mathcal{C}$. Indeed, suppose that $u \in N(C) \cap V^+$. Then there exists an edge $e \in E$ containing u and a node of C. Hence, $e \cap V^+$ is an edge of G_{V^+} containing u and a node of C, contradicting the fact that C is a connected component of G_{V^+}. Now consider any monomial x^α whose support intersects V^+. All nodes in $\operatorname{supp}(\alpha) \cap V^+$ belong to a unique component $C \in \mathcal{C}$. Indeed, this is immediate if $|\operatorname{supp}(\alpha) \cap V^+| = 1$, while if $|\operatorname{supp}(\alpha) \cap V^+| \geq 2$, then $\operatorname{supp}(\alpha) \cap V^+$ is an edge of G_{V^+} and hence all its nodes belong to the same connected component. Moreover, every node in $\operatorname{supp}(\alpha) \setminus C$ belongs to $N(C)$. Thus, every monomial whose support intersects V^+ depends on the variables of exactly one component C and possibly on variables in $N(C)$. Let $f_{V^-}(x_{V^-})$ be the sum of all monomials of f whose support is contained in V^-, and, for each $C \in \mathcal{C}$, let $f_C(x_C, x_{N(C)})$ be the sum of all monomials whose support intersects C. By the preceding observation, these collections of monomials form a partition of the monomials of f, and therefore

$$f(x) = f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} f_C(x_C, x_{N(C)}). \tag{18}$$

Since $C \cap C' = \emptyset$ for all $C \neq C' \in \mathcal{C}$, from (18) it follows that

$$\min_{x_{V^+} \in [0,1]^{V^+}} f(x_{V^-}, x_{V^+}) = f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} \min_{x_C \in [0,1]^C} f_C(x_C, x_{N(C)}).$$

For each $C \in \mathcal{C}$ and each $z \in \{0,1\}^{N(C)}$, define

$$\psi_C(z) := \min_{x_C \in [0,1]^C} f_C(x_C, z).$$

It follows from (17) that

$$\min_{x \in [0,1]^V} f(x) = \min_{x_{V^-} \in \{0,1\}^{V^-}} \Big(f_{V^-}(x_{V^-}) + \sum_{C \in \mathcal{C}} \psi_C(x_{N(C)}) \Big). \tag{19}$$

We now examine the cost of computing the functions ψ_C. Fix $C \in \mathcal{C}$ and $z \in \{0,1\}^{N(C)}$. By assumption (ii), $|C| \in O(1)$. Hence, by Proposition 4 and the assumption that d is polynomially bounded in $\mathcal{L}$, the value $\psi_C(z)$ and a corresponding minimizer can be computed exactly using a number of unit-cost rational arithmetic operations polynomial in $\mathcal{L}$. Moreover, $\psi_C(z)$ can be represented by a square-free polynomial with integer coefficients together with a rational isolating interval, and a corresponding minimizer can be represented by a real univariate representation. The degrees and encoding lengths of these representations are polynomially bounded in $\mathcal{L}$. By assumption (iii), $2^{|N(C)|} = |V|^{O(1)}$. Since $|\mathcal{C}| \leq |V|$, all values $\psi_C(z)$, for $C \in \mathcal{C}$ and $z \in \{0,1\}^{N(C)}$, together with corresponding minimizers, can be computed using a number of unit-cost algebraic RAM operations polynomial in $\mathcal{L}$. Let $\alpha_1, \ldots, \alpha_M$ denote all the algebraic values obtained in this way. Then $M = \sum_{C \in \mathcal{C}} 2^{|N(C)|} = |V|^{O(1)}$, and every α_j has a defining polynomial and isolating interval of degree and encoding length polynomially bounded in $\mathcal{L}$. Each function ψ_C admits a unique multilinear representation $\psi_C(z) = \sum_{S \subseteq N(C)} a_{C,S} \prod_{i \in S} z_i$, where, by Mobius inversion, $a_{C,S} = \sum_{T \subseteq S} (-1)^{|S|-|T|} \psi_C(\mathbf{1}_T)$, and $\mathbf{1}_T$ denotes the binary vector whose entries indexed by T are equal to one and whose remaining entries are equal to zero. We store every coefficient $a_{C,S}$ as a formal integer linear combination of $\alpha_1, \ldots, \alpha_M$. Since $\sum_{S \subseteq N(C)} 2^{|S|} = 3^{|N(C)|} = |V|^{O(1)}$, all these representations can be constructed in polynomial time and have polynomial total encoding length. Therefore, the problem on the right-hand side of (19) can be represented as a binary polynomial optimization problem whose coefficients are formal rational linear combinations of $1, \alpha_1, \ldots, \alpha_M$.

Let G_{red} denote the hypergraph containing the supports of cardinality at least two that occur in this binary polynomial representation. Denote by $\bar{G}$ the hypergraph obtained from G by removing nodes in C and adding the complete hypergraph on $N(C)$ for all $C \in \mathcal{C}$. It then follows that $G_{\mathrm{red}} \subseteq \bar{G}$, and hence $\mathrm{itw}(G_{\mathrm{red}}) \leq \mathrm{itw}(\bar{G})$. If $V^+ = \emptyset$, then $\mathcal{C} = \emptyset$ and $G_{\mathrm{red}} \subseteq G$. Hence, assumption (i) implies $\mathrm{itw}(G_{\mathrm{red}}) \in O(\log |V|)$. Now suppose that $V^+ \neq \emptyset$. By Lemma 7, and assumptions (i) and (iii) in the statement of the theorem, we have

$$\mathrm{itw}(G_{\mathrm{red}}) \leq \mathrm{itw}(\bar{G}) \leq \max\Big(\mathrm{itw}(G),\ \max_{C \in \mathcal{C}} |N(C)|\Big) \in O(\log |V|).$$

Furthermore, G_{red} has a polynomial number of edges, since each function ψ_C contributes at most $2^{|N(C)|}$ candidate monomials. Therefore, it remains to solve a binary polynomial optimization problem over $\bar{G}$ with the incidence treewidth $\mathrm{itw}(\bar{G})$.

By Proposition 3, an optimal binary solution for a hypergraph of incidence treewidth t can be found using $2^{O(t)}\mathrm{poly}(|V|, |E|)$ additions and comparisons of objective function values. We store every intermediate objective value as a formal rational linear combination of $1, \alpha_1, \ldots, \alpha_M$. Addition of two such expressions is performed coordinate-wise. To compare two intermediate objective values, it suffices to determine the sign of their difference, which is a rational linear combination of the algebraic numbers $\alpha_1, \ldots, \alpha_M$. Such a difference is a real algebraic expression of the type considered by Burnikel et al. [5]. By Theorem 1 of [5], if such an expression is nonzero, its absolute value is bounded from below by an explicitly computable separation bound. In our setting, the number of algebraic numbers occurring in the expression, the degrees of their defining polynomials, and the encoding lengths of their representations are polynomially bounded in $\mathcal{L}$. Consequently, the binary encoding length of the precision required by this separation bound is polynomially bounded in $\mathcal{L}$. Using the certified quadratically convergent root-refinement algorithm of Kerber and Sagraloff [16], the isolating intervals of the α_j's can therefore be refined to the required precision using a number of unit-cost rational arithmetic operations polynomial in $\mathcal{L}$. Evaluating the resulting rational linear combination then determines its sign exactly. Consequently, every addition and comparison required by Proposition 3 can be performed using polynomially many unit-cost algebraic RAM operations. Since $\mathrm{itw}(G_{\mathrm{red}}) \in O(\log |V|)$, and since M, the number of edges of G_{red}, and all algebraic representation sizes are polynomially bounded in $\mathcal{L}$, the reduced binary polynomial optimization problem can be solved exactly in time polynomial in $\mathcal{L}$ in the unit-cost algebraic RAM model. Let $x^*_{V^-}$ be an optimal binary solution obtained in this manner. For each $C \in \mathcal{C}$, retrieve the minimizer previously computed for $\psi_C(x^*_{N(C)})$. Since the components in $\mathcal{C}$ are pairwise disjoint, these minimizers, together with $x^*_{V^-}$, define a vector $x^* \in [0,1]^V$. By (19), this vector is a minimizer of Problem (1). The binary coordinates of x^* are represented explicitly, and the continuous coordinates are represented by one real univariate representation for each component $C \in \mathcal{C}$. The optimal value is represented by the exact formal sum $f_{V^-}(x^*_{V^-}) + \sum_{C \in \mathcal{C}} \psi_C(x^*_{N(C)})$. All these representations have total encoding length polynomial in $\mathcal{L}$. This completes the proof. □

Let us conclude by commenting on some of the assumptions and limitations of Theorem 2. First, the condition on the size of the continuous components is more restrictive than in the quadratic case of Theorem 1. In the quadratic setting, the complexity of each continuous component is controlled by the parameter κ_C through assumption (ii). Consequently, when $\kappa_C = \Theta(|C|)$, this condition requires $|C| \in O(\log |V|)$, while if $\kappa_C = \Theta(1)$, there is no restriction on the size of C. In contrast, Theorem 2 requires $|C| \in O(1)$ for all $C \in \mathcal{C}$. This difference arises from the continuous subproblems. The quadratic result exploits the special structure of quadratic functions to reduce the local optimization problem to a polynomial number of convex quadratic programs,

with complexity controlled by κ_C. For general polynomial optimization, the fixed-dimensional algorithm used in Proposition 4 requires $(kd)^{O(k)}\text{poly}(\mathcal{L})$ arithmetic operations in dimension k. Since d is allowed to be polynomial in $\mathcal{L}$, this bound yields polynomial-time solvability only when k is bounded by a constant. Thus, the higher-degree result requires a substantially stronger restriction on the continuous components. Second, unlike Theorem 1, which gives polynomial-time solvability in the standard bit model, Theorem 2 is stated in the unit-cost algebraic RAM model. This distinction arises because eliminating a continuous component in the higher-degree case may produce algebraic, rather than rational, optimal values. The unit-cost algebraic RAM model allows these values to be represented symbolically and compared exactly in polynomially many arithmetic operations, as used in the proof of Theorem 2. Finally, Proposition 3 shows that binary polynomial optimization is tractable not only under bounded incidence treewidth, but also under β-acyclicity of the corresponding hypergraph. However, the proof of Theorem 2 relies on eliminating continuous components by introducing complete hypergraphs on their neighborhoods, and this operation introduces β-cycles. Consequently, extending Theorem 2 to β-acyclic interaction hypergraphs would require different techniques.

References

[1] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*. Springer, 2006.

[2] D. Bienstock and G. Muñoz. LP formulations for polynomial optimization problems. *SIAM Journal on Optimization*, 28(2):1121–1150, 2018.

[3] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[4] E. Boros and P.L. Hammer. Pseudo-Boolean optimization. *Discrete applied mathematics*, 123(1):155–225, 2002.

[5] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. *Algorithmica*, 55(1):14–28, 2009.

[6] F. Capelli, A. Del Pia, and S. Di Gregorio. A knowledge compilation take on binary polynomial optimization. *Mathematical Programming*, pages 1–42, 2026.

[7] V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence UAI'08*, 2008.

[8] Y. Crama, P. Hansen, and B. Jaumard. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Applied Mathematics*, 29(2–3):171–185, 1990.

[9] A. Del Pia and S. Di Gregorio. On the complexity of binary polynomial optimization over acyclic hypergraphs. *Algorithmica*, 85:2189–2213, 2023.

[10] A. Del Pia and A. Khajavirad. A polyhedral study of binary polynomial programs. *Mathematics of Operations Research*, 42(2):389–410, 2017.

[11] A. Del Pia and A. Khajavirad. Beyond hypergraph acyclicity: limits of tractability for pseudo-boolean optimization. *arXiv preprint arxiv:2410.23045*, 2024.

[12] A. Del Pia and A. Khajavirad. The pseudo-boolean polytope and polynomial-size extended formulations for binary polynomial optimization. *Mathematical Programming*, 212(1):717–761, 2025.

[13] S. S. Dey and A. Khajavirad. A second-order cone representable class of nonconvex quadratic programs. *Mathematical Programming*, 2026.

[14] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[15] M. Hladík, M. Černỳ, and M. Rada. A new polynomially solvable class of quadratic optimization problems with box constraints: M. hladík et al. *Optimization Letters*, 15(6):2331–2341, 2021.

[16] M. Kerber and M. Sagraloff. Root refinement for real polynomials using quadratic interval refinement. *Journal of Computational and Applied Mathematics*, 280:377–395, 2015.

[17] A. Khajavirad. Tight semidefinite programming relaxations for sparse box-constrained quadratic programs. *arXiv preprint arXiv:2601.18545*, 2026.

[18] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[19] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *Journal of symbolic computation*, 13(3):255–299, 1992.

[20] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[21] S. P. Tarasov and L. G. Khachiyan. Polynomial solvability of convex quadratic programming. In *Doklady akademii nauk*, volume 248, pages 1049–1051. Russian Academy of Sciences, 1979.

[22] P. Tiwari. A problem that is easier to solve on the unit-cost algebraic ram. *Journal of Complexity*, 8(4):393–397, 1992.

[23] S. A. Vavasis. Quadratic programming is in NP. *Information Processing Letters*, 36(2):73–77, 1990.