Solving Integer Quadratic Programming via Explicit and Structural Restrictions

Eduard Eiben,[1] **Robert Ganian,**[2] **Dušan Knop,**[3] **Sebastian Ordyniak**[4]
[1]Department of Informatics, University of Bergen, Norway
[2]Algorithms and Complexity group, Vienna University of Technology, Austria
[3]Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany
[4]Algorithms group, University of Sheffield, UK
eduard.eiben@uib.no, rganian@gmail.com, dusan.knop@gmail.com, sordyniak@gmail.com

Abstract

We study the parameterized complexity of Integer Quadratic Programming under two kinds of restrictions: explicit restrictions on the domain or coefficients, and structural restrictions on variable interactions. We argue that both kinds of restrictions are necessary to achieve tractability for Integer Quadratic Programming, and obtain four new algorithms for the problem that are tuned to possible explicit restrictions of instances that we may wish to solve. The presented algorithms are exact, deterministic, and complemented by appropriate lower bounds.

Introduction

Integer Quadratic Programming (IQP) is a powerful paradigm for solving computationally intractable optimization problems. Indeed, the use of IQP encodings has found numerous applications in artificial intelligence for settings which are not well-suited to the simpler Integer Linear Programming (ILP) paradigm, with examples including Energy Disaggregation (Shaloudegi et al. 2016), Game Theory (Aziz et al. 2018; Iwashita et al. 2016) and Matching (Wang and Ling 2016).

In spite of the close connection between the problems, IQP—i.e., the task of maximizing a quadratic function over integer variables subject to linear constraints—is known to be a substantially more difficult problem than ILP. For instance, while maximizing a linear function over unconstrained integer variables is a trivial task, already this initial problem becomes NP-hard when we consider quadratic functions—even if the domains of the variables are Boolean (Murty and Kabadi 1987). It is a major open problem whether there exists a polynomial time algorithm for IQP over a constant number of variables, i.e., whether one can generalize Lenstra's celebrated result (Lenstra, Jr. 1983) on ILP to the quadratic setting. Until recently, the problem was not even known to be in NP (Del Pia, Dey, and Molinaro 2017) and the first polynomial-time algorithm for IQP over **two** variables was given in 2014 (Del Pia and Weismantel 2014). In 2015, Lokshtanov made substantial progress in this direction by obtaining an analogue of Lenstra's algorithm for IQP when additionally restricting the coefficients that occur in the instance (Lokshtanov 2015)[1]. It is important to note that the above results consider the **total number of variables** which occur in the instance, i.e., including variables which do not occur in the objective function.

The aim of this work is to obtain an understanding of the fine-grained complexity of IQP, and notably to map the conditions under which this notoriously difficult problem becomes tractable. In principle, there are two prominent kinds of properties which could potentially be exploited to solve IQP efficiently:

1. **Explicit** properties of the instance, notably bounding the *domain* and/or the *coefficients*;
2. **Structural** properties of the instance, aimed at capturing the interactions between variables and/or constraints.

So far, the tractability of IQP has been explored primarily through the lens of the **explicit** properties listed above, but we are not aware of many results which would exploit the latter, **structural** properties. This contrasts with the recent developments for ILP, which saw the introduction of several new algorithms and lower bounds that take into account the specific properties of variable-variable and variable-constraint interactions in instances (Ganian, Ordyniak, and Ramanujan 2017; Dvořák et al. 2017; Ganian and Ordyniak 2018; Eiben et al. 2018; Jansen and Kratsch 2015; Koutecký, Levin, and Onn 2018). In all of these works, the authors captured the way variables interacted with each other via constraints through suitably defined graph representations, and the primary research question was to identify natural properties of these graphs (formalized via a *structural parameter*—an integer k) which allow an instance of size n to be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function k. Note that this goal represents a stronger and more fine-grained notion of tractability than merely polynomial-time solvability for each fixed value of k; in particular, algorithms with running time of this form are called *fixed-parameter algorithms* and are central to the parameterized complexity paradigm (Cygan et al. 2013; Niedermeier 2006; Flum and Grohe 2006; Downey and Fellows 2013).

[1]We remark that Lokshtanov's paper is, as of this time, only a preprint. That being said, the result is well known by now and considered to be correct, and has already been accepted and used in published papers by leading researchers (Aziz et al. 2018; Crampton et al. 2017)

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our Results. In this work, we initiate the study of IQP through the lens of structural parameters that capture interactions between variables. Our results include new fixed-parameter algorithms and matching lower bounds for IQP instances under a varied and comprehensive set of conditions.

Before describing our main results, we note that on their own, neither structural parameters (such as the *treewidth* (Robertson and Seymour 1983) or *treedepth* (Nešetřil and de Mendez 2012) of graph representations for IQP instances) nor explicit restrictions (the domain and/or the coefficients) can lead to natural and interesting tractable classes of IQP instances. To argue the former case, it suffices to refer to the lower bounds that are known already for ILP instances—even in that simpler setting, structural parameters must always be accompanied by explicit restrictions of either the domain or the coefficients.

As for the latter, consider the following. On one hand, IQP (as well as ILP) remains NP-hard even when all coefficients in the instance are 1 and all domains are Boolean (Ganian and Ordyniak 2018) and **even when there is no objective function**. On the other hand, IQP is known to remain NP-hard even when there are **no constraints** and all the domains of all variables are Boolean. In our first result, we strengthen the latter lower bound to the case where all **domains and coefficients** are bounded by a constant.

The above lower bounds (and especially the latter one) have critical implications for our endeavor. Indeed, they clearly show that any structural parameters which we hope to use to efficiently solve IQP must restrict **both the objective function and the constraints**; in contrast, for ILP it is often sufficient to use structural parameters merely for restricting the constraints (Ganian, Ordyniak, and Ramanujan 2017; Dvořák et al. 2017; Eiben et al. 2018; Jansen and Kratsch 2015; Koutecký, Levin, and Onn 2018). With this insight, we can now proceed to a high-level description of our results, divided into four settings based on explicit restrictions of instances (a quick glance of our results is also presented in Table 1).

Setting 1: Bounded Coefficients. Our main result for this setting is a fixed-parameter algorithm for IQP parameterized by (a) the maximum coefficient, (b) the arity of the objective function, and (c) the *treedepth* of the *primal graph* representation of the instance. This result can be seen as a generalization of the aforementioned algorithm of Lokshtanov (Lokshtanov 2015) to instances with a large number of variables that do not occur in the objective function. At the same time, it also extends the recent fixed-parameter algorithm for ILP parameterized by (a) and (c) (Koutecký, Levin, and Onn 2018) to IQP (at the cost of requiring parameterization by (b)). The result is complement by lower bounds which show that neither (a) nor (b) can be dropped, and (c) cannot be weakened to the less restrictive parameter *treewidth*.

Setting 2: Bounded Domain. For this setting, we introduce the *mixed primal graph*—a novel graph representation for IQP instances which captures variable-variable interactions that occur via the constraints and also through the objective function. Our main result is then a fixed-parameter algorithm not only for IQP, but even for the more general **Integer Programming** problem which allows arbitrary polynomials in constraints and the objective function. The algorithm is parameterized by the *treewidth* of the mixed primal graph and the maximum domain; as before, we show that neither of these two parameters can be dropped in order to maintain tractability.

Setting 3: Hybrid Restrictions. Next, we consider IQP instances where some variables have bounded coefficients (as in setting 1), while the rest have bounded domain (as in setting 2). Such instances cannot be handled by either of the algorithms presented above. We will show that structural restrictions can still help us obtain rigorous fixed-parameter algorithms. To do so, we introduce and use the *hybrid primal graph*, a representation which also captures "indirect" variable dependencies enabled by the combined presence of high-domain and high-coefficient variables. Our result for this setting is then a fixed-parameter algorithm for IQP parameterized by the treewidth of this graph plus the bounds in the hybrid restrictions.

Setting 4: Unary IQP. In our final setting, we turn to the case where neither the coefficients nor the domain are bounded by a parameter, but the whole instance (including full domain bounds) is encoded in unary. While it is not feasible to aim for fixed-parameter tractability under such general conditions, we show that the problem can be solved in polynomial time when the *treewidth* of the *mixed incidence graph* is bounded by a constant. The mixed incidence graph is based on the same idea as the mixed primal graph used in Setting 2, but builds on the more expressive incidence graph representation used, among others, for ILPs (Ganian, Ordyniak, and Ramanujan 2017). This algorithm is also essentially optimal, as can be shown via complementary lower bounds established already for the ILP setting.

Setting	Representation	Param.	Complexity
Bd. coef.	Primal	td*	**FPT**
Bd. domain	Mixed Primal	tw	**FPT**
Hybrid	Hybrid Primal	tw	**FPT**
Unary	Mixed Incidence	tw	**XP**

Table 1: An outline of the algorithmic results obtained in this paper. Parameterization "td*" stands for treedepth and the arity of the objective function, while "tw" stands for treewidth. XP is the class of problems that can be solved in polynomial time for each fixed value of the parameter.

Preliminaries

Integer Quadratic Programming

Formally, we view an instance $\mathbf{I}$ of INTEGER QUADRATIC PROGRAMMING (IQP) as a tuple $(\mathcal{F}, \eta)$ where $\mathcal{F}$ is a set of linear integer inequalities over variables $X = \{x_1, \ldots, x_n\}$ and η is an integer quadratic polynomial over X, i.e., η contains integer (and non-zero) multiples of terms such as x_1, x_1^2, and $x_1 \cdot x_2$. Inequalities in $\mathcal{F}$ are called constraints, where

each constraint $A \in \mathcal{F}$ ranges over variables $\text{var}(A)$, is said to have arity $|\text{var}(A)| = \ell$, and is assumed to be of the form $c_{A,1}x_{A,1} + c_{A,2}x_{A,2} + \cdots + c_{A,\ell}x_{A,\ell} \leq b_A$; we also define $\text{var}(\mathbf{I}) = X$. Similarly, $\text{var}(\eta)$ is the set of variables which occur in at least one term in η.

Constraints $A \in \mathcal{F}$ such that $|\text{var}(A)| = 1$ are called *box constraints*, and the domain of a variable $x \in X$ is the set of all integers z such that $x \mapsto z$ satisfies all domain constraints of x. We denote by $\text{dom}(x)$ the maximum absolute value in the domain of x.

We will generally use the term *coefficients* to refer to numbers that occur in $\mathcal{F}$ and η. The *maximum coefficient* in $\mathbf{I}$ is then the largest absolute value of an integer that occurs in $\mathbf{I}$, i.e., the maximum number $|z|$ such that z is either the right-hand side of an inequality in $A \in \mathcal{F}$, a multiple of a variable in $A \in \mathcal{F}$, or a multiple of a quadratic term in η. We also use the terms *left-hand side* and *right-hand side coefficient* to refer to the coefficients occurring on the left respectively right side of an inequality in $\mathcal{F}$. Similarly, the *maximum coefficient of a variable* $x \in X$, denoted $\text{coef}(x)$, is the largest absolute value of an integer that multiplies x in a constraint or a term containing x in η.

We say that an IQP instance $\mathbf{I}$ has *unary domain* if the domain of all variables is finite and all box constraints are encoded in unary. Moreover, we say that $\mathbf{I}$ has *unary coefficients* if all coefficients are encoded in unary.

Example 1. *Let $\mathbf{I} = (\mathcal{F}, \eta)$ be the following IQP instance on the variables x_1, x_2, x_3, and x_4:*

$$\begin{aligned} \textit{maximize} \quad & -5x_1^2 - 2019x_2 + 3x_2x_3 & \\ \textit{subject to} \quad & 3x_1 - 7x_2 \leq 12 & (C_1)\\ & 3x_3 - 4x_4 \leq 10 & (C_2)\\ & -2 \leq x_1 \leq 3 & \\ & -3 \leq x_2 \leq 6 & \\ & -4 \leq x_4 \leq 0 & \end{aligned}$$

Then $\text{dom}(x_1) = 3$, $\text{dom}(x_2) = 6$, $\text{dom}(x_3) = \infty$, and $\text{dom}(x_4) = 4$. The maximum coefficient in η is 5, the maximum left-hand side coefficient is 7, the maximum right-hand side coefficient is 12, $\text{coef}(x_2)$ is 7, and the maximum coefficient of $\mathbf{I}$ is 12.

A *(partial) assignment* α is a mapping from (a subset X' of) X to $\mathbb{Z}$. For an assignment α and an inequality (constraint) A of arity ℓ, we denote by $A(\alpha)$ the left-hand side value of A obtained by applying α, i.e., $A(\alpha) = \sum_{cx \in A \wedge x \in X'} c\alpha(x)$, where $cx \in A$ means that cx occurs as a term on the left-hand side of A. We denote by $\eta[X']$ the restriction of η to all terms that are defined solely on the variables in X' and we let $\eta(\alpha)$ be the valuation of $\eta[X']$ given α. An assignment α is called *feasible* if it satisfies every $A \in \mathcal{F}$, i.e., if $A(\alpha) \leq b_A$ for each $A \in \mathcal{F}$. A feasible assignment is also referred to as a *solution*. A solution α that maximizes $\eta(\alpha)$ is called *optimal*; observe that the existence of a solution does not guarantee the existence of an optimal solution (there may exist an infinite sequence of feasible assignments α with increasing values of $\eta(\alpha)$); in other words such an instance is unbounded. Given an instance $\mathbf{I}$, the task in IQP is to compute an optimal solution for $\mathbf{I}$ if one exists, and otherwise to decide whether there exists a feasible assignment.

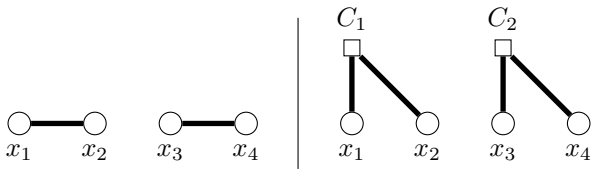

Figure 1: The primal graph (**left**) and the incidence graph (**right**) of the IQP instance given in Example 1.

We note that aside from the inequality representation introduced above and used in other works (Lokshtanov 2015), IQP may equivalently be defined by having $\mathcal{F}$ contain only equalities and with explicit domains for individual variables. While using the former representation is more convenient for formalizing our contributions, none of the results presented herein is sensitive to the specific representation used (changing the representation only changes our parameters by a multiplicative constant).

Finally, we will introduce two basic graph representations of IQP instances, which are also illustrated in Figure 1. Let $\mathbf{I} = (\mathcal{F}, \eta)$ be an IQP instance. The *primal graph* of $\mathbf{I}$, denoted by $P(\mathbf{I})$, is the undirected graph on $\text{var}(\mathbf{I})$ having an edge between x and y if x and y occur together in a constraint in $\mathcal{F}$. The *incidence graph* of $\mathbf{I}$, denoted by $H(\mathbf{I})$, is the bipartite graph with $\text{var}(\mathbf{I})$ on one side, $\mathcal{F}$ on the other side, and there is an edge between $x \in \text{var}(\mathbf{I})$ and $A \in \mathcal{F}$ if and only if x occurs in the constraint A.

Parameterized Complexity

In parameterized algorithmics (Cygan et al. 2013; Niedermeier 2006; Flum and Grohe 2006; Downey and Fellows 2013) the running time of algorithms is studied with respect to a parameter $k \in \mathbb{N}$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Algorithms with this running time are called *fixed-parameter algorithms*.

One of the few (fixed-parameter) tractability results on IQP is due to Lokshtanov (2015) and can be viewed as an analogue to Lenstra's classical result for Integer Linear Programming (Lenstra, Jr. 1983), but with additional conditions.

Proposition 2. *An* IQP *instance* $\mathbf{I}$ *with* $\ell = \max_{x \in \text{var}(\mathbf{I})} \text{coef}(x)$ *and* $k = \max(|\text{var}(\mathbf{I})|, \ell)$ *can be solved in time* $\Lambda(k, \mathbf{I}) = f(k) \cdot \mathbf{I}^{\mathcal{O}(1)}$.

We note that the current proof of Proposition 2 leads to an algorithm with a double-exponential dependency on the parameters, and is hence not yet feasible for use in practice.

Structural Parameters

Here, we give an overview of the graph parameters that will be used in this paper. A *tree-decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair (T, χ), where T is a tree and χ is a function that assigns each tree node t a set $\chi(t) \subseteq V$ of vertices such that the following conditions hold: (1) for every edge $\{u, v\} \in E(G)$ there is a tree node t such that $u, v \in \chi(t)$ and (2) for every vertex $v \in V(G)$, the set of tree nodes t with $v \in \chi(t)$ forms a non-empty subtree of T. The set $\chi(t)$ is also called the bag associated with the tree node t. The *width* of a tree-decomposition (T, χ) is the size of a largest bag minus 1. A tree-decomposition of minimum width is called *optimal*. The *treewidth* of a graph G is the width of an optimal tree-decomposition of G.

For presenting our dynamic programming algorithms, it is convenient to consider tree-decompositions in a normal form; such decompositions are called *nice* (Kloks 1994). A tree-decomposition (T, χ) of G is *nice* if T is rooted at a node r with $\chi(r) = \emptyset$ and each node of T is one of the following four types:

1. a *leaf node*: a node t having no children and $|\chi(t)| = 1$;
2. a *join node*: a node t having exactly two children t_1, t_2, and $\chi(t) = \chi(t_1) = \chi(t_2)$;
3. an *introduce node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \cup \{v\}$ for a vertex v of G;
4. a *forget node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \setminus \{v\}$ for a vertex v of G.

For $t \in V(T)$ we denote by T_t the subtree of T rooted at t and write $\chi(T_t)$ for the set $\bigcup_{t' \in V(T_t)} \chi(t')$.

Proposition 3 (Kloks 1994; Bodlaender 1996)**.** *It is possible to compute an optimal (nice) tree-decomposition of an n-vertex graph G with treewidth k in time $k^{\mathcal{O}(k^3)}n$.*

Our second parameter is called treedepth—a structural parameter closely related to treewidth (Nešetřil and Ossona de Mendez 2012). A useful way of thinking about graphs of bounded treedepth is that they are graphs of bounded treewidth with no long paths.

Since we will only need to directly work with treedepth in a single, technical lemma (notably Lemma 7), we refer to the book of Nešetřil and de Mendez (2012) for the formal definition of the parameter.

IQP Lower Bounds

A simple reduction from SUBSET SUM given in (Murty and Kabadi 1987, Example 2) shows that IQP remains NP-hard even if all variables are binary and the instance has only box constraints. We strengthen this result to instances with maximum coefficient one. The starting point of our reduction is the well-known NP-complete INDEPENDENT SET problem.

Theorem 4. IQP *is* NP*-hard even on instances with binary domain, maximum coefficient one, and where all constraints are box constraints.*

Proof Sketch. Let (G, k) be an instance of INDEPENDENT SET. We construct an instance $\mathbf{I}$ of IQP as follows: We set $\text{var}(\mathbf{I}) = \{ x_v \mid v \in V(G) \}$, $\mathcal{F}$ contains only box constraints restricting the domain of each variable to $\{0, 1\}$, and we set $\eta = (\sum_{v \in V(G)} x_v) - (\sum_{\{u,v\} \in E(G)} x_u x_v)$. To complete the proof, it suffices to show that G has an independent set of size at least k if and only if there is a solution α for $\mathbf{I}$ with $\eta(\alpha) \geq k$. □

Since our reduction preserves the size/value of a solution, it also can be used to carry over the known inapproximability results for INDEPENDENT SET (Håstad 1996) to IQP. Namely, we obtain that IQP cannot be approximated within a factor of $|\text{var}(\mathbf{I})|^{1-\epsilon}$, for $\epsilon > 0$, unless NP $\subseteq$ ZPP.

Bounded Coefficients

In this section we study the setting where the input instances have bounded coefficients. Recently, Koutecký, Levin, and Onn (2018) showed that ILP is fixed parameter tractable parameterized by the treedepth of the primal graph and the maximum left-hand side coefficient occurring in $\mathcal{F}$. The lower bound result from the previous section shows that we cannot hope to obtain the same result for IQP unless we add additional restrictions on the objective function η. As our first and introductory result, we show that IQP is fixed parameter tractable if we parameterize by the maximum coefficient, the treedepth of the primal graph, and $|\text{var}(\eta)|$.

In the following let $\mathbf{I} = (\mathcal{F}, \eta)$ be an IQP instance, $\text{td}(\mathbf{I})$ the treedepth of $P(\mathbf{I})$, and $\ell(\mathbf{I})$ the maximum coefficient of $\mathbf{I}$. We will now state our main result for this section.

Theorem 5. IQP *is fixed-parameter tractable parameterized by $\ell(\mathbf{I})$, $\text{td}(\mathbf{I})$, and $|\text{var}(\eta)|$.*

Before we proceed to the proof of above theorem, let us first argue its tightness. As mentioned previously, Theorem 4 shows that dropping $|\text{var}(\eta)|$ from the parameterization results in the problem becoming NP-hard even when the remaining two parameters are bounded by a constant. The following two known results then similarly rule out the possibility of dropping (or, in the case of treedepth, weakening) any of the other two parameters; note that ILP-feasibility is precisely IQP with $\eta = 0$.

Theorem 6 (Ganian and Ordyniak 2018, Theorem 12 and 13)**.** ILP*-feasibility is* NP*-hard even for instances:*

- *with primal graphs of constant treedepth,*
- *with maximum coefficient one and primal graphs of treewidth at most two.*

The proof of Theorem 5 uses the same strategy as the proof of an analogous theorem for ILP given by Ganian and Ordyniak (2018, Theorem 6). In particular, the algorithm for ILP can be divided into three steps:

(1) Compute an optimal treedepth decomposition of $P(\mathbf{I})$,

(2) Using a bottom-up pruning procedure, the original ILP instance is transformed into an equivalent ILP instance whose number of variables is bounded by a function of the parameters, and

(3) the pruned instance is now solved by invoking Lenstra's algorithm (Lenstra, Jr. 1983)

Step (1) remains the same also in our proof of Theorem 5. In order to perform Step (2), we will obtain a more refined version of (Ganian and Ordyniak 2018, Lemma 11), which will then form the main tool for our pruning procedure.

Lemma 7. *There exist computable functions f, g and an algorithm that takes as input $\mathbf{I} = (\mathcal{F}, \eta)$ and an optimal treedepth decomposition of $P(\mathbf{I})$, runs in time $\mathcal{O}(f(\ell(\mathbf{I}), \mathrm{td}(\mathbf{I}), |\mathrm{var}(\eta)|) \cdot |\mathbf{I}|)$ and outputs an IQP instance $\mathbf{I}'$ containing at most $g(\ell(\mathbf{I}), \mathrm{td}(\mathbf{I}), |\mathrm{var}(\eta)|)$ variables and whose maximum coefficient is no larger than the maximum coefficient of $\mathbf{I}$ with the following property: there exists a solution α of $\mathbf{I}$ of value w if and only if there exists a solution α' of $\mathbf{I}'$ of value w. Moreover, α can be computed from α' in linear time.*

Proof Sketch. The proof is very similar to the proof of Lemma 11 in (Ganian and Ordyniak 2018). The main difference is due to a different definition of the primal graph in the cited proof, which additionally contains a clique on all variables contained in the objective function. It can be shown that adding a clique on all variables contained in the objective function only increases the treedepth by at most $|\mathrm{var}(\eta)|$. After that, the remainder of the proof is virtually identical. □

There are two main differences in the formulation of the above lemma compared to Lemma 11 in (Ganian and Ordyniak 2018): (1) Both the running-time and the size of the reduced instance are given in terms of our parameters $\ell(\mathbf{I})$, $\mathrm{td}(\mathbf{I})$, and $|\mathrm{var}(\mathbf{I})|$ and (2) we state explicitly that the coefficients of the pruned instance are no larger than the coefficients in the original instance. The latter point is critical for our next Step (3), since we are required to use Proposition 2 instead of Lenstra's algorithm in order to solve the instance obtained from Lemma 7.

Proof Sketch of Theorem 5. The algorithm proceeds as follows. First, we compute a treedepth decomposition of $P(\mathbf{I})$ (Nešetřil and de Mendez 2012). Second, we invoke Lemma 7 to obtain an equivalent instance $\mathbf{I}'$ with only a few variables. Finally, we use Proposition 2 to solve $\mathbf{I}'$. □

We remark that due to the employed techniques, the running-time of the algorithm arising from Theorem 5 has a non-elementary dependency on the parameters. Hence, the result should be viewed primarily as a classification result.

Bounded Domain

It is known that ILP parameterized by the treewidth of the primal graph as well as the maximum absolute domain value of any variable is fixed-parameter tractable (Jansen and Kratsch 2015). Because of Theorem 4 this is not the case for IQP, i.e., IQP remains NP-hard even when all variables have a binary domain and the primal graph is edgeless. This is because having non-linear interactions between the variables in the optimization function implicitly allows the expression of linear constraints. To obtain tractability for IQP instances, it hence becomes necessary to also restrict the structure of the non-linear interactions between variables within the optimization function. To this end we propose the

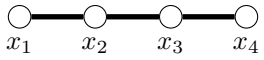

Figure 2: The mixed primal graph of the IQP instance given in Example 1.

mixed primal graph, denoted by $P^M(\mathbf{I})$, which extends the primal graph by adding an edge between any two distinct variables x and y that occur together in a mixed term of η (see Figure 2 for an illustration of the mixed primal graph).

Our main result for this section is establishing that the tractability for ILP using bounded domain and bounded treewidth of the primal graph carries over to IQP with the use of the mixed primal graph. In fact, we show tractability for the much more general *Integer Programming* (IP) problem, i.e., the generalization of IQP where both the objective function as well as the constraints are allowed to be arbitrary polynomials over the set of variables. Since known hardness results for ILP given in Theorem 6 carry over to IQP, it is essentially the case that the much more general IP exhibits the same complexity behavior on instances of bounded **mixed** primal treewidth as ILP on instances of bounded primal treewidth. Note, however, that the tractability results given in Theorem 5 for treedepth do not carry over to IP, since it is known that IP is undecidable even for instances with constantly many variables (Köppe 2012).

In the following let $\mathbf{I}$ be an instance of IQP, $\mathcal{T} = (T, \chi)$ be a nice tree-decomposition of $P^M(\mathbf{I})$ of width ω, and let $d = \max_{x \in \mathrm{var}(\mathbf{I})} \mathrm{dom}(x)$.

Theorem 8. *An* IP *instance $\mathbf{I}$ can be solved in time $O((2d+1)^{\omega+2}|\mathbf{I}|)$, given that a nice tree-decomposition of $P^M(\mathbf{I})$ of width ω is provided with the input.*

Before we proceed, let us formalize some additional notation used in this section. For a subset V' of $\mathrm{var}(\mathbf{I})$, we denote by $\mathcal{F}[V']$ the subset of $\mathcal{F}$ containing all constraints A such that $\mathrm{var}(A) \subseteq V'$. Let $\alpha \colon V' \to \mathbb{Z}$ be a (partial) assignment, $x \in \mathrm{var}(\mathbf{I})$, and $s \in \mathbb{Z}$. We denote by $\alpha_{x \to s}$ the assignment obtained from α after setting $\alpha(x) = s$. Finally, we denote by $\mathbf{I}[V']$ the subinstance of $\mathbf{I}$ induced by V', i.e., the IP instance $(\mathcal{F}[V'], \eta[V'])$.

We prove the theorem by giving a dynamic programming algorithm on $\mathcal{T}$ that computes the set $\mathcal{R}(t)$ of all valid records for every node $t \in V(T)$, where a *record* is a pair (α, o) consisting of an assignment $\alpha \colon \chi(t) \to [-d, d]$ and an integer o. A record (α, o) is *valid* for t if o is the maximum value for $\eta(\alpha')$ over all feasible assignments $\alpha' \colon \chi(T_t) \to [-d, d]$ for $\mathbf{I}[\chi(T_t)]$ that agree with α on the variables in $\chi(t)$. We show next how to compute the set of all valid records for each of the four node types of $\mathcal{T}$.

Lemma 9. *$\mathcal{R}(t)$ can be computed from $\mathcal{R}(t')$ in time $\mathcal{O}(|\mathcal{R}(t')| \cdot d \cdot |I|)$ for an introduce node t of T with child t'.*

Proof Sketch. Let x be the variable introduced by t, i.e., $\chi(t) = \chi(t') \cup \{x\}$. In this case $\mathcal{R}(t)$ is the set of all pairs $(\alpha, o + (\eta[\chi(t)] \setminus \eta[\chi(t')])(\alpha))$ with $(\alpha[\chi(t')], o) \in \mathcal{R}(t')$ and $\alpha \colon \chi(t) \to [-d, d]$ is a feasible assignment for $\mathbf{I}[\chi(t)]$. Hence to obtain $\mathcal{R}(t)$ it suffices to check whether the assignment $\alpha_{x \to v}$ is feasible for $\mathbf{I}[\chi(t)]$ for every $(\alpha, o) \in$

$\mathcal{R}(t')$ and every $v \in [-d, d]$ and to evaluate the terms in $(\eta[\chi(t)] \setminus \eta[\chi(t')])$ for the assignment $\alpha_{x \to v}$, which is possible since all "new" constraints, i.e., the constraints in $\mathcal{F}(\chi(t)) \setminus \mathcal{F}(\chi(t'))$, only contain variables in $\chi(t)$ and the same also holds for all terms in $\eta(\chi(t)) \setminus \eta(\chi(t'))$. □

The computation of $\mathcal{R}(t)$ for forget and join nodes follows similar lines as the proof of Lemma 9, and is trivial for leaf nodes. We summarize below.

Lemma 10. *$\mathcal{R}(t)$ can be computed in time $\mathcal{O}((2d+1)^{\omega+1} d|\mathbf{I}|)$ for a leaf, forget, and join-node t.*

Proof Sketch for Theorem 8. The algorithm computes the set of all valid records $\mathcal{R}(t)$ for every node t of T using a bottom-up dynamic programming algorithm starting in the leaves of T, until it computes $\mathcal{R}(r)$ for the root node r. It then outputs o if $\mathcal{R}(r) = \{(\emptyset, o)\}$ and otherwise correctly returns that $\mathbf{I}$ has no optimal solution. □

Theorem 8 together with Proposition 3 now imply that IP is fixed-parameter tractable parameterized by the maximum domain and the treewidth of $P^M(\mathbf{I})$. Since IQP is a special case of IP, we obtain:

Corollary 11. IQP *is fixed-parameter tractable parameterized by maximum domain and the treewidth of $P^M(\mathbf{I})$.*

Hybrid Restrictions

In this section, we turn our attention to instances where each variable is restricted in one of two possible ways: either the domain is bounded, or its maximum coefficient is bounded. Naturally, such instances are more general than those covered in the first two settings, and hybrid restrictions may occur naturally when modeling systems where variables represent elements with different properties. Even though neither of the results and graph representations introduced up to this point can be used to obtain fixed-parameter algorithms for such instances, we will show that one can still exploit the treewidth of variable interactions to deal with such instances.

Formally, the *hybrid bound* $\text{hyb}(\mathbf{I})$ of an IQP instance $\mathbf{I}$ is defined as $\max_{x \in \text{var}(\mathbf{I})} \min(\text{dom}(x), \text{coef}(x))$. Observe that every IQP instance with domain at most k or maximum coefficient at most k has a hybrid bound of at most k, but the converse is not true. We note that IQP remains NP-hard even when both $\text{hyb}(\mathbf{I})$ and the treewidth of the mixed primal graph are bounded by a constant; indeed, even ILP-feasibility remains NP-hard when restricted to instances with a maximum coefficient of 1 and primal graphs of treewidth 2 (Ganian and Ordyniak 2018). The underlying reason for this is that variables with unbounded domain can, in some sense, carry long-distance interactions between variables. We introduce the *hybrid primal graph* representation in order to account for this and capture such interactions. For the purposes of this section, we will say that a variable x is *large-domain* if $\text{dom}(x) > \text{hyb}(\mathbf{I})$.

Definition 12. *The* hybrid primal graph *of $\mathbf{I} = (\mathcal{F}, \eta)$ is the graph $P^H(\mathbf{I}) = (V, E \cup E')$ where $(V, E) = P^M(\mathbf{I})$ and E' is the set of all variable pairs that are connected by a path in $P^M(\mathbf{I})$ whose internal vertices are large-domain variables.*

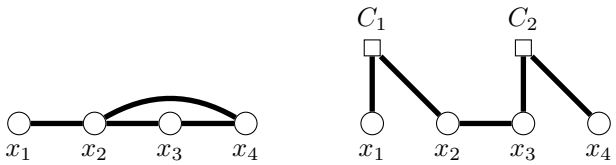

Figure 3: **Left:** The hybrid primal graph of the IQP instance $\mathbf{I}$ given in Example 1. Observe that $\text{hyb}(\mathbf{I}) = 7$ and x_3 is the only large-domain variable. **Right:** The mixed incidence graph of the IQP instance given in Example 1.

We state our main result for this section below.

Theorem 13. IQP *is fixed-parameter tractable parameterized by* $\text{hyb}(\mathbf{I})$ *and* $\text{tw}(P^H(\mathbf{I}))$.

Our first task on the way to proving Theorem 13 is to define a type of tree-decomposition that treats the *large-domain variables* (i.e., variables x such that $\text{dom}(x) > \text{hyb}(\mathbf{I})$) in a certain way. To this end, we call a tree-decomposition of $P^H(\mathbf{I})$ *hybrid* if:

1. for each large-domain variable x, there exists a leaf node t such that $x \in \chi(t)$, and
2. every leaf node t has a parent q (called the *guard node*) of degree 2 such that $\chi(q)$ is obtained by removing all large-domain variables from $\chi(t)$, and
3. all nodes other than guard and leaf nodes satisfy the conditions of nice tree-decompositions, i.e., are either forget, introduce or join nodes.

It is not difficult to show that any nice tree-decomposition of $P^H(\mathbf{I})$ can be transformed into a hybrid one in linear time—the core idea is to create new leaves that will contain the large-domain variables together with their neighborhood.

Our strategy for obtaining the fixed-parameter algorithm of Theorem 13 will now be the following. We will employ dynamic programming along the lines of Theorem 8; in particular, at each node t that is neither a guard nor a leaf node, we will use the same records $\mathcal{R}(t)$ and apply Lemmas 9 and 10. Hence the crucial part will be to obtain the records $R(q)$ for each guard node q.

Lemma 14. *Let (T, χ) be a hybrid width-k tree-decomposition of $P^H(\mathbf{I})$ and let q be a guard node in T. Then it is possible to compute $\mathcal{R}(q)$ in time $\Lambda(\text{hyb}(\mathbf{I}), \mathbf{I}) \cdot (2\text{hyb}(\mathbf{I}) + 1)^k \cdot |\mathbf{I}|^{\mathcal{O}(1)}$.*

Proof Sketch. The procedure has two steps. First, we branch over all feasible assignments of the variables in $\chi(q)$. Second, for each such assignment α we apply Proposition 2 on the instance resulting by the application of α on $\mathbf{I}[\chi(p)]$, where p is the unique child of q. It can be shown that this instance satisfies the conditions required by Proposition 2. □

We can now proceed to a proof of the desired result.

Proof of Theorem 13. We begin by computing an optimal hybrid tree-decomposition. Next, we use Lemma 14 to compute the records $\mathcal{R}(q)$ for each guard node q in T. Once that

is done, we proceed by dynamically computing the records for all other nodes in T in a leaves-to-root fashion by invoking Lemmas 9 and 10. Finally, we solve $\mathbf{I}$ by reading the record at the root node r of T, in the same manner as in the proof of Theorem 8. The resulting runtime can be upper-bounded by $|\text{var}(\mathbf{I})|$ times the runtime of Lemma 14. □

Unary Coefficients and Unary Domain

In this section we generalize the polynomial-time algorithm for ILP with unary coefficients and unary domain using incidence treewidth (Ganian, Ordyniak, and Ramanujan 2017) to IQP. Again, because of our hardness result (Theorem 4), we will need additional restrictions on the variable dependencies expressed by mixed terms in the objective function. We therefore introduce the *mixed incidence graph* of an IQP instance $\mathbf{I}$, denoted by $H^M(\mathbf{I})$, which is obtained from the incidence graph by adding edges between any two distinct variables occurring in mixed terms of the objective function (see Figure 3 for an illustration).

In the following let $\mathbf{I} = (\mathcal{F}, \eta)$ be an IQP instance and let $\mathcal{T} = (T, \chi)$ be a nice tree-decomposition of $H^M(\mathbf{I})$ of width ω. Similarly to the previous approach employed for ILP (Ganian, Ordyniak, and Ramanujan 2017), we will first show a slightly stronger result that will then imply tractability for unary domain and unary coefficients for constant ω. Let Γ be the maximum absolute value of $A(\alpha)$ over every constraint $A \in \mathcal{F}$ and every feasible assignment α for $\mathbf{I}$.

Theorem 15. IQP *can be solved in time* $\mathcal{O}(\Gamma^{\omega+3}\omega^3|I|)$ *given that a nice tree decomposition of* $H^M(\mathbf{I})$ *of width* ω *is provided with the input.*

As the algorithm uses dynamic programming on $\mathcal{T}$, the overall structure of the algorithm is the same as for the algorithm presented in Theorem 8. We will hence only present the records together with the corresponding replacements of the lemmas required for showing the correctness for the four different node types of $\mathcal{T}$.

To distinguish between variables and constraints contained within the bags of $\mathcal{T}$, we use $\chi_{\text{var}}(t)$ and $\chi_A(t)$ to denote the sets $\chi(t) \cap \text{var}(\mathbf{I})$ and $\chi(t) \cap \mathcal{F}$, respectively. A *record* for a node $t \in V(T)$ is a triple (α, γ, o), where:

- $\alpha\colon \chi_{\text{var}}(t) \to [-\Gamma, \Gamma]$,
- $\gamma\colon \chi_A(t) \to [-\Gamma, \Gamma]$, and
- o is a non-negative integer.

Note here that we use that Γ also bounds the domain of every variable. We say that a record (α, γ, o) for a node $t \in V(T)$ is *valid* if o is the maximum value of $\eta(\alpha')$ for any assignment $\alpha'\colon \chi_{\text{var}}(T_t) \to [-\Gamma, \Gamma]$ satisfying:

R1 $\alpha'(x) = \alpha(x)$ for every variable $x \in \chi_{\text{var}}(t)$,

R2 $A(\alpha') = \gamma(A)$ for every constraint $A \in \chi_A(t)$.

We denote by $\mathcal{R}(t)$ the set of all valid records for t. The following lemma summarizes the computation for leaf, introduce, forget, and join nodes.

Lemma 16. $\mathcal{R}(t)$ *can be computed in time* $O(\Gamma^{\omega+2}\omega^2 \log \Gamma)$ *for a leaf, introduce, forget, or join node* t *of* T.

Using Lemma 16, the proof of Theorem 15 now follows along the same lines as the proof of Theorem 8.

Note that together with Proposition 3, Theorem 15 implies polynomial-time tractability of IQP instances with bounded mixed incidence treewidth as long as Γ can be bounded by a polynomial of the input size. Namely, all tractability results considered in (Ganian, Ordyniak, and Ramanujan 2017) for ILP and bounded incidence treewidth carry over to IQP and bounded **mixed** incidence treewidth:

Corollary 17. IQP *is solvable in polynomial-time for instances with bounded mixed incidence treewidth that additionally satisfy either:*

- *unary domain and unary coefficients, or*
- *non-negativity and unary coefficients on the right-hand side of constraints.*

An IQP instance is non-negative *if so is the domain of all variables as well as all coefficients occurring on the left-hand side of constraints.*

Similarly, the hardness results given for ILP-feasibility by Ganian, Ordyniak and Ramanujan (2017) carry over to IQP, i.e., IQP is NP-hard for non-negative instances with binary domain and mixed incidence treewidth at most three as well as instances with maximum coefficient two and mixed incidence treewidth at most three.

Note, however, that we cannot (at least not directly) generalize our tractability result for IQP for bounded mixed incidence treewidth to IP. Informally this is because even the mixed incidence treewidth does not take into account dependencies between variables that arise from mixed terms in the constraints.

Theorem 18. *IP-feasibility is* NP*-hard even for instances with mixed incidence treewidth one, binary domain variables, and maximum coefficient one.*

Proof Sketch. The reduction is similar to our reduction from INDEPENDENT SET used in Theorem 4 only that this time we model the objective function as a constraint. □

We note that it would be possible to generalize our algorithm given in Theorem 15 to IP, if we were to extend the mixed incidence graph by adding all edges between variables that appear together in mixed terms of the constraints. Since IQP and not IP is the focus of this paper, we decided against providing the algorithm in its full generality.

Concluding Notes

Our results provide the first (and already surprisingly detailed) picture of the complexity of IQP w.r.t. natural structural and syntactical restrictions. To our surprise, IQP behaves very similar to the much simpler ILP once the role of the mixed terms in the optimization function is taken into account. Our results raise several interesting questions concerning the complexity of IQP and even ILP, and we highlight two such questions below:

1. Is IQP fixed-parameter tractable parameterized by the treedepth of the mixed primal graph and the maximum coefficient, i.e., can treedepth be used even when optimizing functions of large arity?

2. What is the complexity of IQP (and ILP) parameterized by the treedepth of the mixed primal graph and hyb($\mathbf{I}$)?

The latter question is particularly interesting in light of the recently growing interest in multi-stage stochastic ILPs (Koutecký, Levin, and Onn 2018), whereas ILP instances with bounded primal treedepth and hyb($\mathbf{I}$) are significantly more general than multi-stage stochastic ILPs.

Acknowledgments. The authors wish to thank the reviewers for their insightful comments. Eduard Eiben was supported by Pareto-Optimal Parameterized Algorithms (ERC Starting Grant 715744) and by Parameterized Complexity for Practical Computing (RCN Toppforsk Grant 274526). Robert Ganian acknowledges support from the FWF Austrian Science Fund (Project P31336: **NFPC**) and is also affiliated with FI MU, Brno, Czech Republic. Dušan Knop is partially supported by DFG project MaMu (NI369/19) and is also affiliated with Department of Theoretical Computer Science, FIT, ČVUT, Prague, Czech Republic.

References

Aziz, H.; Gaspers, S.; Lee, E. J.; and Najeebullah, K. 2018. Defender stackelberg game with inverse geodesic length as utility metric. In *Proc. AAMAS 2018*, 694–702. International Foundation for Autonomous Agents and Multiagent Systems.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Crampton, J.; Gutin, G. Z.; Koutecký, M.; and Watrigant, R. 2017. Parameterized resiliency problems via integer linear programming. In *Proc. CIAC 2017*, volume 10236 of *Lecture Notes in Computer Science*, 164–176.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2013. *Parameterized Algorithms*. Texts in Computer Science. Springer.

Del Pia, A., and Weismantel, R. 2014. Integer quadratic programming in the plane. In Chekuri, C., ed., *Proc. SODA 2014*, 840–846.

Del Pia, A.; Dey, S. S.; and Molinaro, M. 2017. Mixed-integer quadratic programming is in NP. *Math. Program.* 162(1-2):225–240.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Dvořák, P.; Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2017. Solving integer linear programs with a small number of global variables and constraints. In Sierra, C., ed., *Proc. IJCAI 2017*, 607–613. ijcai.org.

Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2018. Unary integer linear programming with structural restrictions. In *Proc. IJCAI 2018*, 1284–1290. ijcai.org.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer.

Ganian, R., and Ordyniak, S. 2018. The complexity landscape of decompositional parameters for ilp. *Artificial Intelligence* 257:61 – 71.

Ganian, R.; Ordyniak, S.; and Ramanujan, M. S. 2017. Going beyond primal treewidth for (M)ILP. In Singh, S. P., and Markovitch, S., eds., *Proc. AAAI 2017*, 815–821. AAAI Press.

Håstad, J. 1996. Clique is hard to approximate within $n^{1\text{-epsilon}}$. In *Proc. FOCS 1996*, 627–636.

Iwashita, H.; Ohori, K.; Anai, H.; and Iwasaki, A. 2016. Simplifying urban network security games with cut-based graph contraction. In *Proc. AAMAS 2016*, AAMAS '16, 205–213. International Foundation for Autonomous Agents and Multiagent Systems.

Jansen, B. M. P., and Kratsch, S. 2015. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In *Proc. ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, 779–791. Springer.

Kloks, T. 1994. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer.

Köppe, M. 2012. On the complexity of nonlinear mixed-integer optimization. In *Mixed Integer Nonlinear Programming*. Springer. 533–557.

Koutecký, M.; Levin, A.; and Onn, S. 2018. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In Chatzigiannakis, I.; Kaklamanis, C.; Marx, D.; and Sannella, D., eds., *Proc. ICALP 2018*, volume 107 of *LIPIcs*, 85:1–85:14. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Lenstra, Jr., H. W. 1983. Integer programming with a fixed number of variables. *Math. Oper. Res.* 8(4):538–548.

Lokshtanov, D. 2015. Parameterized integer quadratic programming: Variables and coefficients. *CoRR* abs/1511.00310.

Murty, K. G., and Kabadi, S. N. 1987. Some np-complete problems in quadratic and nonlinear programming. *Math. Program.* 39(2):117–129.

Nešetřil, J., and de Mendez, P. O. 2012. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer.

Nešetřil, J., and Ossona de Mendez, P. 2012. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford: Oxford University Press.

Robertson, N., and Seymour, P. D. 1983. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B* 35(1):39–61.

Shaloudegi, K.; György, A.; Szepesvári, C.; and Xu, W. 2016. SDP relaxation with randomized rounding for energy disaggregation. In *Proc. NIPS 2016*, 4979–4987.

Wang, T., and Ling, H. 2016. Path following with adaptive path estimation for graph matching. In *Proc. AAAI 2016*, 3625–3631.