Faster Algorithms Parameterized by Clique-width

Sang-il Oum[1]*, Sigve Hortemo Sæther[2]**, and Martin Vatshelle[2]

[1] Department of Mathematical Sciences, KAIST
South Korea
`sangil@kaist.edu`

[2] Department of Informatics, University of Bergen
Norway
`[sigve.sether,vatshelle]@ii.uib.no`

Abstract. Many NP-hard problems, such as Dominating Set, are FPT parameterized by clique-width. For graphs of clique-width k given with a k-expression, Dominating Set can be solved in $4^k n^{\mathcal{O}(1)}$ time. However, no FPT algorithm is known for computing an optimal k-expression. For a graph of clique-width k, if we rely on known algorithms to compute a $(2^{3k}-1)$-expression via rank-width and then solving Dominating Set using the $(2^{3k}-1)$-expression, the above algorithm will only give a runtime of $4^{2^{3k}} n^{\mathcal{O}(1)}$. There have been results which overcome this exponential jump; the best known algorithm can solve Dominating Set in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(1)}$ by avoiding constructing a k-expression. We improve this to $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. Indeed, we show that for a graph of clique-width k, a large class of domination and partitioning problems (LC-VSP), including Dominating Set, can be solved in $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. Our main tool is a variant of rank-width using the rank of a 0-1 matrix over the rational field instead of the binary field.

Keywords: clique-width, parameterized complexity, dynamic programming, generalized domination, rank-width

1 Introduction

Parameterized complexity is a field of study dedicated to solving NP-hard problems efficiently on restricted inputs, and has grown to become a well known field over the last 20 years. Especially the subfields of Fixed Parameter Tractable (FPT) algorithms and kernelizations have attracted the interest of many researchers. Parameterized algorithms measure the runtime in two parameters; the input size n and a secondary measure k (called a parameter, either given as part of the input or being computable from the input). An algorithm is FPT

* Supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2011-0011653).

** Supported by the Research Council of Norway.

if it has runtime $f(k)n^{\mathcal{O}(1)}$. Since we study NP-hard problems, we must expect that $f(k)$ is exponentially larger than n for some instances. However, a good parameter is one where $f(k)$ is polynomial in n for a large class of inputs. For a survey on parameterized complexity and FPT, we refer the reader to [10,8,18].

The clique-width of a graph is a well studied parameter in parameterized complexity theory. Courcelle, Makowsky, and Rotics [6] showed that, for an input graph of clique-width at most k, every problem expressible in $MSOL_1$ (monadic second-order logic of the first kind) can be solved in FPT time parameterized by k if a k-expression[3] for the graph, that is a certificate that the graph has clique-width at most k, is given together with the input graph. Later, Oum and Seymour [20] gave an algorithm to find a $(2^{3k+2}-1)$-expression of a graph having clique-width at most k in time $2^{3k}n^{\mathcal{O}(1)}$.[4] By combining these results, we deduce that for an input graph of clique-width at most k, every $MSOL_1$ problem is FPT, even if a k-expression is not given as an input. However the dependency in k is huge and can not be considered of practical interest. In order to increase the practicality of FPT algorithms, it is very important to control the runtime as a function of k.

If we rely on first calculating a k-expression and then doing dynamic programming on the calculated k-expression, we have two ways to make improvements; either we improve the algorithm that uses the k-expression, or we find a better approximation for clique-width. Given a k-expression, Independent Set and Dominating Set can be solved in time $2^k n^{\mathcal{O}(1)}$ [12] and $4^k n^{\mathcal{O}(1)}$ [2], respectively. Lokshtanov, Marx and Saurabh [15] show that unless the Strong ETH fails[5], Dominating Set can not be solved in $(3-\epsilon)^k n^{\mathcal{O}(1)}$ time when given a k-expression[6]. By this, there is not much room for improvement in the existing algorithms when a k-expression is given.

Since the best approximation of clique-width is exponential in the optimal clique-width, even for the simple NP-hard problems Independent Set and Dominating Set, all known algorithms following this procedure has a runtime where the dependency is double exponential in the clique-width. The question of finding a better approximation for clique-width is an important and challenging open question in parameterized complexity. However, there is a way around this by not using a k-expression at all: Bui-Xuan, Telle and Vatshelle [3] showed that by doing dynamic programming directly on a rank decomposition, Dominating Set can be solved in $2^{k^2}n^{\mathcal{O}(1)}$ for graphs of rank-width k and hence also for graphs of clique-width k, since rank-width $\leq$ clique-width [20]. In this paper we improve on this algorithm.

---

[3] See [7] for the definition clique-width and k-expressions.

[4] Later, Oum [19] obtained an improved algorithm to find a $(2^{3k}-1)$-expression of a graph having clique-width at most k in time $2^{3k}n^{\mathcal{O}(1)}$.

[5] The Strong Exponential Time Hypothesis (Strong ETH) states that SAT can not be solved in $\mathcal{O}((2-\epsilon)^n)$ time for any constant $\epsilon > 0$. Here n denotes the number of variables.

[6] Their proof uses pathwidth, but the statement holds since clique-width is at most 1 higher than pathwidth.

Table 1. A table of some vertex subset properties whose optimization problems belong to LC-VSP. The meaning of the problem specific constant $d(\pi)$ is discussed in subsection 2.3.

$d(\pi)$	Standard name
d	d-Dominating set
$d+1$	Induced d-Regular Subgraph
d	Subgraph of Min Degree $\geq d$
$d+1$	Induced Subg. of Max Degree $\leq d$
2	Strong Stable set or 2-Packing
2	Perfect Code or Efficient Dom. set
2	Total Nearly Perfect set
2	Weakly Perfect Dominating set
2	Total Perfect Dominating set
2	Induced Matching
2	Dominating Induced Matching
2	Perfect Dominating set
1	Independent set
1	Dominating set
1	Independent Dominating set
1	Total Dominating set

Table 2. A table of some homomorphism problems in LC-VSP for fixed simple graph H. These are expressible with a degree constraint matrix D_q where $q(\pi) = |V(H)|$. The meaning of D_q, $d(\pi)$ and $q(\pi)$ is explained in subsection 2.3.

$d(\pi)$	Standard name
1	H-coloring or H-homomorphism
1	H-role assignment or H-locally surj. hom.
2	H-covering or H-locally bij. hom.
2	H-partial covering or H-locally inj. hom.

We initiate a study to find a parameter P which is lower than both treewidth and clique-width, and for which Independent Set and Dominating Set are solvable on any graph G in runtime better than $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$ time, where $k = P(G)$. We give such a parameter for which we obtain a runtime of $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$, not only for Independent Set and Dominating Set but a wide range of problems. We study a particular subclass of $MSOL_1$ problems, called the *locally checkable vertex subset and partitioning problems* (LC-VSP problems) which contains among others Independent Set and Dominating Set. Tables 1 and 2 list some well known problems in LC-VSP.

Bui-Xuan et al. [4] showed that by using rank-width as a parameter we can get a runtime of $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$ for any of the LC-VSP problems. The idea of [4] is to introduce an alternative width parameter nec_d-width (where d is a positive integer depending on the specific LC-VSP problem being solved, as explained in subsection 2.3) of graphs such that

– for every decomposition of rank-width k, the nec_d-width is at most $2^{\mathcal{O}\left(k^2\right)}$, and
– each LC-VSP problem can be solved in time $\mathcal{O}\left(n^4 p^{\mathcal{O}(1)}\right)$, when a decomposition of nec_d-width p is given together with the input graph.

Since rank-width is never more than clique-width, by the approximation algorithm of rank-width by Oum and Seymour [20], a decomposition certifying that the nec_d-width of G is at most $2^{\mathcal{O}\left(\mathrm{cw}(G)^2\right)}$ can be found in time $2^{3\,\mathrm{cw}(G)}n^{\mathcal{O}(1)}$. It follows that every fixed LC-VSP-problem can be solved in $2^{\mathcal{O}\left(\mathrm{cw}(G)^2\right)}n^{\mathcal{O}(1)}$-time parameterized by the clique-width $\mathrm{cw}(G)$ of the input graph G.

In this paper we improve on these results by using a slightly modified definition of rank-width, which we call *$\mathbb{Q}$-rank-width*, based on the rank function over the rational field instead of the binary field. The idea of using fields other than the binary field for rank-width was investigated earlier in [14]. We will show the following:

– For any graph, its $\mathbb{Q}$-rank-width is no more than its clique-width.
– There is an algorithm to find a decomposition confirming that $\mathbb{Q}$-rank-width is at most $3k+1$ for graphs of $\mathbb{Q}$-rank-width at most k in time $2^{3k}n^{\mathcal{O}(1)}$.
– If a graph has $\mathbb{Q}$-rank-width at most k, then the nec_d-width is at most $2^{\mathcal{O}(k\log k)}$.

These results allow us to use $\mathbb{Q}$-rank-width instead of rank-width to improve the runtime of the algorithm of Bui-Xuan et al. [4]. By using $\mathbb{Q}$-rank-width, the algorithm runs in time $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$ for graphs of clique-width at most k. This suggests that for LC-VSP problems, the parameter $\mathbb{Q}$-rank-width is more useful than both clique-width and rank-width.

We also relate the parameter $\mathbb{Q}$-rank-width to other existing parameters. There are several factors affecting the quality of a parameter, such as: Can we compute or approximate the parameter? Which problems can we solve in FPT time? Can we reduce the exponential dependency in the parameter for specific problems? And, how large and natural is the class of graphs having a bounded parameter value?

This paper is organized as follows: In Section 2 we define branch-decompositions and introduce the main parts of the framework used by Bui-Xuan et al. [4], including the general algorithm they give for LC-VSP problems. Section 3 revolves around $\mathbb{Q}$-rank-width and is where the results of this paper reside. We show how $\mathbb{Q}$-rank-width relates to nec_d-width and clique-width, and reveal why we have a good FPT algorithm for approximating a decomposition. At the end of this section we add all the parts together to form the main theorem, which is an improved upper bound on solving LC-VSP problems parameterized by clique-width when we are not given a decomposition. We end the paper with Section 4 where we have some concluding remarks and open problems.

2 Framework

We write $V(G)$ and $E(G)$ to denote the set of vertices and edges, respectively, of a graph G. For $A \subseteq V(G)$, let $\overline{A} = V(G) \setminus A$. For a vertex $v \in V(G)$, let $N_G(v)$ be the set of all neighbours of v in G. We omit the subscript if it is clear from the context. For a set $S \subseteq V(G)$ we define $N(S) = \bigcup_{v \in S} N(v) \setminus S$.

2.1 Branch Decompositions

The algorithm of Bui-Xuan et al. [4] needs a branch-decomposition as input. A *branch-decomposition* (T, δ) of a graph G consists of a subcubic tree T (a tree of maximum degree 3) and a bijective function δ from the leaves of T to the vertices of G. (Note that this definition differs from that of [21] by δ mapping to the vertices of G instead of the edges.)

Every edge in a tree splits the tree into two connected components. In a branch decomposition (T, δ) for a graph G, we say that each edge e induces a *cut* in G. This induced cut is a bipartition $(A, \overline{A})$ of the vertices of $V(G)$ so that A is the set of vertices mapped by δ from vertices of one component of $T - e$, and $\overline{A}$ is the set of vertices mapped by δ from the other component of $T - e$.

For a function $f : 2^{V(G)} \rightarrow \mathbb{R}$, we define a *cut-function* f on the set of all cuts $(A, \overline{A})$ such that $f(A, \overline{A}) = \max\{f(A), f(\overline{A})\}$.

Given a cut-function f and a branch-decomposition (T, δ) of a graph G,

- the *f-width* of (T, δ) is the maximum value of f over all the cuts of (T, δ), and
- the *f-width* of G is the minimum f-width over all possible branch-decompositions of G.

When we speak of the f-width of a graph, we address it as a *width parameter* of the graph.

2.2 Neighbourhood Equivalence

Two sets of vertices S_1, S_2 are *neighbourhood equivalent* if they have the same set of neighbours, in other words, $N(S_1) = N(S_2)$. We are particularly interested in neighbourhood equivalence in bipartite graphs, or more specifically, cuts defined by a branch decomposition. This concept was generalized with respect to cuts in [4]. We define the *d-neighbour equivalence* relation $\equiv_d$, and use this to define the parameter nec_d.

For a cut $\left(A, \overline{A}\right)$ of a graph G, and a positive integer d, two subsets $X, Y \subseteq A$ are *d-neighbour equivalent*, $X \equiv_d Y$, over $\left(A, \overline{A}\right)$ if:

$$\text{for each vertex } v \in \overline{A}, \quad \min\{d,\ |N(v) \cap X|\} = \min\{d,\ |N(v) \cap Y|\}.$$

The *number of d-neighbour equivalence classes*, $nec_d(A)$, is the number of equivalence classes of $\equiv_d$ over $\left(A, \overline{A}\right)$.

In other words, $X \equiv_d Y$ over the cut $\big(A, \overline{A}\big)$ if each vertex in $\overline{A}$ is either adjacent to at least d vertices in both X and Y, or is adjacent to exactly the same number of vertices in X as in Y. The algorithm in [4] uses this relation to limit the number of solutions to try. Therefore, the runtime is dependent on the number of d-neighbour equivalence classes.

2.3 Locally Checkable Vertex Subset and Vertex Partitioning Problems

Telle and Proskurowski [23] introduced the *Locally Checkable Vertex Subset and Vertex Partitioning*-problems (LC-VSP), also called $[\sigma, \rho]$-problems and D_q-partition problems. This framework for describing graph problems captures many well known graph problems, see [23,4]. Tables 1 and 2 list some of them. For completeness, we give the definitions of the problem class LC-VSP, however, they are not used directly in this paper and can be skipped by the reader.

For finite or co-finite sets σ and ρ of non-negative integers, a set S of vertices of a graph G is a $[\sigma, \rho]$*-set* of G if for each vertex v of G,

$$|N(v) \cap S| \in \begin{cases} \sigma & \text{if } v \in S, \\ \rho & \text{if } v \in V(G) \setminus S. \end{cases}$$

The *Locally Checkable Vertex Subset-problems* (LC-VS), or $[\sigma, \rho]$*-problems*, are those problems that consist of finding a minimum or maximum $[\sigma, \rho]$-set of the input graph.

The LC-VSP problems, or D_q-partition problems, is a generalization of the LC-VS problems. The goal of these problems is to partition the vertex set of the input graph into multiple $[\sigma, \rho]$-sets. A *degree constraint matrix* D_q is a $q \times q$ matrix such that each cell is a finite or co-finite set of non-negative integers. We say that a partition $V_1, V_2, \ldots, V_q$ of $V(G)$ satisfies D_q if for $1 \leq i, j \leq q$, the number of neighbours in V_j of a vertex of V_i is in the set $D_q[i, j]$. In other words,

$$|N(v) \cap V_j| \in D_q[i, j] \text{ for all } 1 \leq i, j \leq q \text{ and } v \in V_i.$$

For each LC-VSP-problem π, there are two problem-specific constants $d(\pi)$ and $q(\pi)$. The number $q(\pi)$ equals the number of parts in a partition that the problem requests, or equivalently, the row/column size of the constraint matrix. The number $d(\pi)$ is the largest number in all the finite sets and in all the complements of the co-finite sets of the degree constraint matrix used for expressing π. When solving the problem π, we require the use of the d-neighbour equivalence relation $\equiv_{d(\pi)}$.

The algorithm of Bui-Xuan et al. [4] solves each of the LC-VSP problems with a runtime dependent on $nec_{d(\pi)}$-width and $q(\pi)$.

Theorem 1 (Bui-Xuan et al. [4]). *Let π be a problem in LC-VSP. For a graph G given along with a branch-decomposition for that graph of $nec_{d(\pi)}$-width k, the problem π can be solved in time $\mathcal{O}\big(|V(G)|^4 q(\pi) k^{3q(\pi)}\big)$.*

3 The $\mathbb{Q}$-rank-width of a Graph

The $\mathbb{Q}$-*cut-rank* function of a graph G is a function on the subsets of $V(G)$ that maps $X \subseteq V(G)$ to the rank of an $|X| \times |\overline{X}|$-matrix $A = (a_{ij})_{i \in X, j \in \overline{X}}$ over the rational field such that $a_{ij} = 1$ if i and j are adjacent in G and $a_{ij} = 0$ otherwise. We let $\operatorname{cutrk}^{\mathbb{Q}}(X)$ denote the $\mathbb{Q}$-cut-rank of X. For a subset $X \subseteq V(G)$, the matrix A associated with $\operatorname{cutrk}^{\mathbb{Q}}(X)$ is the *adjacency matrix* of the cut $(X, \overline{X})$. Note that if the underlying field of the matrix A is the binary field $GF(2)$, then we obtain the definition of the usual cut-rank function [20]. By $\mathbb{Q}$-*rank-width* of a graph, we mean its $\mathbb{Q}$-cut-rank-width (see subsection 2.1). We may denote the $\mathbb{Q}$-rank-width simply as $\mathrm{rw}_{\mathbb{Q}}$.

Since the $\mathbb{Q}$-cut-rank function is symmetric submodular and is computable in polynomial time, by applying the result of Oum and Seymour [20], we get the following theorem.

Theorem 2 (Oum and Seymour [20]). *There is a $2^{3k}n^{\mathcal{O}(1)}$-time algorithm for which, given a graph G as input and a parameter k, either outputs a branch-decomposition for G of $\mathbb{Q}$-rank-width at most $3k+1$ or confirms that $\mathbb{Q}$-rank-width of G is more than k.*

3.1 Relating $\mathbb{Q}$-rank-width to Other Graph Parameters

The question of how useful the $\mathbb{Q}$-rank-width is as a width parameter is hard to answer. To better understand this question, it would be interesting to know the relation to other well known width parameters such as treewidth, rank-width and clique-width.

The following relates $\mathbb{Q}$-rank-width to the closely related parameter rank-width, yet we see that rank-width can be substantially lower than $\mathbb{Q}$-rank-width.

Lemma 3. *For any graph G we have $\mathrm{rw}(G) \leq \mathrm{rw}_{\mathbb{Q}}(G) \leq \mathrm{cw}(G) \leq 2^{\mathrm{rw}(G)+1} - 1$.*

Proof. The first inequality is from the fact that a set of 0-1 vectors linearly independent over $\mathbb{Q}$ must also be linearly independent over $GF[2]$.

The second and third inequalities follow from [20, Proposition 6.3] since their proof is not dependent on the type of field rank-width uses. They show that a k-expression can be translated to a branch decomposition where for every cut $(A, \overline{A})$ in the decomposition, either the number of unique rows or the number of unique columns in the adjacency matrix M of its induced bipartite graph, is bounded by k. Since this means the rank of M over $\mathbb{Q}$ is at most k, we have $\mathrm{rw}_{\mathbb{Q}}(G) \leq \mathrm{cw}(G)$. The idea of showing $\mathrm{cw}(G) \leq 2^{\mathrm{rw}(G)+1} - 1$, is that a branch decomposition where the adjacency matrix of each cut has its number of unique columns/rows (approximately) bounded by some k, can be translated to a k-expression. As the number of unique columns/rows for any 0-1 matrix of rank rw is at most 2^{rw}, we get our inequality. The last two inequalities are also proved in [14]. □

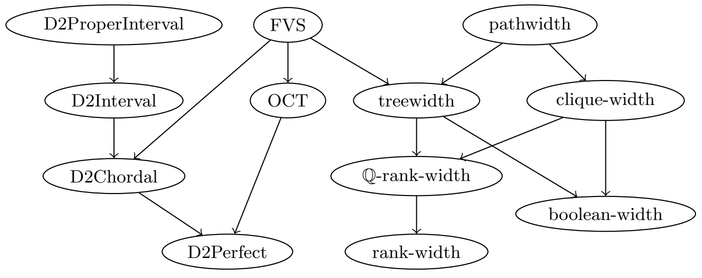

Fig. 1. A comparison diagram of graph parameters relevant to this article. A parameter P is drawn below a parameter Q if for all graphs G we have $P(G) \in \mathcal{O}(Q(G))$. The abbreviations are: FVS = Feedback Vertex Set number, OCT = Odd Cycle Transversal number, D2Π = Vertex Deletion distance to a member of Π.

We believe Lemma 3 is tight. There are existing results showing that it is almost tight. A $n \times n$ grid has rank-width $n-1$ [13] and clique-width $n+1$ [11], hence the first two inequalities are almost tight. There exist graphs with treewidth k and hence $\mathbb{Q}$-rank-width $\leq k$ and clique-width $2^{\lfloor k/2 \rfloor - 1}$ [5].

From Vatshelle [25, Section 4.2.1] we deduce the following observation, which will help us bound $\mathbb{Q}$-rank-width in terms of treewidth.

Observation 4 ([25]). *A graph of treewidth k has a branch decomposition where each cut has a vertex cover of size at most $k+1$.*

Lemma 5. *The $\mathbb{Q}$-rank-width of a graph is never more than the treewidth $+1$.*

Proof. If a graph G has a treewidth of k, then by Observation 4, there exists a branch decomposition of G where every cut has a vertex cover of at most $k+1$ vertices. Suppose we have a cut $\left(X, \overline{X}\right)$ with such a vertex cover S. Since the adjacency matrix of the cut decreases its rank by at most one by the removal of one vertex, and the adjacency matrix of $\left(X \setminus S,\ \overline{X} \setminus S\right)$ has rank zero (since the matrix consist of only 0's), the cut $\left(X, \overline{X}\right)$ must have $\mathbb{Q}$-cut-rank at most $k+1$. This holds for all cuts, and thus $\mathbb{Q}$-rank-width is at most the treewidth $+1$. $\square$

Figure 1 shows a comparison diagram of graph parameters. The idea of such a diagram is that parameterized complexity results will propagate up and down in this diagram. Positive results propagate upward; for instance, since DOMINATING SET is solvable in $2^{\mathcal{O}(tw)} n^{\mathcal{O}(1)}$ for a graph of treewidth tw [22], we see that DOMINATING SET is solvable in $2^{\mathcal{O}(pw)} n^{\mathcal{O}(1)}$ for a graph of pathwidth pw. Negative results propagate downward; for example, since unless ETH fails, DOMINATING SET can not be solved in $2^{o(pw)} n^{\mathcal{O}(1)}$ where pw is the pathwidth of the input graph[16], so is the case for treewidth, clique-width, $\mathbb{Q}$-rank-width, rank-width and boolean-width.

3.2 Relating nec_d-width to $\mathbb{Q}$-rank-width

Now we know how to find a branch-decomposition with a low $\mathbb{Q}$-rank-width, but we have yet to show what this means in terms of runtime for the algorithm in [4] (i.e. prove Theorem 1).

We know the runtime of the algorithm in terms of the nec_d-width of the given decomposition. So, if we manage to give a bound on the nec_d-width of a decomposition in terms of the $\mathbb{Q}$-rank-width, we will also get a bound on the runtime of the algorithm in terms of $\mathbb{Q}$-rank-width. We will prove such a bound shortly, but in order to do this we first need the following lemma, based on a proof of Belmonte and Vatshelle [1, Lemma 1].

Lemma 6. *Given a positive integer d and a cut $(A, \overline{A})$ of $\mathbb{Q}$-cut-rank k, for every subset $S \subseteq A$, there exists a subset $R \subseteq S$ so that $|R| \leq dk$ and $R \equiv_d S$ over the cut.*

Proof. We will give a proof by induction. First we show that the lemma holds for $d = 1$. Let S' be a minimal subset of S so that $S' \equiv_1 S$. Since S' is minimal, removing any vertex of S' will decrease $|N(S')|$. Therefore, every vertex of S' is adjacent to at least one vertex that none of the other vertices in S' are adjacent to. In the adjacency matrix M of $(A, \overline{A})$, this means that each of the corresponding rows of S' has a 1 in a column where all the other rows of S' has a 0. Hence, the rows are all linearly independent. As the $\mathbb{Q}$-cut-rank of $(A, \overline{A})$ is the maximum number of linearly independent rows of M, we can deduce that $|S'| \leq \text{cutrk}^{\mathbb{Q}}(A) \leq k$.

For the induction step, we show that if the lemma holds for all values up to d, then it must also hold for $d+1$. By the above, we know that there exists a subset $S' \subseteq S$ of cardinality at most k such that $S' \equiv_1 S$. By the induction hypothesis, there exists a set $R' \subseteq (S \setminus S')$ so that $R' \equiv_d (S \setminus S')$ and $|R'| \leq d \cdot k$. We will show that the set $R = S' \cup R'$, which is of size $\leq k(d+1)$, satisfies $R \equiv_{d+1} S$.

Let S^* be the set $S \setminus R$. Since $R' \subseteq (S \setminus S')$, for all vertices $v \in N(S^*)$ we have $|N(v) \cap (S \setminus S')| > |N(v) \cap R'|$. However, as $R' \equiv_d (S \setminus S')$, every vertex in $N(S^*)$ must be adjacent to at least d vertices of R'. And since $N(S') = N(S)$, every vertex in $N(S^*)$ must also be adjacent to at least one vertex of S'. Thus, for every vertex $v \in \overline{A}$, either $v \in N(S^*)$ and therefore $|N(v) \cap S| > |N(v) \cap R| \geq d + 1$, or $v \notin N(S^*)$ and so $N(v) \cap S = N(v) \cap R$. □

Because of Lemma 6, in order to count distinct d-neighbour equivalence classes for a cut, we need only count those neighbourhoods that are produced by sets of size no more than d times $\text{rw}_{\mathbb{Q}}$.

So far, the results of $\mathbb{Q}$-rank-width have been similar to those of rank-width. For both rank-width, $\mathbb{Q}$-rank-width and boolean-width it can be shown that for a parameter value of k, the number of vertices with distinct neighbourhood over the cut is no more than 2^k, and each d-neighbourhood can be represented by at most dk vertices [25]. Putting this together gives a trivial bound of $nec_d \leq 2^{dk^2}$. However, when counting the number of d-neighbour equivalence classes for a cut, we can improve this trivial bound when using the $\mathbb{Q}$-rank-width. This is because the row space of a $\mathbb{Q}$-basis of some matrix not only contains all the rows of the matrix, but also all the different sums of the rows in the matrix. So, we can bound nec_d by a more direct connection between $\mathbb{Q}$-rank-width and the number of d-neighbourhoods than that of the trivial bound.

Theorem 7. *If the $\mathbb{Q}$-rank-width of a branch-decomposition is k, then the nec_d-width of the same decomposition is no more than $2^{k \log{(dk+1)}}$.*

Proof. To prove this we only need to show that the inequivalence holds for any cut of a graph: Suppose we have a cut $\left(A, \overline{A}\right)$ of $\mathbb{Q}$-cut-rank k and adjacency matrix M. Since the rank of M is k, there are exactly k linearly independent columns $C = \{c_1, c_2, c_3, \dots, c_k\}$ in M. This means that every row in the row space of M can be identified by the k entries in the columns of C. Also, since M is an adjacency matrix, its entries are either 0 or 1.

Let $r(S)$ be the sum of row vectors of M corresponding to $S \subseteq A$. If $r(S) = r(S')$ then $S \equiv_d S'$ (for any d), since the entries of $r(S)$ and $r(S')$ state exactly how many vertices of the set are adjacent to each of the vertices of $\overline{A}$.

From Lemma 6, to compute nec_d we only need to count d-neighbour equivalence classes for sets of size at most dk. Therefore, to calculate nec_d, we will instead calculate an upper bound to the number of distinct rows that represent some neighbourhood of a set of size $\leq dk$.

For a set $S \subseteq A$, each vertex $v \in S$ contributes to each entry of $r(S)$ either a zero or a one. That means if $|S| \leq dk$, then the entries of $r(S)$ take on values in the integer range $[0, dk]$. As $r(S)$ is a linear combination of the rows of M, it is in the row space of M. That means $r(S)$ can be uniquely identified by the values of the k entries in the columns of C. Therefore, the number of distinct d-neighbour equivalence classes of the cut is no more than the number of ways to choose one out of $dk + 1$ values for k entries. In other words, $nec_d \leq ((dk) + 1)^k = 2^{k \log{(dk+1)}}$. This proves the theorem. $\square$

This result, combined with Theorems 1 and 2, shows that all the LC-VSP problems can be solved in $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$-time. Expressing the runtime in terms of clique-width, we get the following teorem.

Theorem 8. *Every LC-VSP problem π can be solved in $2^{\mathcal{O}(\mathrm{cw} \log{(\mathrm{cw} \cdot d(\pi)) q(\pi)})} n^{\mathcal{O}(1)}$-time on graphs of clique-width* cw.

Proof. By Theorem 2 we can create a branch-decomposition of $\mathbb{Q}$-rank-width at most $3 \,\mathrm{rw}_{\mathbb{Q}} + 1$ in $2^{3 \,\mathrm{rw}_{\mathbb{Q}}} n^{\mathcal{O}(1)}$-time. By Theorems 1 and 7, every LC-VSP problem can be solved in $2^{9 \,\mathrm{rw}_{\mathbb{Q}} \log{(3 \,\mathrm{rw}_{\mathbb{Q}} \cdot d(\pi) + 1) q(\pi)}} n^{\mathcal{O}(1)}$. Since $\mathrm{rw}_{\mathbb{Q}} \leq \mathrm{cw}$ by Lemma 3, we have the same runtime parameterized by cw instead of $\mathrm{rw}_{\mathbb{Q}}$. $\square$

4 Conclusion

If we are given a k-expression as input, the best known FPT algorithm parameterized by k solving the Dominating Set is by Bodlaender et al. [2] and runs in time $4^k n^{\mathcal{O}(1)}$. However, it is currently open whether we can construct a k-expression of an input graph of clique-width at most $\mathcal{O}(k)$ in polynomial time. We have shown the existence of algorithms with runtime $2^{\mathcal{O}(\text{cw} \log \text{cw})} n^{\mathcal{O}(1)}$ for all LC-VSP problems, without assuming that a k-expression is given as an input. This still leaves the natural open question:

Open Problem 1. *Can* Independent Set *or* Dominating Set *be solved in* $2^{\mathcal{O}(\text{cw})} n^{\mathcal{O}(1)}$ *time, where* cw *is the clique-width of the graph?*

We know that for a graph of treewidth tw, Independent Set can be solved in $2^{\mathcal{O}(tw)} n^{\mathcal{O}(1)}$ time leading to an interesting question of what parameters give a linear exponential runtime for Independent Set. Two such parameters are the Vertex Deletion Distance to Proper Interval graphs (D2PI) and the Odd Cycle Transversal number (OCT number):

1. For a graph G, the D2PI of a graph is the minimum number of vertices needed to be removed in order to make G into a Proper Interval graph. For a graph G with D2PI equal k, Villanger and van't Hof [24] gave a $6^k n^{\mathcal{O}(1)}$ algorithm for finding such a set S to be removed. To solve Independent Set on a graph $G = (V, E)$, we guess the intersection $S' \subseteq S$ between the optimal solution and S, and then combining it with the optimal solution of Independent Set on the proper interval graph $G[V \setminus (S \cup N[S'])]$.
2. The OCT number of a graph G is the minimum number of vertices needed to remove from G in order to make it bipartite. For a graph G with OCT number equal k, Lokshtanov, Saurabh and Sikdar [17] gave a $(n^{\mathcal{O}(1)} 3^k)$ algorihm for finding the minimum sized set S of vertices to remove from G to make it bipartite. As with the algorithm above, we can solve Independent Set by guessing the intersection $S' \subseteq S$ between the optimal solution and S and combine it with the optimal solution of the bipartite graph $G[V \setminus (S \cup N[S'])]$. As Independent Set is solvable in $n^{\mathcal{O}(1)}$ time on bipartite graphs [9], this yields a $2^k n^{\mathcal{O}(1)}$ time algorithm.

Note, however, that these parameters are not bounded by treewidth (or clique-width), see Figure 1.

References

1. R. Belmonte and M. Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theoret. Comput. Sci.*, 2013. doi:10.1016/j.tcs.2013.01.011.
2. H. Bodlaender, E. van Leeuwen, J. van Rooij, and M. Vatshelle. Faster algorithms on clique and branch decompositions. In *Proceedings of MFCS*, volume 6281 of *LNCS*, pages 174–185. Springer, 2010.
3. B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. H-join decomposable graphs and algorithms with runtime single exponential in rankwidth. *Discrete Appl. Math.*, 158(7):809–819, 2010.

4. B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theoret. Comput. Sci.*, 2013. doi: 10.1016/j.tcs.2013.01.009.
5. D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847 (electronic), 2005.
6. B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.
7. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101(1-3):77–114, 2000.
8. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer-Verlag, New York, 1999.
9. U. Faigle and G. Frahling. A combinatorial algorithm for weighted stable sets in bipartite graphs. *Discrete Appl. Math.*, 154(9):1380–1391, 2006.
10. J. Flum and M. Grohe. *Parameterized complexity theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.
11. M. C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *Internat. J. Found. Comput. Sci.*, 11(3):423–443, 2000.
12. F. Gurski. A comparison of two approaches for polynomial time algorithms computing basic graph parameters. *CoRR*, abs/0806.4073, 2008.
13. V. Jelínek. The rank-width of the square grid. *Discrete Appl. Math.*, 158(7):841–850, 2010.
14. M. M. Kanté and M. Rao. The rank-width of edge-coloured graphs. *Theory Comput. Syst.*, 52(4):599–644, 2013.
15. D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 777–789, Philadelphia, PA, 2011. SIAM.
16. D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
17. D. Lokshtanov, S. Saurabh, and S. Sikdar. Simpler parameterized algorithm for OCT. In *Combinatorial algorithms*, volume 5874 of *Lecture Notes in Comput. Sci.*, pages 380–384. Springer, Berlin, 2009.
18. R. Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications.* Oxford University Press, Oxford, 2006.
19. S. Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):Art. 10, 20, 2008.
20. S. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.
21. N. Robertson and P. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.
22. J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 610–621. Springer-Verlag, 1993.
23. J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997.
24. P. van 't Hof and Y. Villanger. Proper interval vertex deletion. *Algorithmica*, 65(4):845–867, 2013.
25. M. Vatshelle. *New width parameters of graphs.* PhD thesis, University of Bergen, May 2012. `http://hdl.handle.net/1956/6166`.