Breaking the 2^n-Barrier for Irredundance: Two Lines of Attack ☆

Daniel Binkele-Raible[a], Ljiljana Brankovic[b,1], Marek Cygan[c], Henning Fernau[a], Joachim Kneis[d,2], Dieter Kratsch[e,3], Alexander Langer[d], Mathieu Liedloff[f,3], Marcin Pilipczuk[c], Peter Rossmanith[d], Jakub Onufry Wojtaszczyk[g]

[a] *FB 4 – Abteilung Informatik, Universität Trier, Germany*
{fernau,raible}@informatik.uni-trier.de
[b] *School of Electrical Engineering and Computer Science, The University of Newcastle, Australia*
ljiljana.brankovic@newcastle.edu.au
[c] *Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland*
{cygan,malcin}@mimuw.edu.pl
[d] *Dept. of Computer Science, RWTH Aachen University, Germany*
{kneis,langer,rossmani}@cs.rwth-aachen.de
[e] *Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine Metz, France*
kratsch@univ-metz.fr
[f] *Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, France*
liedloff@univ-orleans.fr
[g] *Institute of Mathematics, Polish Academy of Sciences, Poland*
onufry@mimuw.edu.pl

Abstract

The lower and the upper irredundance numbers of a graph G, denoted $\mathrm{ir}(G)$ and $\mathrm{IR}(G)$, respectively, are conceptually linked to the domination and independence numbers and have numerous relations to other graph parameters. It has been an open question whether determining these numbers for a graph G on n vertices admits exact algorithms running in time faster than the trivial $\Theta(2^n \cdot \mathrm{poly}(n))$ enumeration, also called the 2^n-barrier.

The main contributions of this article are exact exponential-time algorithms breaking the 2^n-barrier for irredundance. We establish algorithms with running times of $O^*(1.99914^n)$ for computing $\mathrm{ir}(G)$ and $O^*(1.9369^n)$ for computing $\mathrm{IR}(G)$. Both algorithms use polynomial space. The first algorithm uses a parameterized approach to obtain (faster) exact algorithms. The second one is based, in addition, on a reduction to the Maximum Induced Matching problem providing a branch-and-reduce algorithm to solve it.

Keywords: Graph algorithms, Irredundance number

☆ This paper is based on two preliminary ones that appear in the Proceedings of the Seventh International Conference on Algorithms and Complexity (CIAC 2010, Rome, Italy) [1, 2].

[1] Supported by the RGC CEF grant G0189479 of The University of Newcastle.
[2] Supported by the DFG under grant RO 927.
[3] Supported by the grant ANR blanc AGAPE.

2010 MSC: 05C85 Graph algorithms,
2010 MSC: 68R05 Combinatorics,
2010 MSC: 68Q17 Computational difficulty of problems,
2010 MSC: 68Q25 Analysis of algorithms and problem complexity

---

1. Introduction

The field of exact exponential-time algorithms for NP-hard problems has attracted a lot of attention in recent years. Many hard problems can be solved much faster than they could be solved by obvious brute-force algorithms; examples include the well-known graph problems Maximum Independent Set [3] and Minimum Dominating Set [3, 4]. See Woeginger's survey [5] from 2003 or the recent monograph [6] for an overview. We will add to this list the NP-hard problems asking to compute the well-known graph-theoretic parameters, the lower and the upper irredundance number.

A set $I \subseteq V$ is called an *irredundant set* of a graph $G = (V, E)$ if each $v \in I$ is either isolated in $G[I]$, or there is at least one vertex $u \in V \setminus I$ with $N(u) \cap I = \{v\}$, called a *private neighbor* of v. An irredundant set I is *(inclusion-wise) maximal* if no proper superset of I is an irredundant set. The *lower irredundance number* $\mathrm{ir}(G)$ equals the minimum size taken over all maximal irredundant sets of G. Similarly, the *upper irredundance number* $\mathrm{IR}(G)$ equals the maximum size taken over all maximal irredundant sets of G.

In graph theory, the irredundance numbers have been extensively studied due to their relation to numerous other graph parameters. An estimated 100 research papers [7] have been published on the properties of irredundant sets in graphs, e.g., [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. A lot of relations to other established graph parameters are known. For instance, consider the classical graph parameters $\alpha(G)$ and $\gamma(G)$ defined as follows. A set $I \subseteq V$ is an *independent set* of G, if and only if no vertex in I is adjacent to any other vertex in I. If $I \subseteq V$ is a maximum size independent set of G, then $\alpha(G) = |I|$ denotes the *independence number* of G. A set $D \subseteq V$ is called a *dominating set* of a graph $G = (V, E)$, if and only if each vertex either belongs to D or is adjacent to a vertex in D. If $D \subseteq V$ is a minimum size dominating set of G, then $\gamma(G) = |D|$ denotes the *domination number* of G.

Now, if $D \subseteq V$ is an (inclusion-wise) minimal dominating set, i.e. no proper subset of D is dominating, then for every $v \in D$, there is some minimality witness, i.e., a vertex that is only dominated by v. In fact, a set is minimal dominating if and only if it is irredundant and dominating [19]. Since each independent set is also an irredundant set, the well-known *domination chain*

$$\mathrm{ir}(G) \leq \gamma(G) \leq \alpha(G) \leq \mathrm{IR}(G)$$

is a simple observation. It is also known [8, 9] that

$$\gamma(G)/2 < \mathrm{ir}(G) \leq \gamma(G) \leq 2 \cdot \mathrm{ir}(G) - 1.$$

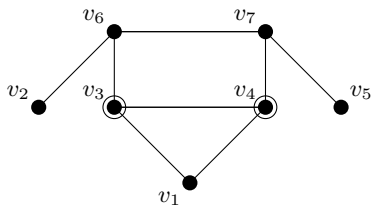

Figure 1: A minimum maximal irredundant set is $X = \{v_3, v_4\}$. Note that X is not a dominating set.

Note that a maximal irredundant set does not necessarily have to dominate the whole vertex set of G, see Figure 1.

There are also some applications of irredundant sets in combinatorial optimization, e.g., locating senders in broadcast and packet radio networks [20]. We mention that determining the lower and upper irredundance numbers are NP-hard problems [16, 17]. Computing $\mathrm{ir}(G)$ remains NP-hard on bipartite graphs [17], while here $\mathrm{IR}(G)$ can be computed in polynomial time using the equality $\mathrm{IR}(G) = \alpha(G)$ [21].

A few powerful techniques have been developed for the design and worst-case analysis of exact algorithms for combinatorially hard problems like independence, domination and coloring; among them dynamic programming, branching algorithms with a Measure & Conquer analysis [3] and inclusion-exclusion algorithms [22, 23, 24]. However, there is still a couple of problems for which no faster exact algorithm than trivial enumeration is known.

For example, the fastest hitherto known exact algorithm to compute $\mathrm{IR}(G)$ and $\mathrm{ir}(G)$ is the simple $O^*(2^n)$ brute-force approach enumerating all vertex subsets and verifying for each one whether it is irredundant.[4] The aforementioned techniques did not seem sufficient to break the 2^n-barrier for irredundance problems, at least not in an obvious fashion; so breaking the 2^n-barrier was publicly posed as an open question by van Rooij at a Dagstuhl seminar in 2008 (see [25]).

New ideas were needed to attack these problems. In fact, two research groups independently managed to affirmatively solve this puzzle, proposing two different methodologies. One line of attack was based on looking at parameterized algorithms to be used for establishing exact algorithms, and the other line of attack was based on structural insights and graph-theoretic reformulations leading to a (direct) construction of exact algorithms [1, 2]. In this work, we present both approaches and also show that a combination of the basic ideas from both lines of attack yields even better results.

[4]The O^*-notation suppresses polynomial factors. Hence $f(n)\mathrm{poly}(n) = O^*(f(n))$.

1.1. Our Contributions

We will deal with the lower irredundant set and the upper irredundant set problems both from the perspective of exact and from the perspective of parameterized algorithms. We will present the derived results in a sequence that shows the easiest results first and then presents more and more elaborate approaches.

1. Upper bounding the number of irredundant sets of size at most t, for all $t \in \{1, 2, \ldots, n\}$, results in a $O^*(1.999956^n)$ enumeration (exact) algorithm to compute the lower irredundance number $\mathrm{ir}(G)$.
2. We prove that deciding whether $\mathrm{ir}(G) \leq k$ for a given parameter k is W[2]-hard as opposed to the known W[1]-completeness of $\mathrm{IR}(G)$ [26]. Then we study the dual parameterizations (with parameter $n - k$) and present linear size problem kernels. The parameterized problem to decide whether $\mathrm{ir}(G) \leq n-k$ allows a kernel with $2k-1$ vertices, deciding whether $\mathrm{IR}(G) \geq n-k$ has a kernel with $3k$ vertices. This already shows that both problems can be solved with a running time of $O(c^k \mathrm{poly}(n))$, $c \leq 8$. In particular, this improves the kernel with $3k^2$ vertices and the corresponding running time of $O(8^{k^2}\mathrm{poly}(n))$ of [26], established for the question "$\mathrm{IR}(G) \geq n - k$?".
3. We design a simple parameterized algorithm with a running time bounded by $O(3.841^k \mathrm{poly}(n))$ which solves both problems simultaneously. The price we pay for this generality is that the running time is only slightly better than $O(4^k \mathrm{poly}(n))$, since we cannot exploit any special properties of $\mathrm{IR}(G)$ that do not hold for $\mathrm{ir}(G)$, and vice versa. Nevertheless, this can already be used to achieve exact algorithms with a running time $O^*(c^n)$ for some $c < 2$, i.e. to break the 2^n-barrier for both problems.
4. We provide an exact algorithm computing $\mathrm{IR}(G)$ in $O^*(1.9369^n)$ time and polynomial space. We use a reduction to the maximum induced matching problem on bipartite graphs and develop a branching algorithm for it as in [2], but we analyze the algorithm using the parameterized approach. The resulting exact algorithm is both faster and simpler than the algorithms of [1, 2].

2. Preliminaries

We use standard notation from graph theory. For example, $G[I]$ denotes the subgraph induced by the vertex set I. For a vertex v, we use $N(v)$ and $N[v] = N(v) \cup \{v\}$ for the open and closed neighborhoods of v and let $\deg(v) := |N(v)|$ be the degree of v. We extend this notation to any subset $W \subseteq V$ by letting $N[W] := \bigcup_{v \in W} N[v]$. We say that a set W *dominates* a vertex u if $u \in N[W]$. For a graph $G = (V, E)$ and a subset of edges $E_0 \subseteq E$ by $V(E_0)$ we denote the set of endpoints of edges in the set E_0.

The algorithms in this paper are exact exponential-time and parameterized algorithms for computing the lower and the upper irredundance number. Let us briefly define some essentials of these areas.

When studying exact algorithms for NP-hard problems one is interested in the worst-case running time, which is often, and also in this paper, of the type $O^*(c^n)$, where n is the number of vertices of the input graph and the goal is to decrease the value of c. As already pointed out our two problems can be solved in time $\Theta^*(2^n)$ by trivial enumeration, and thus we are interested in $c < 2$ only. A major technique used in exact algorithms (and also in parameterized algorithms) are branching algorithms. The *Measure & Conquer* analysis for the worst-case running time of exact branching algorithms was introduced by Fomin et al. [3]. A fundamental idea of this approach is to use a *measure* for the inputs to the recursive calls (subproblems) and to upper bound the size of the corresponding search tree in terms of this measure. Typically this requires to solve a (large number of) linear recurrences and to compute a measure as to achieve the best bound on the running time. For more information on exact algorithms for NP-hard problems we refer to the survey of Woeginger [5] or the monograph [6].

A problem with inputs of the form (x, k) with k being termed the *parameter* is called *fixed-parameter tractable* if it can be solved in time $O(p(|x|) \cdot f(k))$ for some polynomial function p and some arbitrary function f. It can be argued that in practice, for small values of k, such running times can be as good as polynomial time. Problems admitting such a running time are collected in the complexity class FPT. It can be shown that a problem is in FPT if and only if it has a *problem kernel*, a polynomial-time computable self-reduction of each instance (x, k) to (x', k'), where $|x|, k' \leq g(k)$ for some function g. Analogously to the P vs. NP question, there is also a hardness theory for parameterized algorithms, leading to the W-hierarchy, in which the classes W[1] and W[2] constitute the lower levels. More details can be found in monographs like [27, 28, 29].

In this paper, we use parameterized algorithms to establish exact algorithms. To do this we will often use one of the following "win-win"-approaches: Say $|V| = n$ and we are looking for a subset $I \subseteq V$ with certain properties, e.g., such that I is an irredundant set. We can then distinguish two cases:

1. $|I|$ is "small", i.e., $|I| < (1/2 - \epsilon)n$ for a suitable $\epsilon > 0$. In this case, even brute-force enumeration of all $\sum_{i=0}^{(1/2-\epsilon)n} \binom{n}{i} < 2^n$ subsets of V is sufficiently fast.
2. $|I| \geq (1/2 - \epsilon)n$, which yields some structural properties that can be exploited by an algorithm.

Similarly, it has been known for a while (see, e.g., [30]) that it is possible to break the 2^n-barrier for some vertex selection problems by designing parameterized algorithms that run in time $O(c^k \text{poly}(n))$ for some $c < 4$. Here, we can use the parameterized algorithm as follows: If the parameter is "small", i.e., $k < \epsilon n$ for an appropriate $\epsilon > 0$, we use the parameterized algorithm to solve the problem in time $O^*(c^{\epsilon n})$ with $c^\epsilon < 2$. Otherwise, we use standard subset enumeration as above.

The following alternative definition of *irredundance* is more descriptive and eases understanding of some of the algorithms in this paper: The vertices in

an irredundant set can be thought of as kings, where each such king ought to have his very own private garden that no other king can see (where "seeing" means adjacency). Each king has exactly one cultivated private garden, and all additional (possible) private gardens degenerate to wilderness. It is also possible that the garden is already built into the king's own castle. One can easily verify that this alternate definition is equivalent to the formal one given above.

Definition 1. Let $G = (V, E)$ be a graph and $I \subseteq V$ an irredundant set. We call the vertices in I *kings*. Private neighbors are called *gardens*, and all remaining vertices are *wilderness*. If a king has more than one private neighbor, we designate one of these vertices as the king's unique private garden and the other private neighbors as wilderness. If a vertex $v \in I$ has no neighbors in I, we (w.l.o.g.) say v has an *internal* garden, otherwise its garden is *external*. We denote the corresponding sets as $\mathcal{K}, \mathcal{G}, \mathcal{W}$ ($\mathcal{K}$ and $\mathcal{G}$ are not necessarily disjoint). Kings with external gardens are denoted by $\mathcal{K}_e$ and kings with internal garden by $\mathcal{K}_i$. Similarly, the set of external gardens is $\mathcal{G}_e := \mathcal{G} \setminus \mathcal{K}$.

In what follows these sets are also referred to as *labels*. We say that I *respects* a labeling $(\mathcal{K}_i', \mathcal{K}_e', \mathcal{G}_e', \mathcal{W}')$ if $\mathcal{K}_i' \subseteq \mathcal{K}_i$, $\mathcal{K}_e' \subseteq \mathcal{K}_e$, $\mathcal{G}_e' \subseteq \mathcal{G}_e$, and $\mathcal{W}' \subseteq \mathcal{W}$.

The exact algorithms of the paper solve the NP-hard irredundance problems "Given a graph G, compute $\mathrm{IR}(G)$" and "Given a graph G, compute $\mathrm{ir}(G)$". To this extent various related parameterized problems are studied in this paper, formally defined as follows.

Maximum Irredundant Set (MaxIR)

Input: An undirected graph $G = (V, E)$
Parameter: k
Question: Is $\mathrm{IR}(G) \geq k$?

Minimum Maximal Irredundant Set (MinMaxIR)

Input: An undirected graph $G = (V, E)$
Parameter: k
Question: Is $\mathrm{ir}(G) \leq k$?

For the parameterized approach, we also study their dual parameterizations defined as follows [26].

Co-Maximum Irredundant Set (Co-MaxIR)

Input: An undirected graph $G = (V, E)$, a positive integer k
Parameter: k
Question: Is $\mathrm{IR}(G) \geq n - k$?

Co-Minimum Maximal Irredundant Set (Co-MinMaxIR)

Input: An undirected graph $G = (V, E)$, a positive integer k
Parameter: k
Question: Is $\mathrm{ir}(G) \leq n - k$?

Equal Co-Minimum Maximal Irredundant Set (Equal Co-MinMaxIR)

Input: An undirected graph $G = (V, E)$, a positive integer k

Algorithm 1 An iterative-DFS algorithm to compute $\mathrm{ir}(G)$.

Algorithm MinMaxIR(G):
Input: Graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_n\}$.
Output: The size of the minimum maximal irredundant set.

```
01: procedure Depth-First-Search(G, k, I, i):
        Searches for a maximal irredundant set I' with |I'| ≤ k
        and I' ∩ {v_1, v_2, ..., v_{i-1}} = I.
02:     if i = n + 1 then
03:         if I is a maximal irredundant set then return True
04:         else return False
05:     if |I| < k and I ∪ {v_i} is an irredundant set then
06:         if Depth-First-Search(G, k, I ∪ {v_i}, i + 1) then
07:             return True
08:     return Depth-First-Search(G, k, I, i + 1)

08: procedure MinMaxIR(G):
09:     for k := 0 to n do
10:         if Depth-First-Search(G, k, ∅, 1) then return k
```

Parameter: k
Question: Is $\mathrm{ir}(G) = n - k$?

3. Computing ir(G)

In this section we design an exact algorithm to compute $\mathrm{ir}(G)$. We present a simple iterative-DFS algorithm that requires polynomial space and we prove that it works in $O^*(1.999956^n)$ time. W.l.o.g., we may assume that G contains no isolated vertices, since they need to be included in any maximal irredundant set.

The algorithm is inspired by Björklund et al. [31]. Let $\mathcal{F}_k$ be the family of irredundant sets in G of size at most k. Note that checking if a set is a (maximal) irredundant set can be done in polynomial time. Moreover, the family of irredundant sets is closed under taking subsets. Therefore, $\mathcal{F}_k$ can be enumerated in $O(|\mathcal{F}_k|\mathrm{poly}(n))$ time and space polynomial in n by a simple depth-first search algorithm, sketched as procedure Depth-First-Search in Algorithm 1.

Let us now quickly analyze the procedure Depth-First-Search. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a fixed ordering of the vertex set. The procedure Depth-First-Search, invoked with parameters (G, k, I, i) checks if there exists a maximal irredundant set I' of size at most k such that $I' \cap \{v_1, v_2, \ldots, v_{i-1}\} = I$. As the family of irredundant sets is closed under taking subsets, the procedure keeps the invariant that I is an irredundant set of size at most k and $I \subset \{v_1, v_2, \ldots, v_{i-1}\}$. Thus, to check if there exists a maximal irredundant set of size at most k we invoke the procedure with parameters $(G, k, \emptyset, 1)$.

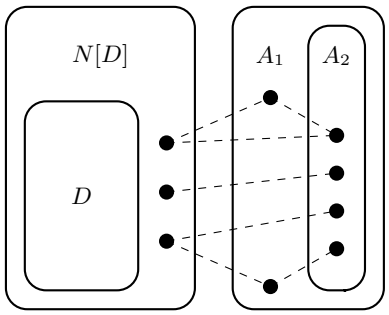

Figure 2: Notation in the proof of Lemma 1

As the stack of the procedure Depth-First-Search is of polynomial size, the procedure uses polynomial space. We now prove that the call Depth-First-Search$(G, k, \emptyset, 1)$ works in $O(|\mathcal{F}_k|\text{poly}(n))$ time. To see this note that for each irredundant set I of size at most k and for each $1 \leq i \leq n+1$, there will be at most one call to Depth-First-Search(G, k, I, i). Indeed, if $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_s}\}$ and $i_1 < i_2 < \ldots < i_s$, then Depth-First-Search(G, k, I, i) is called only from Depth-First-Search$(G, k, I, i-1)$ if $i > i_s$ and from Depth-First-Search$(G, k, I \setminus \{v_{i_s}\}, i-1)$ if $i = i_s$. Thus, for each $I \in \mathcal{F}_k$, there are at most $n+1$ calls to the Depth-First-Search procedure and the time bound follows.

The simple iterative-DFS algorithm (Algorithm 1) enumerates $\mathcal{F}_k$ for $k = 0, 1, 2, \ldots, n$ until it finds a maximal irredundant set, thus finding a minimum one. Now we prove that it works in $O^*(1.999956^n)$ time.

We first present a $O^*((2-\varepsilon_\Delta)^n)$ time bound for graphs with maximum degree bounded by Δ, where ε_Δ depends on Δ. Construct a set $A \subseteq V$ greedily: repeatedly add any vertex $v \in V$ to A and remove from V all vertices at distance at most 2 from v. At each step, at most $1 + \Delta + \Delta(\Delta-1) = 1 + \Delta^2$ vertices are removed, therefore $|A| \geq n/(1+\Delta^2)$. The set A is an independent set; moreover, closed neighborhoods $\{ N[v] : v \in A \}$ are disjoint. If S is an irredundant set, then $S \cap N[v] \neq N[v]$ and, therefore, for each $v \in A$ we have at most $2^{|N[v]|} - 1$ possibilities to choose $S \cap N[v]$ instead of $2^{|N[v]|}$. As these neighborhoods are disjoint, this leads to the following bound on the number of irredundant sets:

$$|\mathcal{F}_n| \leq 2^n \prod_{v \in A} \frac{2^{|N[v]|} - 1}{2^{|N[v]|}} \leq 2^n \left(\frac{2^{\Delta+1} - 1}{2^{\Delta+1}} \right)^{\frac{n}{1+\Delta^2}} = (2 - \varepsilon_\Delta)^n,$$

and the time bound for the algorithm follows. Note that in this case the time bound is $O^*((2-\varepsilon_\Delta)^n)$ even if we invoke Depth-First-Search only once, for $k = n$.

Now we show how to bypass the maximum degree assumption. Note that if an irredundant set is a dominating set, it is maximal irredundant. Moreover a minimal dominating set is also a maximal irredundant set. Assume

that G admits a dominating set of size at most $149n/300$. Then the algorithm stops before or at the step $k = 149n/300$ and up to this point consumes $O^*(\binom{n}{149n/300}) = O^*(1.999956^n)$ time. Therefore we may consider only the case where every dominating set in G is of size greater than $149n/300$.

The following structural lemma is crucial for the analysis.

Lemma 1. *Let $G = (V, E)$ be a graph with n vertices that contains no dominating set of size smaller than $149n/300$. Then there exists a set $A \subseteq V$ satisfying:*

1. *A is an independent set and the neighborhoods $\{ N[v] : v \in A \}$ are disjoint,*
2. *every vertex in A has degree at most 6,*
3. *$|A| \geq 41n/9800$.*

Proof. We construct a dominating set D greedily. Start with $D = \emptyset$. In a single step, take any vertex v that adds at least 3 new vertices to $N[D]$, i.e., $|N[D \cup \{v\}] \setminus N[D]| \geq 3$, and add v to D. This algorithm stops at some point. Let $A_1 = V \setminus N[D]$, i.e., the vertices not dominated by D. For every vertex v, we have $|N[v] \cap A_1| \leq 2$, since D cannot be extended any more. In particular, every vertex in $G[A_1]$ has degree at most 1, so $G[A_1]$ is a graph of isolated vertices and isolated edges. Let A_2 be any maximal independent set in $G[A_1]$, i.e., A_2 contains all isolated vertices of $G[A_1]$ and one endpoint of every isolated edge. The set A_2 is an independent set in G, too. The notation is presented in Figure 2.

Note that $D \cup A_2$ is a dominating set in G, since A_2 dominates A_1. Therefore $|D| + |A_2| \geq 149n/300$. By the construction procedure of D, we have $|D| \leq \frac{1}{3}|N[D]| = \frac{1}{3}|V \setminus A_1|$, and $|A_2| \leq |A_1|$, so:

$$149/300 \leq \frac{|D| + |A_2|}{|V|} \leq \frac{1}{3} - \frac{|A_1|}{3|V|} + \frac{|A_2|}{|V|} \leq \frac{1}{3} + \frac{2}{3} \cdot \frac{|A_2|}{|V|}.$$

Therefore $|A_2| \geq 49n/200$.

Now recall that every vertex in V has at most two vertices from A_1 in its closed neighborhood. Therefore, every vertex in V has at most two neighbors in A_2. Let n_7 be the number of vertices in A_2 with degree at least 7. By counting edge endpoints we obtain that $7n_7 \leq 2(n - |A_2|) \leq 151n/100$ and $n_7 \leq 151n/700$. Let $A_3 \subseteq A_2$ be the set of vertices of degree at most 6. Then $|A_3| \geq 41n/1400$.

Now construct $A \subseteq A_3$ greedily. In a single step, add any $v \in A_3$ to A and remove from A_3 the vertex v and all vertices that share a neighbor with v (recall that A_3 is an independent set). Since the vertices in A_3 have degree at most 6 and every vertex in V is a neighbor of at most two vertices in A_3, then at one step we remove at most 7 vertices from A_3. Therefore $|A| \geq 41n/9800$. □

The bound for our iterative-DFS algorithm is now straightforward. Note that for every non-isolated vertex v, at least one vertex in $N[v]$ does not belong to an irredundant set. By Lemma 1 we obtain $41n/9800$ disjoint sets $\{ N[v] : v \in A \}$,

such that all these sets are of size at most 7 and no $N[v]$ can be contained in an irredundant set. Therefore the total number of irredundant sets is bounded by:

$$2^n \cdot \left(\frac{2^7-1}{2^7}\right)^{\frac{41n}{9800}} = O^*(1.99994^n).$$

Let us note here that the algorithm, instead of invoking the procedure Depth-First-Search for $k = 0, 1, 2, \ldots, n$, may check by brute-force all subsets of size at most $149n/300$ and then run Depth-First-Search once with $k = n$. However, we prefer the algorithm written in the iterative-DFS form as it is independent of the constants appearing in the analysis.

Let us emphasize that by Lemma 1 one is able to avoid the maximum degree condition which seemed necessary for this kind of approach (see [31]).

Theorem 1. *The lower irredundance number $ir(G)$ can be computed in time $O^*(1.999956^n)$.*

4. Parameterized complexity: Hardness and Kernels

In order to break the 2^n-barrier using the "win-win"-approach mentioned before, it is tempting to study the problems of computing $\mathrm{ir}(G)$ and $\mathrm{IR}(G)$ from a parameterized complexity viewpoint. Unfortunately, the problem of finding an irredundant set of size at least k is $W[1]$-complete when parameterized by k, as shown by Downey et al. [26]. Therefore, it is unlikely that an algorithm of running time $O(c^k \mathrm{poly}(n))$ for this problem exists.

We now show that computing the lower irredundance number is even harder by giving a reduction from the W[2]-hard Dominating Set problem.

Comparing the hardness results to the aforementioned domination chain $\mathrm{ir}(G) \leq \gamma(G) \leq \alpha(G) \leq \mathrm{IR}(G)$, it is an interesting fact that the problems of computing $\mathrm{IR}(G)$ and $\alpha(G)$ (the Independent Set problem) are both W[1]-complete, while the problems of computing $\mathrm{ir}(G)$ and $\gamma(G)$ are W[2]-hard.

Theorem 2. *The* MinMaxIR *problem is W[2]-hard.*

Proof. We use a reduction from Dominating Set, which is known to be W[2]-hard (cf., [27]). Let $V = \{v_1, \ldots, v_n\}$, $G = (V, E)$, and let (G, k) be a Dominating Set instance. We construct an input instance $(G' = (V', E'), k)$ for MinMaxIR as follows:

For each $v_i \in V$, we add the vertex v_i and the vertices u_i^j and w_i^j for $1 \leq j \leq k+1$ to V'. Let $L = \{\, v_i \in V' \mid v_i \in V \,\} \cup \{\, u_i^j \in V' \mid v_i \in V, 1 \leq j \leq k+1 \,\}$ and $R = \{\, w_i^j \in V' \mid v_i \in V, 1 \leq j \leq k+1 \,\}$.

We add edges between each u_i^j and the corresponding w_i^j as well as between v_i and all w_l^j if v_i and w_l are adjacent in G or if $i = l$. Finally, we add edges between each $x, y \in L$. See Figure 3 for an overview on the construction. For clarity, the edges in L are not shown there. We claim that G has a dominating set of size at most k iff $\mathrm{ir}(G') \leq k$.

Figure 3: Construction used in Theorem 2 for the graph on the left and $k = 2$. All edges within the clique L are not depicted.

Assume that G contains a dominating set D of size at most k. W.l.o.g., we may assume that D is a minimal dominating set.

Thus, for each vertex $v_i \in D$, there is some vertex $v_j \in V$ such that $N[v_j] \cap D = \{v_i\}$, because otherwise D was not minimal. Thus, each vertex in D is a king in G', having an external garden on one of the vertices in $w_j^1, \ldots, w_j^{k+1}$.

Moreover, all vertices in G' are adjacent to some vertex in D, because L is a clique and each vertex in R has at least one neighbor in D, as D dominates G. The set D is therefore a maximal irredundant set of size at most k, which implies that $\mathrm{ir}(G') \leq k$.

Let us now assume that $\mathrm{ir}(G') \leq k$. Let I be a minimum maximal irredundant set of size at most k.

- Assume that $u_i^j \in I$ for some i, j. Recall that $N(w_i^j) \subseteq L$ and that L is a clique. Then $w_i^j \notin I$, as $N[w_i^j] \subseteq N[u_i^j]$ and w_i^j has thus no garden. Furthermore, $N(w_i^j) \cap I = \{u_i^j\}$, because otherwise u_i^j has no garden.

  Obviously, no vertex in $L \subseteq N(u_i^j)$ can be used as a garden for some king in $I \setminus \{u_i^j\}$. Let $u_i^s \notin I$ for some $1 \leq s \leq k+1$. Then $I \cap N(w_i^s) = \emptyset$, because $N(w_i^s) \setminus \{u_i^s\} \subseteq N(w_i^j)$ and $N(w_i^j) \cap I = \{u_i^j\}$. Moreover, we know that no vertex in $N(w_i^s) \subseteq L$ is a garden. These facts imply $w_i^s \in I$ as I is maximal.

  Thus, we have $w_i^s \in I$ or $u_i^s \in I$ for all $1 \leq s \leq k+1$. But then, we have $|I| \geq k+1$, a contradiction.

- We thus know that $I \subseteq V \cup R$. Assume that $w_i^j \in I$ for some i, j. Since $N(w_i^j) \subseteq L$ is a clique, we have $N(w_i^j) \cap I = \emptyset$. Now consider w_i^s for some $1 \leq s \leq k+1$ with $s \neq j$. Since I is maximal and since $N(w_i^s) \setminus N(w_i^j) = \{u_i^s\}$, we know that $w_i^s \notin I$ implies that u_i^s is the (unique) garden of some king $v_l \in I$. Otherwise, making w_i^s a king would not destroy any garden and w_i^s could have an internal garden.

  But since $N(u_i^s) \setminus \{w_i^s\} \subseteq L$ and $I \cap L \subseteq V$, this implies that $v_l \in V \setminus N(w_i^s)$. Moreover, we know that $|I \cap L| = 1$, as otherwise u_i^s would not be a garden. Now by definition, v_l has at least $k+1$ neighbors $w_l^1, \ldots, w_l^{k+1}$ in R, none of which are gardens because u_i^s is the unique garden of v_l. But since $I \cap L = \{v_l\}$, they cannot be gardens of other kings in L except for v_l. Because of $N[w_l^t] \subseteq N[v_l]$ for all $1 \leq t \leq k+1$, we also know that $w_l^t \notin I$, which implies $N[w_l^t] \cap I = \{v_l\}$. Hence, u_i^s is not a unique garden of v_l, as w_l^t is also a garden, a contradiction to our assumption.

  Thus, we have $w_i^s \in I$ for all $1 \leq s \leq k+1$, a contradiction to the fact that $|I| \leq k$.

- Therefore, we know that $I \subseteq V$. Since L is a clique, each $v_i \in I$ has a garden in R: either $|I| = 1$ and every neighbor of v_i can be a garden, or $|I| \geq 2$, which implies that each vertex in L is adjacent to at least two vertices in I and thus cannot act as a garden.

  Now, all w_i^j must have a neighbor in I, because otherwise we can make w_i^j a king without destroying gardens in R. However, this would violate the condition that I is a maximal irredundant set. Therefore, $I \subseteq V$ is a dominating set in G of size at most k.

□

However, Downey, Fellows, and Raman [26] also proved that Co-MaxIR admits a problem kernel of size $3k'^2$ and is therefore fixed parameter tractable. Even though, the running time still has a superexponential dependency on the parameter. We prove that both Co-MinMaxIR and Co-MaxIR admit linear sized problem kernels.

Our first theorem makes use of the aforementioned inequality $\mathrm{ir}(G) \leq \gamma(G) \leq n/2$ in graphs without isolated vertices for Co-MinMaxIR. The second theorem for Co-MaxIR uses crown reductions [32, 33]. This already shows that Co-MinMaxIR and Co-MaxIR allow fixed-parameter tractable algorithms with a running time exponential in k, a new contribution.

Theorem 3. *The* Co-MinMaxIR *problem admits a kernel with at most* $2k-1$ *vertices.*

Proof. It is known that $\mathrm{ir}(G)$ is upper bounded by the domination number $\gamma(G)$ of any graph G of minimum degree one (see above). Since $\gamma(G) \leq n/2$ for any graph of minimum degree one (see, e.g., [34]), we can derive that $\mathrm{ir}(G) \leq n/2$. So, in the given Co-MinMaxIR instance (G, k) we can first delete all isolated vertices (they will be in any maximal irredundant set), without changing the parameter, and then kernelize as follows: if $k \leq n/2$, then we are looking for a maximal irredundant set of size at most $n - k \geq n/2 \geq \mathrm{ir}(G)$, so that we can immediately return YES. If $k > n/2$, then we have obtained the desired kernel, which is just the current graph, with the claimed bound. □

Theorem 4. Co-MaxIR *admits a kernel with at most* $3k$ *vertices.*

Proof. Let $G = (V, E)$ be a graph and let I be an irredundant set of size at least $n - k$ in a graph G.

We use a crown reduction, see [32, 33]. A crown is a subgraph $G' = (C, H, E') = G[C \cup H]$ of G such that C is an independent set in G, H are all neighbors of C in G (i.e., H separates C from $V \setminus (H \cup C)$), and such that there is a matching M of size $|H|$ between C and H.

We first show that if G contains a crown (C, H, E'), then G contains a maximum irredundant set I such that $C \subseteq I$ and $H \subseteq V \setminus I$.

Let I be an irredundant set and let $I_C = \{\, v \in I \cap C \mid N[v] \cap I = \{v\} \,\}$ denote the kings in C with internal gardens. Moreover, let $I_H = H \cap I$ and

$$I_R = \{\, v \in N(H) \mid \exists u \in H \text{ with } N[u] \cap I = \{v\} \,\}.$$

That is, I_H are all kings in H, and I_R are all kings that have a garden in H but are not in H. Moreover, we denote the set of private neighbors of I_R in H by

$$U = \{\, u \in H \mid |N[u] \cap I| = 1 \text{ and } N[u] \cap I_R \neq \emptyset \,\}.$$

Obviously, we have $|U| \geq |I_R|$ and $U \cap I_H = \emptyset$. Moreover, $N[I_C] \cap (U \cup I_H) = \emptyset$, since neighbors of I_C cannot be gardens of other kings and neighbors of I_H cannot have internal gardens. This implies that $|C \setminus I_C| \geq |U \cup I_H| \geq |I_R \cup I_H|$, as the matching M cannot match vertices in $U \cup I_H$ with vertices in I_C. Thus, $|(I \setminus (I_H \cup I_R)) \cup (C \setminus I_C)| \geq |I|$, because $(I \setminus (I_H \cup I_R)) \cap (C \setminus I_C) = \emptyset$. Furthermore, $(I \setminus (I_H \cup I_R)) \cup (C \setminus I_C) = (I \setminus (I_H \cup I_R)) \cup C$ is an irredundant set, since all kings in C have internal gardens and all other remaining vertices do not have gardens in H and are thus not affected by these new vertices in $I \cap C$.

It remains to show that we can always efficiently find a large crown in $G = (V, E)$, provided that $|V| > 3k$.

Let L be a maximal matching in G. We claim that if $|L| > k$, then we can safely return a trivial NO-instance. Let I be a maximum irredundant set in G. Let $I = \mathcal{K}_i \cup \mathcal{K}_e$ be an arbitrary partition of I into internal and external kings. Let $\mathcal{G}_e \subseteq V \setminus I$ be an arbitrary set that can serve as a set of gardens for $\mathcal{K}_e$. Let $\mathcal{W} = V \setminus (I \cup \mathcal{G}_e)$. We assign the following weights $w(x)$ for all $x \in V$: Let $w(x) = 1$ if $x \in \mathcal{W}$, $w(x) = 0$ if $x \in \mathcal{K}_i$, and $w(x) = 0.5$ otherwise. Notice that $|V| - |I| = \sum_{x \in V} w(x)$.

Now consider an edge $\{x, y\} \in L$. If $x \in \mathcal{K}_i$, then $w(x) = 0$, $y \in \mathcal{W}$ and hence $w(y) = 1$. So, consider $x \in (\mathcal{K}_e \cup \mathcal{G}_e)$, say $x \in \mathcal{K}_e$. Then $y \in \mathcal{W} \cup \mathcal{G}_e$ (resp. $y \in \mathcal{W} \cup \mathcal{K}_e$) and hence $w(y) \geq 0.5$. Thus for every $\{x, y\} \in L$, we have $w(u) + w(v) \geq 1$. Therefore, if $|L| > k$, then $|V| - |I| > k$, so that we can return a NO-instance as claimed.

Hence, L contains at most k edges if G contains an irredundant set of size at least $n - k$. This reasoning also holds for maximal matchings that contain no L-augmenting path of length three, a technical notion introduced in [32]. The demonstration given in [32, Theorem 3] shows the claimed kernel bound. □

Lemma 2. *Let $G = (V, E)$ and $v \in V$ with $\deg(v) = 1$. Then there is a maximum irredundant set I for G with $v \in I$.*

This follows immediately from the proof of Theorem 4, since a vertex v with only one neighbor u induces a crown $(\{v\}, \{u\}, \{e\})$ with $e = uv$.

5. A Simple Algorithm for Computing the Irredundance Numbers

In this section we develop a parameterized algorithm that solves both Co-MaxIR and Equal Co-MinMaxIR. Our algorithm for the irredundance numbers recursively branches on the vertices of the graph and assigns each vertex one of the four possible labels $\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}$, until a labeling that forms a solution has been found (if one exists). If I is an irredundant set of size at least $n-k$, then it is easy to see that $|\mathcal{K}_e| + |\mathcal{W}| = |\mathcal{K} \setminus \mathcal{G}| + |\mathcal{W}| \leq k$ and $|\mathcal{G} \setminus \mathcal{K}| + |\mathcal{W}| \leq k$,

which indicates a first termination condition. Furthermore, one can easily observe that for any irredundant set $I \subseteq V$ the following simple properties hold for all $v \in V$:

1. If $|N(v) \cap \mathcal{K}| \geq 2$ then $v \in \mathcal{K} \cup \mathcal{W}$.
2. If $|N(v) \cap \mathcal{G}| \geq 2$ then $v \in \mathcal{G} \cup \mathcal{W}$.
3. If $|N(v) \cap \mathcal{K}| \geq 2$ and $|N(v) \cap \mathcal{G}| \geq 2$ then $v \in \mathcal{W}$.
4. For all $v \in \mathcal{K}_i$, we have $N(v) \subseteq \mathcal{W}$.

This gives us a couple of conditions the labeling has to satisfy in order to yield an irredundant set: each external garden is connected to exactly one external king and vice versa. Once the algorithm constructs a labeling that cannot yield an irredundant set anymore the current branch can be terminated.

Definition 2. *Let $G = (V, E)$ be a graph and let $\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \subseteq V$ be a labeling of V. Let $\overline{V} = V \setminus (\mathcal{K}_i \cup \mathcal{K}_e \cup \mathcal{G}_e \cup \mathcal{W})$. We call* $(\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ valid *if the following conditions hold, and* invalid *otherwise.*

1. *$\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}$ are pairwise disjoint,*
2. *for each $v \in \mathcal{K}_i$, $N(v) \subseteq \mathcal{W}$,*
3. *for each $v \in \mathcal{K}_e$, $N(v) \cap (\mathcal{G}_e \cup \overline{V}) \neq \emptyset$,*
4. *for each $v \in \mathcal{K}_e$, $|N(v) \cap \mathcal{G}_e| \leq 1$,*
5. *for each $v \in \mathcal{G}_e$, $N(v) \cap (\mathcal{K}_e \cup \overline{V}) \neq \emptyset$, and*
6. *for each $v \in \mathcal{G}_e$, $|N(v) \cap \mathcal{K}_e| \leq 1$.*

As a direct consequence, we can define a set of vertices that can no longer become external gardens or kings without invalidating the current labeling:

$$\begin{aligned} \text{Not}\mathcal{G} &:= \{\, v \in \overline{V} \mid \text{the labeling } (\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \text{ is invalid} \,\} \\ \text{Not}\mathcal{K} &:= \{\, v \in \overline{V} \mid \text{the labeling } (\mathcal{K}_i, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) \text{ is invalid} \,\} \end{aligned}$$

It is easy to see that Not$\mathcal{G}$ and Not$\mathcal{K}$ can be computed in polynomial time, and since vertices in Not$\mathcal{G} \cap$ Not$\mathcal{K}$ can only be wilderness, we can also assume that Not$\mathcal{G} \cap$ Not$\mathcal{K} = \emptyset$ once the following reduction rules have been applied.

Let $G = (V, E)$ be a graph and let $\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \subseteq V$ be a valid labeling of V. Let $\overline{V} = V \setminus (\mathcal{K}_i \cup \mathcal{K}_e \cup \mathcal{G}_e \cup \mathcal{W})$. We define the following reduction rules, to be applied in this order, one at a time:

R_1 If there is some $v \in \mathcal{W}$ with $\deg(v) > 0$, remove all edges incident to v.

R_2 If there is some $v \in \overline{V}$ with $\deg(v) = 0$, then set $\mathcal{K}_i := \mathcal{K}_i \cup \{v\}$.

R_3 If there is some isolated edge uv, where $u, v \in \overline{V}$ with $\deg(u) = \deg(v) = 1$, set $\mathcal{K}_i := \mathcal{K}_i \cup \{v\}$ and $\mathcal{W} := \mathcal{W} \cup \{u\}$.

R_4 If there is $v \in \mathcal{K}_e$ with $N(v) \cap \mathcal{G}_e = \emptyset$ and $N(v) \cap \overline{V} = \{w\}$, then set $\mathcal{G}_e := \mathcal{G}_e \cup \{w\}$.

R_5 If there is $v \in \mathcal{G}_e$ with $N(v) \cap \mathcal{K}_e = \emptyset$ and $N(v) \cap \overline{V} = \{w\}$, then set $\mathcal{K}_e := \mathcal{K}_e \cup \{w\}$.

Algorithm 2 A fast yet simple algorithm for Co-MaxIR.

Algorithm CO-IR$(G, k, \mathcal{K}_e, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W})$:
Input: Graph $G = (V, E)$, $k \in \mathbf{N}$, labels $\mathcal{K}_e$, $\mathcal{K}_i$, $\mathcal{G}_e$, $\mathcal{W} \subseteq V$

Compute the sets Not$\mathcal{G}$, Not$\mathcal{K}$.
Apply the reduction rules exhaustively, updating Not$\mathcal{G}$ and Not$\mathcal{K}$.
if the current labeling is invalid **then** return NO.
if $|\mathcal{K}_e| + |\mathcal{W}| + |\text{Not}\mathcal{G}| > k$ **or** $|\mathcal{G}_e| + |\mathcal{W}| + |\text{Not}\mathcal{K}| > k$ **then** return NO.
if $|\mathcal{K}_e| + |\mathcal{W}| = k$ **or** $|\mathcal{G}_e| + |\mathcal{W}| = k$ **or** all vertices are labeled **then**
    return whether $V \setminus (W \cup \mathcal{G}_e)$ is a solution.
if Not$\mathcal{G} \neq \emptyset$ (or analogously, Not$\mathcal{K} \neq \emptyset$) **then**
    choose $v \in \text{Not}\mathcal{G}$;
    return CO-IR$(G, k, \mathcal{K}_e \cup \{v\}, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W})$
        or CO-IR$(G, k, \mathcal{K}_e, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W} \cup \{v\})$
Choose (in this preferred order) unlabeled $v \in V$ of degree one,
of maximum degree with $N(v) \cap (\mathcal{G}_e \cup \mathcal{K}_e) \neq \emptyset$ or any unlabeled v
with maximum degree.
return CO-IR$(G, k, \mathcal{K}_e, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W} \cup \{v\})$
    or CO-IR$(G, k, \mathcal{K}_e, \mathcal{K}_i \cup \{v\}, \mathcal{G}_e, \mathcal{W} \cup N(v))$
    or $\exists u \in N(v) \setminus (\mathcal{K}_e \cup \mathcal{K}_i \cup \mathcal{W})$:
        CO-IR$(G, k, \mathcal{K}_e \cup \{v\}, \mathcal{K}_i, \mathcal{G}_e \cup \{u\}, \mathcal{W})$
    or $\exists u \in N(v) \setminus (\mathcal{G}_e \cup \mathcal{K}_i \cup \mathcal{W})$:
        CO-IR$(G, k, \mathcal{K}_e \cup \{u\}, \mathcal{K}_i, \mathcal{G}_e \cup \{v\}, \mathcal{W})$

R_6 For every $v \in \text{Not}\mathcal{G} \cap \text{Not}\mathcal{K}$, set $\mathcal{W} := \mathcal{W} \cup \{v\}$.

A graph and a labeling of its vertices as above is called *reduced* if no further reduction rules can be applied.

Lemma 3. *Let $G = (V, E)$ be a graph and let $\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \subseteq V$ be a valid labeling of V and let $I \subseteq V$ be a maximal irredundant set respecting $L = (\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$. Let G' and $L' = (\mathcal{K}_i', \mathcal{K}_e', \mathcal{G}_e', \mathcal{W}')$ be the graph and labeling obtained from G and L by applying one of the reduction rules. Then there is also a maximal irredundant set of size $|I|$ that respects L'.*

Proof. R_6 is obvious. R_1–R_3 are true, since isolated vertices can always be added to $\mathcal{K}_i$ without decreasing the size of a solution, endpoints of an isolated edge can contain at most one king, and adding or removing edges incident to vertices in wilderness cannot result in an invalid labeling. R_4 is true, since $v \in \mathcal{K}_e$ needs a vertex in $\mathcal{G}_e$ as a neighbor, but there is only one possibility left. (Likewise for $v \in \mathcal{G}_e$, rule R_5.) □

Since Algorithm 2 uses exhaustive branching, we easily obtain:

Lemma 4. *Algorithm 2 correctly solves* Co-MaxIR *(when initially called with empty label sets).*

Proof. Note that in each recursive call at least one vertex is added to $\mathcal{K}_e \cup \mathcal{G}_e \cup \mathcal{W}$. Thus, the search tree has a height of at most $2k$ (see also the runtime analysis). Moreover, it can never falsely output YES, as solutions are verified in Line 6.

Thus, we can assume that there is some solution, an irredundant set $\mathcal{I}$ with a corresponding set of gardens $\mathcal{G}$. But then, $\mathcal{I}$ and $\mathcal{G}$ imply a labeling $(\overline{\mathcal{K}_i}, \overline{\mathcal{K}_e}, \overline{\mathcal{G}_e}, \overline{\mathcal{W}})$ by setting $\overline{\mathcal{K}_i} := \mathcal{I} \cap \mathcal{G}$, $\overline{\mathcal{K}_e} := \mathcal{I} \setminus \mathcal{G}$, adding a unique external garden in $N(v) \cap \mathcal{G}$ for each $v \in \overline{\mathcal{K}_e}$ into $\overline{\mathcal{G}_e}$ and setting $\overline{\mathcal{W}} := V \setminus (\overline{\mathcal{K}_i} \cup \overline{\mathcal{K}_e} \cup \overline{\mathcal{G}_e})$.

It remains to show inductively that Algorithm 2 returns YES if called on a labeling $(\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ such that $\overline{\mathcal{K}_i} \supseteq \mathcal{K}_i$, $\overline{\mathcal{K}_e} \supseteq \mathcal{K}_e$, $\overline{\mathcal{G}_e} \supseteq \mathcal{G}_e$, and $\overline{\mathcal{W}} \supseteq \mathcal{W}$.

Note that $(\overline{\mathcal{K}_i}, \overline{\mathcal{K}_e}, \overline{\mathcal{G}_e}, \overline{\mathcal{W}})$ can only be valid, if $(\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ is valid, too. Moreover, $(\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ must obviously be valid. Since $|\mathcal{I}| \geq |V| - k$, and vertices in Not$\mathcal{G}$ can be labeled only $\mathcal{K}_e$ or $\mathcal{W}$ (similarly vertices in Not$\mathcal{K}$ must be labeled $\mathcal{G}_e$ or $\mathcal{W}$) we have $|\mathcal{K}_e| + |\mathcal{W}| + |\text{Not}\mathcal{G}| \leq k$ and $|\mathcal{G}_e| + |\mathcal{W}| + |\text{Not}\mathcal{K}| \leq k$.

If we have $|\mathcal{K}_e| + |\mathcal{W}| = k$ or $|\mathcal{G}_e| + |\mathcal{W}| = k$, the algorithm obviously outputs YES, if we have found $\mathcal{I}$. If $|\mathcal{K}_e| + |\mathcal{W}| < k$ or $|\mathcal{G}_e| + |\mathcal{W}| < k$, there are two possibilities:

- All vertices are labeled, and the algorithm has found $\mathcal{I}$, which is checked in Line 6.
- Some vertex $v \in V$ is not labeled yet. If $v \in \text{Not}\mathcal{G}$, it cannot be in $\overline{\mathcal{G}_e}$, as this would imply that $(\overline{\mathcal{K}_i}, \overline{\mathcal{K}_e}, \overline{\mathcal{G}_e}, \overline{\mathcal{W}})$ is invalid, since $(\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e \cup \{v\}, \mathcal{W})$ is invalid. Thus, it suffices to test the two possibilities $v \in \mathcal{K}_e$ or $v \in \mathcal{W}$ to find the correct solution (and similar for $v \in \text{Not}\mathcal{K}$).

  If $v \notin \text{Not}\mathcal{G} \cup \text{Not}\mathcal{K}$, Algorithm 2 exhaustively branches on v (including the choice which vertex acts as external garden or external king), which obviously yields the correct solution. □

Let $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ be the number of recursive calls that reach Line 5 where none of the (possibly zero) following recursive calls (in Lines 9 and 11) reach this line. This way, we do not count recursive calls that fail immediately in the first four lines. This allows us to ignore the up to $2\deg(v) + 2$ failing calls (Line 11), that only contribute a polynomial runtime factor (since they do not trigger further recursive calls).

Note that recursive calls only require polynomial time. Moreover, the depth of the recursion tree is bounded by $2k$. Thus the running time of Algorithm 2 is polynomially bounded in $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$. Let our measure be

$$\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) = k - |\mathcal{W}| - 0.5|\mathcal{K}_e| - 0.5|\mathcal{G}_e|.$$

We first show that no recursive calls will be triggered if $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) < 0$. Since furthermore the reduction rules clearly do not increase the measure, $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$ is indeed a valid measure.

Lemma 5. *If $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) < 0$, then the algorithm correctly outputs* NO.

Proof. If $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) < 0$, then we claim that $|\mathcal{K}_e| + |\mathcal{W}| + |\text{Not}\mathcal{G}| > k$ or $|\mathcal{G}_e| + |\mathcal{W}| + |\text{Not}\mathcal{K}| > k$. Assume $|\mathcal{K}_e| + 2|\mathcal{W}| + |\mathcal{G}_e| + |\text{Not}\mathcal{G}| + |\text{Not}\mathcal{K}| \leq 2k$ contrary. Therefore, we can deduce $0.5 \cdot (|\mathcal{K}_e| + |\mathcal{G}_e|) + |\mathcal{W}| \leq k$, which contradicts the fact that $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) < 0$. □

Lemma 6. *$T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}$ with $\alpha \leq 3.841$.*

Proof. We prove the claim by induction over search trees for $(G, k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$. In the following, we analyze each of the many possible cases how the algorithm branches. To get a better bound, we sometimes include subsequent calls in the estimation.

Let $\beta := \frac{1+\alpha^{0.5}}{\alpha} = \alpha^{-1} + \alpha^{-0.5} < 1$. The first case is used later on to overcome some bad cases. We are thus forced to analyze it very closely. More precisely, we need to guarantee that a certain number of "good" branches are executed.

Case 1. $\text{Not}\mathcal{G} \cup \text{Not}\mathcal{K} \neq \emptyset$.

Proof. We can assume that both $|\mathcal{K}_e| + |\mathcal{W}| + |\text{Not}\mathcal{G}| \leq k$ and $|\mathcal{G}_e| + |\mathcal{W}| + |\text{Not}\mathcal{K}| \leq k$ hold, because otherwise this branch fails immediately. Let $d = |\text{Not}\mathcal{G} \cup \text{Not}\mathcal{K}| = |\text{Not}\mathcal{G}| + |\text{Not}\mathcal{K}|$, where the last equivalence follows from reduction rule R_6. Note that this implies that $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \geq (|\text{Not}\mathcal{G}| + |\text{Not}\mathcal{K}|)/2$.

We show $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \beta^d$ by induction over d. Let $v \in \text{Not}\mathcal{G} \cup \text{Not}\mathcal{K}$ and assume, w.l.o.g., that $v \in \text{Not}\mathcal{G}$.

We can assume that both recursive calls on v do reach Line 5, because otherwise we have either $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) = T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) - 0.5}$ or $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) = T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) - 1}$, and $\alpha^{-0.5} \leq \beta$ as well as $\alpha^{-1} \leq \beta$.

For $d = 1$, we therefore obtain by our overall induction over $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})$

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \\ &\leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}(\alpha^{-0.5} + \alpha^{-1}) = \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}\beta. \end{aligned}$$

We can hence assume $d > 1$.

If both recursive calls reach Line 5, we have $\varphi(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) \geq |\text{Not}\mathcal{G} \setminus \{v\} \cup \text{Not}\mathcal{K}|/2 = (d-1)/2$ and $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \geq |\text{Not}\mathcal{G} \setminus \{v\} \cup \text{Not}\mathcal{K}|/2 = (d-1)/2$, because otherwise the condition in Line 4 is already true. If the call where $v \in \mathcal{K}_e$ satisfies one of the conditions in Line 5, we therefore have

$$T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) = 1 \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \alpha^{-(d-1)/2} \alpha^{-0.5}$$

and

$$T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) = 1 \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \alpha^{-(d-1)/2} \alpha^{-1}$$

in the call where $v \in \mathcal{W}$. We used here the trivial observation that

$$\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) = \varphi(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) + 0.5 = \varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) + 1.$$

Thus, the two recursive calls by induction over d yield the bounds

$$T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) \leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\alpha^{-0.5} \max\left\{\alpha^{-(d-1)/2}, \beta^{d-1}\right\},$$

and

$$T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\alpha^{-1} \max\left\{\alpha^{-(d-1)/2}, \beta^{d-1}\right\}.$$

But since $\beta = \alpha^{-1} + \alpha^{-0.5} > \alpha^{-0.5}$ we have that

$$\alpha^{-(d-1)/2} = (\alpha^{-0.5})^{d-1} \leq \beta^{d-1},$$

and we obtain

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathcal{W}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\left(\beta^{d-1}\alpha^{-0.5} + \beta^{d-1}\alpha^{-1}\right) \\ &= \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\beta^{d-1}(\alpha^{-1} + \alpha^{-0.5}) = \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\beta^{d}. \end{aligned}$$

□

In the following, we assume Not$\mathcal{G}$ = Not$\mathcal{K}$ = $\emptyset$ and let $\overline{V} = V \setminus (\mathcal{K}_i \cup \mathcal{K}_e \cup \mathcal{G}_e \cup \mathcal{W})$ be the set of yet unlabeled vertices. Also note that for all $v \in \overline{V}$ we have that $|N(v) \cap \mathcal{K}_e| \leq 1$ and $|N(v) \cap \mathcal{G}_e| \leq 1$.

Case 2. *$v \in \overline{V}$ such that $N(v) = \{u\}$.*

Proof. Let $v \in \overline{V}$ be a vertex of degree one and let $\{u\} = N(v)$. By the reduction rules (removal of edges), $u \notin \mathcal{W}$ and by the preferred branching for Not$\mathcal{K}$ and Not$\mathcal{G}$ vertices, $u \in \overline{V}$. We can assume $N := N(u) \setminus \{v\} \neq \emptyset$. Indeed, otherwise uv is an isolated edge and should be resolved by reduction R_3. Distinguish the following two cases:

If there is $z \in N$ with $z \in \overline{V}$, then z will become a member of Not$\mathcal{G}$ in the branch $v \in \mathcal{G}_e$, since u must be the king of v, and similarly z will become a member of Not$\mathcal{K}$ in the $v \in \mathcal{K}_e$ branch. After inserting Case 1 above once, we gain:

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{u\}) + \\ &\quad T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u\}, \mathcal{W}) + T(k, \mathcal{K}_e \cup \{u\}, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\left(\alpha^{-1} + \alpha^{-1} + 2\alpha^{-1} \cdot \beta\right) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}. \end{aligned}$$

Finally, if there is $z \in N \cap \mathcal{K}_e$ (and similarly, $z \in N \cap \mathcal{G}_e$), then the branch $v \in \mathcal{K}_e$ immediately fails. Therefore,

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{u\}) + \\ &\quad T(k, \mathcal{K}_e \cup \{u\}, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\left(\alpha^{-1} + \alpha^{-1} + \alpha^{-1}\right) \leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}. \end{aligned}$$

□

Case 3. *$v \in \overline{V}$ such that $N(v) \cap \mathcal{K}_e = \{v_K\}$ and $N(v) \cap \mathcal{G}_e = \{v_G\}$ for some vertices v_K and v_G.*

Proof. Note that the branch $v \in \mathcal{K}_i$ does not reach Line 5, as this imposes an invalid coloring. The same holds for each recursive call where $v \in \mathcal{K}_e$ and $u \in N(v) \setminus (\mathcal{K}_e \cup \mathcal{K}_i \cup \mathcal{W})$ except for the case where $u = v_G$ due to invalid labeling (v is a $\mathcal{K}_e$ king with two external gardens $\mathcal{G}_e$). Similarly, the branches where $u \neq v_K$ do not reach Line 5 for the cases where we try $v \in \mathcal{G}_e$.

Furthermore, $N(v_K) \cap \mathcal{G}_e = \emptyset$, otherwise $v \in \text{Not}\mathcal{G}$. Similarly, $N(v_G) \cap \mathcal{K}_e = \emptyset$. In particular, v_K and v_G are not adjacent. Moreover, v_K and v_G must have at least another unlabeled neighbor (maybe the same) since the instance is reduced.

If $|N(v_K) \setminus \mathcal{K}_e| = 2$, setting $\mathcal{K}_e = \mathcal{K}_e \cup \{v\}$ implies that the remaining unlabeled neighbor u_K of v_K must be added to $\mathcal{G}_e$ (and analogously u_G in case we consider v_G).

Thus, if both $|N(v_K) \setminus \mathcal{K}_e| = 2$ and $|N(v_G) \setminus \mathcal{G}_e| = 2$, we obtain

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u_K\}, \mathcal{W}) \\ &\quad + T(k, \mathcal{K}_e \cup \{u_G\}, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \\ &\quad + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) \\ &\leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}(\alpha^{-1} + \alpha^{-0.5-0.5} + \alpha^{-0.5-0.5}) \\ &\leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \end{aligned}$$

If $|N(v_K) \setminus \mathcal{K}_e| = 2$ and $|N(v_G) \setminus \mathcal{G}_e| > 2$, we only obtain

$$T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \quad \leq \quad \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}(\alpha^{-1} + \alpha^{-0.5-0.5} + \alpha^{-0.5}).$$

However, after setting $\mathcal{K}_e = \mathcal{K}_e \cup \{v\}$, all other neighbors of v_G cannot be kings. Since at least one of these vertices u_1 is not the neighbor of v_K (recall that $N(v_G) \cap \mathcal{K}_e = \emptyset$), at least one vertex is added to Not$\mathcal{K}$ in this case. The remaining unlabeled neighbor u_K of v_K must be added to $\mathcal{G}_e$.

After setting $\mathcal{G}_e = \mathcal{G}_e \cup \{v\}$, the unique neighbor u_K of v_K cannot be a garden, and is thus added to Not$\mathcal{G}$.

Combining the branch on v with the branches on Not$\mathcal{K}$ and Not$\mathcal{G}$ in the very next recursive calls yields

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) &\leq \alpha^{\varphi(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u_1, u_K\}, \mathcal{W})} + \alpha^{\varphi(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u_K\}, \mathcal{W} \cup \{u_1\})} \\ &\quad + \alpha^{\varphi(k, \mathcal{K}_e \cup \{u_K\}, \mathcal{G}_e \cup \{v\}, \mathcal{W})} + \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e \cup \{v\}, \mathcal{W} \cup \{u_K\})} \\ &\quad + \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\})} \\ &= \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}(\alpha^{-1.5} + \alpha^{-2} + \alpha^{-1} + \alpha^{-1.5} + \alpha^{-1}) \\ &\leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}. \end{aligned}$$

Hence, we can assume, w.l.o.g, that $|N(v_K) \setminus \mathcal{K}_e| > 2$ and $|N(v_G) \setminus \mathcal{G}_e| > 2$. But then, we gain at least two vertices in Not$\mathcal{G}$ or Not$\mathcal{K}$ whenever $v \notin \mathcal{W}$. This

implies (analogously to Case 1)

$$\begin{aligned} T(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W}) &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\Bigg(\alpha^{-1} + \sum_{i=0}^{2}\binom{2}{i}\alpha^{-0.5-i\cdot 0.5-(2-i)\cdot 1} \\ &\qquad + \sum_{i=0}^{2}\binom{2}{i}\alpha^{-0.5-i\cdot 0.5-(2-i)\cdot 1}\Bigg) \\ &= \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}(\alpha^{-1}+2\alpha^{-1.5}+4\alpha^{-2}+2\alpha^{-2.5}) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})} \end{aligned}$$

□

Case 4. *$v \in \overline{V}$ of maximum degree, such that $N(v)\cap\mathcal{K}_e = \{v_K\}$ and $N(v)\cap\mathcal{G}_e = \emptyset$ for some vertex v_K.*

Proof. Let $d := \deg(v)$. Again, note that the branch $v \in \mathcal{K}_i$ does not reach Line 5, and whenever we branch $v \in \mathcal{G}_e$, the same holds unless the respective $u = v_K$.

For the $v \in \mathcal{K}_e$ branch, however, we need to test all possible external gardens, which are $d-1$ branches. Whenever we branch $v \in \mathcal{K}_e$ (or $v \in \mathcal{G}_e$), the $d-2$ vertices in $N(v) \setminus \{v_K, u\}$ (the $d-1$ vertices in $N(v) \setminus \{v_k\}$) become Not$\mathcal{G}$ (Not$\mathcal{K}$) vertices in the very next branch. Branching on these vertices will give us a bonus to overcome the poor branching on v. It should be noted that none of these branches reaches Line 5, if $|\text{Not}\mathcal{G} \cup \text{Not}\mathcal{K}|$ becomes too large, i.e., $|\text{Not}\mathcal{G} \cup \text{Not}\mathcal{K}|/2 > \varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})$. Using the bound of Case 1, we thus obtain

$$\begin{aligned} T(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W}) &\leq T(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W}\cup\{v\}) + T(k,\mathcal{K}_e,\mathcal{G}_e\cup\{v\},\mathcal{W}) \\ &\quad + \sum_{u\in N(v)\setminus\{v_K\}} T(k,\mathcal{K}_e\cup\{v\},\mathcal{G}_e\cup\{u\},\mathcal{W}) \\ &\leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}\left(\alpha^{-1} + \alpha^{-0.5}\cdot\beta^{d-1} + (d-1)\alpha^{-1}\cdot\beta^{d-2}\right) \end{aligned}$$

for any $d \geq 2$. Since

$$f(d) := \alpha^{-1} + \alpha^{-0.5}\cdot\beta^{d-1} + (d-1)\alpha^{-1}\cdot\beta^{d-2}$$

is strictly decreasing for $d \geq 4$, $f(2) \leq 0.914$, $f(3) \leq 0.965$, and $f(4) \leq 0.958$, we have

$$T(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W}) \leq \alpha^{\varphi(k,\mathcal{K}_e,\mathcal{G}_e,\mathcal{W})}.$$

□

Case 5. *The case $N(v)\cap\mathcal{G}_e = \{v_G\}$ and $N(v)\cap\mathcal{K}_e = \emptyset$ for some v_G is analogous to Case 4.*

Case 6. *$v \in \overline{V}$ of maximum degree, such that $N(v) \subseteq \overline{V}$.*

Proof. Here, the $\mathcal{K}_i$ branch can reach Line 5, but enforces $N(v) \subseteq \mathcal{W}$.

Just as in the previous case, if v becomes a king with external garden, the branching "guesses" where the corresponding garden is (the same holds for the garden branch). Note that $N(u) \subseteq \overline{V}$ as well, since otherwise the algorithm would branch on u instead.

We obtain the general recurrence

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \leq\ & T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup N(v)) \\ & + \sum_{u \in N(v)} T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u\}, \mathcal{W}) \\ & + \sum_{u \in N(v)} T(k, \mathcal{K}_e \cup \{u\}, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \end{aligned}$$

In the branches where $v \in \mathcal{K}_e$ and some $u \in N(v)$ becomes its unique external garden, we also restrict the further possibilities of all vertices in $N(\{v, u\})$: vertices in $N(v) \setminus N[u]$ cannot become external gardens, vertices in $N(u) \setminus N[v]$ cannot become kings, and thus in particular all the vertices in $N(v) \cap N(u)$ must become wilderness. For each $u \in N(v)$, we let $S_u = N(v) \cap N(u)$ and $T_u = N(\{v, u\} \setminus (N(v) \cap N(u)))$. We thus obtain by the induction hypothesis (and inserting Case 1),

$$T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u\}, \mathcal{W}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \cdot \alpha^{-1} \alpha^{-|S_u|} \beta^{|T_u|}$$

Note that $|T_u| + 2|S_u| = \deg(v) + \deg(u) - 2$ and $\alpha^{-|S_u|} \leq \beta^{2|S_u|}$ implies

$$\alpha^{-1} \cdot \alpha^{-|S_u|} \beta^{|T_u|} \leq \alpha^{-1} \cdot \beta^{\deg(u) + \deg(v) - 2}.$$

Since the case $v \in \mathcal{G}_e$ is similar and since $\deg(u) \geq 2$, we obtain

$$\begin{aligned} T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \leq\ & T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup \{v\}) + T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W} \cup N(v)) \\ & + \sum_{u \in N(v)} T(k, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u\}, \mathcal{W}) \\ & + \sum_{u \in N(v)} T(k, \mathcal{K}_e \cup \{u\}, \mathcal{G}_e \cup \{v\}, \mathcal{W}) \\ \leq\ & \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \Big(\alpha^{-1} + \alpha^{-d} + 2 \sum_{u \in N(v)} \alpha^{-1} \cdot \beta^{\deg(u) + \deg(v) - 2} \Big) \\ \leq\ & \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \left(\alpha^{-1} + \alpha^{-\deg(v)} + 2 \cdot \deg(v) \cdot \alpha^{-1} \cdot \beta^{\deg(v)} \right). \end{aligned}$$

Finally, we have

$$1 \quad \geq \quad \alpha^{-1} + \alpha^{-\deg(v)} + 2 \cdot \deg(v) \cdot \alpha^{-1} \cdot \beta^{\deg(v)}$$

because $f(d) := \alpha^{-1} + \alpha^{-d} + 2 \cdot d \cdot \alpha^{-1} \cdot \beta^d$ is strictly decreasing for $d \geq 4$ and $f(2) \leq 0.947$, $f(3) \leq 0.993$, and $f(4) \leq 0.9994$. We therefore obtain the desired bound

$$\alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})} \left(\alpha^{-1} + \alpha^{-\deg(v)} + 2 \cdot \deg(v) \cdot \alpha^{-1} \cdot \beta^{\deg(v)} \right) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}$$

for $\alpha \geq 3.841$. □

This concludes the proof. □

Remark 1. *Algorithm 2 can also be used, with slight modifications, to answer the question whether a graph G has a minimum maximal co-irredundant set of size exactly k (*Equal Co-MinMaxIR*). Namely, if the measure drops to zero, then either the current labeling corresponds to a valid co-irredundant set of size k that is maximal or not; this has to be tested in addition.*

Theorem 5. Co-MaxIR *and* Equal Co-MinMaxIR *can be solved in time $O(3.841^k \text{poly}(n))$, and thus it can be tested in time $O(3.841^k \text{poly}(n))$ whether $\text{IR}(G) \geq n - k$ and whether $\text{ir}(G) = n - k$. Consequently, the lower and the upper irredundance number of a graph G with n vertices can be computed in time $O^*(1.99914^n)$.*

Proof. The first part of the theorem is a straightforward corollary from the previous discussion. For the second part, first enumerate all vertex subsets of size up to $0.485252n$. Then for all $1 \leq k \leq 0.514748n$ invoke the algorithm for Co-MaxIR. □

We remark that with some changes to Algorithm 2 and a more involved analysis using a new measure, the running time required to compute $\text{IR}(G)$ can be further improved, see [35, 1].

Theorem 6. Co-MaxIR *can be solved in time $O(3.069^k \text{poly}(n))$. Thus $\text{IR}(G)$ can be computed in time $O^*(1.9601^n)$.*

We skip a presentation of these mainly technical improvements, since the method presented in the next section improves the running time even further.

6. Computing IR(G)

In this section we provide a branching algorithm to compute a maximum (size) irredundant set. Notice that a maximum irredundant set is necessarily maximal. In [2], we presented a branching algorithm that does not use the "detour" via some corresponding parameterized problem, but uses a reduction to the problem of finding a maximum induced matching (also known in the literature as a strong matching) in bipartite graphs (the Bipartite Induced Matching problem). In [35, 1], we presented a major modification of the algorithm from Section 5 that allows us to obtain a faster parameterized algorithm for the Co-MaxIR problem, see Theorem 6. Here we merge the two ideas: We develop a branching algorithm for the Bipartite Induced Matching problem (as defined below), but we analyze it using the parameterized approach. The resulting algorithm is both faster and simpler than the previous ones [1, 2]. However, note that while using the parameterized approach we cannot use the memoization trick as in [2] and achieve a faster algorithm at the cost of exponential space complexity: For memoization in FPT-algorithms, we need a

vertex linear sized problem kernel that respects induced subgraphs (also called an induced kernel in [36]). We are not aware of such a kernel for the Bipartite Induced Matching problem.

Let us now give the formal definitions. We call a set of edges $M \subseteq F$ in a graph $H = (W, F)$ an *induced matching* if:

- no two edges in M share an endpoint;
- the set $W(M)$ is an independent set in the graph $(W, F \setminus M)$, i.e., no edge connects endpoints of different edges from M.

In other words, the graph induced by the set of endpoints of M is the matching M. The parameterized Bipartite Induced Matching problem is defined as follows.

Bipartite Induced Matching

Input:	An undirected bipartite graph $H = (W, F)$.
Parameter:	k
Question:	Does there exist an induced matching M in H, such that at most k vertices are *not* endpoints of edges in M? (i.e., $\|W\| - 2\|M\| \leq k$)

Notice that this type of parameterization is non-standard and does not correspond to the usual dual parameterization of this edge-selection problem. We rather took a kind of "vertex dual" of the natural parameterization of this maximization problem, since this nicely links to the parameterized dual of Irredundant Set, as seen below.

6.1. Bipartite induced matching reduction

For a graph $G = (V, E)$ let us construct a bipartite graph $H = (W, F)$, with $W = V \cup V'$ (where V' is a disjoint copy of V) and edges $\{u, v'\} \in F$ iff $uv \in E$ or $u = v$. We show a correspondence between maximum irredundant sets in G and maximum induced matchings in H:

Lemma 7. *If M is an induced matching in H, then $S := W(M) \cap V$ is an irredundant set in G. Conversely, if S is an irredundant set in G, then there exists an induced matching $M \subseteq F$ such that $S = W(M) \cap V$. In both cases, the induced matching M and the irredundant set S are of the same size.*

Proof. Let T be an induced matching in H. If $uv' \in M$, then v' is not a neighbor of any $w \in W(M) \cap V$ other than u, thus v is a private vertex dominated by u in G. Conversely, if we have an irredundant set S in G then letting $u(v)$ to be any private vertex dominated by v for $v \in S$ we obtain an induced matching $M := \{\, \{v, u(v)\} \mid v \in S \,\}$ in H.

□

Thus, to find a maximum irredundant set in G it suffices to look for a maximum induced matching in H.

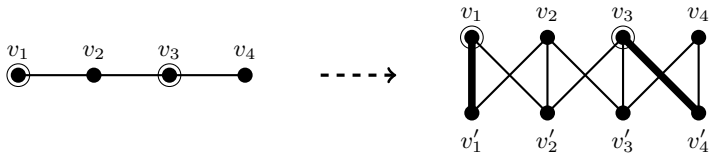

Figure 4: From the irredundant set $\{v_1, v_3\}$ to the induced matching $\{v_1v'_1, v_3v'_4\}$.

Note that the same reduction was presented by Moser and Sikdar in [37] to show W[1]-hardness of the problem of determining an induced matching in bipartite graphs of size at least k.

A remark to make here is that the correspondence between induced matchings of H and irredundant sets of G does not preserve maximality — e.g., in the graph in Figure 4 there are four maximal irredundant sets ($\{v_1, v_3\}$, $\{v_1, v_4\}$, $\{v_2, v_3\}$, and $\{v_2, v_4\}$), while in the corresponding bipartite graph the edge $v_2v'_3$ forms a singleton maximal induced matching. This does no harm in the search for $\mathrm{IR}(G)$ (as the size of the set is preserved), but prevents us from using this approach to determine $\mathrm{ir}(G)$.

6.2. Branch and reduce analysis

Let $\alpha = 1.69563$. We are to develop an algorithm that, given a bipartite graph H on n vertices and an integer k, decides whether there exists an induced matching M in H, such that at most k vertices of H are not endpoints of M. The algorithm runs in $O(\alpha^k \mathrm{poly}(n))$ time. This yields an $O(\alpha^{2k}\mathrm{poly}(n)) = O(2.8752^k\mathrm{poly}(n))$ algorithm for Co-MaxIR. To use it in the win–win approach, assume we have a Maximum Irredundant Set instance and we need to determine whether there exists an irredundant set of size at least $n-k$. If $n-k \leq 0.374043\,n$, we enumerate all subsets. Otherwise, we construct the bipartite graph and look for an induced matching such that at most $2k$ vertices are not endpoints of the matching. This leads to an $O^*(1.9369^n)$ algorithm to compute $\mathrm{IR}(G)$.

The algorithm is a typical branch-and-reduce algorithm. We provide a set of reduction and branching rules; in a single step the algorithm finds the first applicable rule and applies it. By $T(k)$ we denote the number of leaves of the search tree of the algorithm for a graph with n vertices and for the parameter k. For each rule, we provide a recurrence for $T(k)$. Since the function $T(k) = C \cdot \alpha^k$ satisfies all inequalities for any $C > 0$, every step takes polynomial time, and the depth of the search tree is polynomial (we simply delete at least one vertex in each step), the algorithm halts within the claimed time bound. The algorithm uses polynomial space.

Let $H = (W, F)$ be a bipartite graph on n vertices. We use the word *choose* (an edge) to mean "consider as a part of the induced matching being built in the considered subcase", consequently we remove two endpoints of the chosen edge from the graph in this subcase. Similarly we use the word *drop* (a vertex) to mean "consider that this vertex is not the endpoint of any edge in the induced

matching being built in the considered subset, so we can consider the graph without this vertex in this subcase".

1. If $k < 0$, return NO from the current branch.
2. There is an isolated vertex $v \in W$. Obviously it cannot be an endpoint of an edge, so we drop it and solve the problem for $(W \setminus \{v\}, F)$ without branching.
3. There is a vertex v of degree 1 in H, and its only neighbor u is of degree at most 2. Then we can greedily take uv as a part of the constructed maximum induced matching. If u is of degree 1 this holds trivially. Otherwise, if w is the other neighbor of u and if in any induced matching an edge wx is used, we may instead use edge uv. Thus we choose it and solve the problem for the remaining graph without branching.
4. There is a connected component C of the graph that is a simple cycle of length m. Such a cycle contains a maximum induced matching of size $\lfloor m/3 \rfloor$. Thus we remove C and decrease k by $m - 2\lfloor m/3 \rfloor$ without branching.
5. There is a vertex v of degree 1 in H. Let u be its only neighbor, $\deg(u) \geq 3$. Consider any induced matching M which contains some edge ux for $x \neq v$. Then $(M \setminus \{ux\}) \cup \{uv\}$ is also an induced matching of the same size. Thus we can branch out into two cases — either we drop u (and consequently drop v, as it becomes isolated), or we choose uv and drop all the neighbors of u (at least two of them). Here we obtain the inequality $T(k) \leq 2T(k-2)$.
6. There are two adjacent vertices u and v of degree 2 each. Let u_1 be the other neighbor of u and v_1 be the other neighbor of v ($u_1 \neq v_1$ due to bipartiteness). As we ruled out connected components that are simple cycles, we may assume that the degree of u_1 is at least 3. We claim that there exists a maximum induced matching which contains one of the edges uu_1, uv or vv_1. Indeed, consider any induced matching M. Then:
   - if neither u_1 nor v_1 is an endpoint of an edge in M, we may add uv to M, preserving independence and increasing size;
   - if, say, u_1 is the endpoint of some edge u_1x, but none of the three aforementioned edges belong to M, we may remove u_1x from M and add u_1u instead, thus preserving size and keeping independence.

   Thus we consider three cases (each case consisting of choosing one of the three edges). In each case we remove the neighbors of two adjacent vertices, each of degree at least two, thus we remove at least 4 vertices in each case, at least 2 of them are not used in the induced matching. Moreover, as u_1 has degree at least 3, in the case when we choose u_1u, we remove at least 5 vertices. We get the inequality $T(k) \leq 2T(k-2) + T(k-3)$.
7. The remaining case is when we have a vertex v of degree $d \geq 2$ and all its neighbors have degree at least 3. We branch into $d+1$ cases. In the first case we drop v. In the other d cases we choose one edge incident with v, say vu, and drop all neighbors of v and u. We remove at least

$d+3$ vertices from the graph, and out of them two are part of the induced matching. Thus we obtain the inequality:

$$T(k) \leq T(k-1) + d \cdot T(k-d-1) \quad \text{for} \quad d \geq 2.$$

Now we check if $T(k) = C\alpha^k$ satisfies this inequality for all $d \geq 2$. However, note that $\alpha > \frac{d+1}{d}$ for $d \geq 2$, so we only need to check the case $d = 2$, which holds by straightforward calculations. The case $d = 2$ is the only tight recurrence in the whole algorithm.

Theorem 7. *The* Bipartite Induced Matching *problem can be solved in time bounded by* $O(1.69563^k \text{poly}(n))$. *The* Co-MaxIR *problem can be solved in time bounded by* $O(2.8752^k \text{poly}(n))$. *Using the win-win approach, the upper irredundance number of a graph can be computed by an* $O^*(1.9369^n)$ *time algorithm. All aforementioned three algorithms use polynomial space.*

We mention the following result shown in [2] obtained by using memoization on the exact algorithm presented there:

Theorem 8. *The upper irredundance number of a graph can be computed in* $O^*(1.8475^n)$ *time, using exponential space.*

Conclusions

We have presented three approaches to break the 2^n-barrier for irredundance problems:

- The first one (Sec. 3) is purely enumerative, inspired by work of Björklund et al. [31]. This approach only works for the graph parameter $\text{ir}(G)$.
- The second one (Sec. 5) is a uniform approach to solve both variants of irredundance problems using a parameterized route to exact problems, employing Measure & Conquer techniques.
- The third one (Sec. 6) combines the parameterized approach with the idea of transforming the problem to another one that is somehow easier to handle. This approach only works for the graph parameter $\text{IR}(G)$.

It would be interesting to see whether these approaches are useful for other problems as well. Some of the vertex partitioning parameters discussed in [38] seem to be appropriate. Also, related problems like that of Open Irredundance (see [20]) should be amenable to the proposed procedures.

References

[1] D. Binkele-Raible, L. Brankovic, H. Fernau, J. Kneis, D. Kratsch, A. Langer, M. Liedloff, P. Rossmanith, A parameterized route to exact puzzles: breaking the 2^n-barrier for irredundance, in: T. Calamoneri, J. Díaz (Eds.), Proceedings of the 7th Italian Conference on Algorithms and Complexity (CIAC), number 6078 in LNCS, Springer, 2010, pp. 311–322.

[2] M. Cygan, M. Pilipczuk, J. O. Wojtaszczyk, Irredundant set faster than $O^*(2^n)$, in: T. Calamoneri, J. Díaz (Eds.), Proceedings of the 7th Italian Conference on Algorithms and Complexity (CIAC), number 6078 in LNCS, Springer, 2010, pp. 288–298.

[3] F. Fomin, F. Grandoni, D. Kratsch, A measure & conquer approach for the analysis of exact algorithms, Journal of the ACM 56 (2009).

[4] J. M. M. van Rooij, J. Nederlof, T. C. van Dijk, Inclusion/exclusion meets measure and conquer, in: A. Fiat, P. Sanders (Eds.), Proceedings of the 17th European Symposium on Algorithms (ESA), volume 5757 of *LNCS*, Springer, 2009, pp. 554–565.

[5] G. J. Woeginger, Exact algorithms for NP-hard problems: A survey, in: Combinatorial Optimization — Eureka! You shrink!, volume 2570 of *LNCS*, Springer, 2003, pp. 185–207.

[6] F. V. Fomin, D. Kratsch, Exact Exponential Algorithms, Springer, 2010.

[7] O. Favaron, T. W. Haynes, S. T. Hedetniemi, M. A. Henning, D. J. Knisley, Total irredundance in graphs, Discrete Mathematics 256 (2002) 115–127.

[8] R. B. Allan, R. Laskar, On domination and independent domination numbers of a graph, Discrete Mathematics 23 (1978) 73–76.

[9] B. Bollobás, E. J. Cockayne, Graph-theoretic parameters concerning domination, independence, and irredundance, Journal of Graph Theory 3 (1979) 241–250.

[10] B. Bollobás, E. J. Cockayne, On the irredundance number and maximum degree of a graph, Discrete Mathematics 49 (1984) 197–199.

[11] M.-S. Chang, P. Nagavamsi, C. P. Rangan, Weighted irredundance of interval graphs, Information Processing Letters 66 (1998) 65–70.

[12] E. J. Cockayne, P. J. P. Grobler, S. T. Hedetniemi, A. A. McRae, What makes an irredundant set maximal?, Journal of Combinatorial Mathematics and Combinatorial Computing 25 (1997) 213–224.

[13] E. J. Cockayne, C. M. Mynhardt, Irredundance and maximum degree in graphs, Combinatorics, Probability and Computing 6 (1997) 153–157.

[14] O. Favaron, Two relations between the parameters of independence and irredundance, Discrete Mathematics 70 (1988) 17–20.

[15] O. Favaron, A note on the irredundance number after vertex deletion, Discrete Mathematics 121 (1993) 51–54.

[16] M. R. Fellows, G. Fricke, S. T. Hedetniemi, D. P. Jacobs, The private neighbor cube, SIAM Journal on Discrete Mathematics 7 (1994) 41–47.

[17] S. T. Hedetniemi, R. Laskar, J. Pfaff, Irredundance in graphs: a survey, Congressus Numerantium 48 (1985) 183–193.

[18] R. Laskar, J. Pfaff, Domination and irredundance in graphs, Technical Report 434, Clemson Univ., Dept. of Math. SC., 1983.

[19] E. J. Cockayne, S. T. Hedetniemi, D. J. Miller, Properties of hereditary hypergraphs and middle graphs, Canadian Mathematical Bulletin 21 (1978) 461–468.

[20] C. J. Colbourn, A. Proskurowski, Concurrent transmissions in broadcast networks, in: J. Paredaens (Ed.), Proceedings of the 11th International Colloquium on Automata, Languages, and Programming (ICALP), volume 172 of *LNCS*, Springer, 1984, pp. 128–136.

[21] E. Cockayne, O. Favaron, C. Payan, A. Thomason, Contributions to the theory of domination, independence and irredundance in graphs, Discrete Mathematics 33 (1981) 249–258.

[22] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Fourier meets Möbius: fast subset convolution, in: Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), pp. 67–74.

[23] A. Björklund, T. Husfeldt, M. Koivisto, Set partitioning via inclusion-exclusion, SIAM Journal on Computing 39 (2009) 546–563.

[24] J. Nederlof, Fast polynomial-space algorithms using Möbius inversion: Improving on Steiner tree and related problems, in: S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, W. Thomas (Eds.), Proceedings of the 36th International Colloquium on Automata, Languages, and Programming (ICALP), volume 5555 of *LNCS*, Springer, 2009, pp. 713–725.

[25] F. V. Fomin, K. Iwama, D. Kratsch, P. Kaski, M. Koivisto, L. Kowalik, Y. Okamoto, J. van Rooij, R. Williams, 08431 Open problems – Moderately exponential time algorithms, in: Moderately Exponential Time Algorithms, number 08431 in Dagstuhl Seminar Proceedings.

[26] R. G. Downey, M. R. Fellows, V. Raman, The complexity of irredundant sets parameterized by size, Discrete Applied Mathematics 100 (2000) 155–167.

[27] R. G. Downey, M. R. Fellows, Parameterized Complexity, Springer, 1999.

[28] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, 2006.

[29] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.

[30] V. Raman, S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments, Theoretical Computer Science 351 (2006) 446–458.

[31] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, The Travelling Salesman Problem in bounded degree graphs, in: L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, I. Walukiewicz (Eds.), Proceedings of the 35th International Colloquium on Automata, Languages, and Programming (ICALP), volume 5125 of *LNCS*, Springer, 2008, pp. 198–209.

[32] B. Chor, M. Fellows, D. Juedes, Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps, in: J. Hromkovic, et al. (Eds.), Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), number 3353 in LNCS, Springer, 2004, pp. 257–269.

[33] M. R. Fellows, Blow-ups, win/win's, and crown rules: Some new directions in FPT, in: H. L. Bodlaender (Ed.), Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), number 2880 in LNCS, Springer, 2003, pp. 1–12.

[34] O. Ore, Theory of Graphs, volume XXXVIII of *Colloquium Publications*, American Mathematical Society, 1962.

[35] D. Binkele-Raible, Amortized Analysis of Exponential Time- and Parameterized Algorithms: Measure & Conquer and Reference Search Trees, Ph.D. thesis, Abteilung Informatik – Wirtschaftsinformatik, 2010.

[36] F. Abu-Khzam, H. Fernau, Kernels: annotated, proper and induced, in: H. L. Bodlaender, M. Langston (Eds.), International Workshop on Parameterized and Exact Computation IWPEC, volume 4169 of *LNCS*, Springer, 2006, pp. 264–275.

[37] H. Moser, S. Sikdar, The parameterized complexity of the induced matching problem, Discrete Applied Mathematics 157 (2009) 715–727.

[38] J. A. Telle, Vertex Partitioning Problems: Characterization, Complexity and Algorithms on Partial k-Trees, Ph.D. thesis, Department of Computer Science, University of Oregon, USA, 1994.