A Parameterized Route to Exact Puzzles: Breaking the 2^n-Barrier for Irredundance

(Extended Abstract)

Daniel Binkele-Raible[1], Ljiljana Brankovic[2], Henning Fernau[1], Joachim Kneis[3], Dieter Kratsch[4], Alexander Langer[3], Mathieu Liedloff[5], and Peter Rossmanith[3]

[1] Universität Trier, FB 4—Abteilung Informatik, D-54286 Trier, Germany. {fernau,raible}@uni-trier.de
[2] The University of Newcastle, University Drive, NSW 2308 Callaghan, Australia.⋆ ljiljana.brankovic@newcastle.edu.au
[3] Department of Computer Science, RWTH Aachen University, Germany.⋆⋆ {kneis,langer,rossmani}@cs.rwth-aachen.de
[4] Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. kratsch@univ-metz.fr
[5] Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, 45067 Orléans Cedex 2, France. liedloff@univ-orleans.fr

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 domination and independence numbers and have numerous relations to other graph parameters. It is a long-standing open question whether determining these numbers for a graph G on n vertices admits exact algorithms running in time less than the trivial $\Omega(2^n)$ enumeration barrier. We solve this open problem by devising parameterized algorithms for the duals of the natural parameterizations of the problems with running times faster than $\mathcal{O}^*(4^k)$. For example, we present an algorithm running in time $\mathcal{O}^*(3.069^k)$ for determining whether $\mathrm{IR}(G)$ is at least $n - k$. Although the corresponding problem has been shown to be in FPT by kernelization techniques, this paper offers the first parameterized algorithms with an exponential dependency on the parameter in the running time. Furthermore, these seem to be the first examples of a parameterized approach leading to a solution to a problem in exponential time algorithmics where the natural interpretation as exact exponential-time algorithms fails.

1 Introduction

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 $G[I]$, the subgraph induced by 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 *maximal* if no proper superset of I is an irredundant set. The

⋆ Supported by the RGC CEF grant G0189479 of The University of Newcastle.
⋆⋆ Supported by the DFG under grant RO 927/7.

lower irredundance number ir(G) equals the minimum cardinality taken over all maximal irredundant sets of G; similarly, the upper irredundance number IR(G) equals the maximum cardinality taken over all such sets.

In graph theory, the irredundance numbers have been extensively studied due to their relation to numerous other graph parameters. An estimated 100 research papers [1] have been published on the properties of irredundant sets in graphs, e.g., [2–8]. For example, a set is minimal dominating if and only if it is irredundant and dominating [9]. 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. Here, as usual, $\gamma(G)$ denotes the size of a minimum dominating set, and $\alpha(G)$ denotes the size of a maximum independent set in G. It is also known that $\gamma(G)/2 < \mathrm{ir}(G) \leq \gamma(G) \leq 2 \cdot \mathrm{ir}(G) - 1$, see [10].

There are also some applications of irredundant sets in combinatorial optimization, e.g., locating senders in broadcast and packet radio networks [11]. Determining the irredundance numbers is NP-hard even for bipartite graphs [5]. The fastest currently known exact algorithm is the simple $\mathcal{O}^*(2^n)$ brute-force approach enumerating all subsets.[6] So it is tempting to study these problems from a parameterized complexity viewpoint (for an introduction, see, e.g., [12]). It has been known for a while (see, e.g., [13]) that it is possible to break the so-called 2^n*-barrier* for (some) vertex-selection problems by designing parameterized algorithms that run in time $\mathcal{O}^*(c^k)$ for some $c < 4$ by a "win-win" approach: either the parameter is "small" ($k < n/2 + \epsilon$ for an appropriate $\epsilon > 0$) and we use the parameterized algorithm, or we enumerate all $\binom{n}{n/2-\epsilon} < 2^n$ subsets.

Unfortunately, the problem of finding an irredundant set of size k is $W[1]$-complete when parameterized in k as shown by Downey et al. [14]. However, they also proved that the parameterized dual, where the parameter is $k' := n - k$, admits a problem kernel of size $3k'^2$ and is therefore in FPT (but the running time has a superexponential dependency on the parameter). Therefore in this paper we study the parameterized problems (following the notation of [14]) Co-Maximum Irredundant Set (Co-MaxIR) and Co-Minimum Maximal Irredundant Set (Co-MinMaxIR), which given a graph $G = (V, E)$ and a positive integer parameter k are to decide whether, respectively, $\mathrm{IR}(G) \geq n - k$ and $\mathrm{ir}(G) \leq n - k$. We also consider the variant exact Co-Minimum Maximal Irredundant Set (exact Co-MinMaxIR), which given a graph $G = (V, E)$ and positive integer parameter k, asks to decide whether $\mathrm{ir}(G) = n - k$.

Our contribution. First, we present linear problem kernels with $2k-1$ vertices for the Co-MinMaxIR problem and $3k$ vertices for Co-MaxIR, which already shows that both problems can be solved with a running time of $\mathcal{O}^*(c^k)$, $c \leq 8$. In particular, this improves the kernel with $3k^2$ vertices and the corresponding running time of $O^*(8^{k^2})$ of [14]. Secondly, we present a simple algorithm with a running time of $\mathcal{O}^*(3.841^k)$ which solves both Co-MaxIR and exact Co-MinMaxIR simultaneously. The price we pay for this generality is that the running time is only slightly better than $O^*(4^k)$, since we cannot exploit any

[6] The $\mathcal{O}^*$-notation hides polynomial factors, e.g., $f(n, k) \cdot \mathrm{poly}(n, k)) = \mathcal{O}^*(f(n, k))$.

special properties of Co-MaxIR that do not hold for exact Co-MinMaxIR and vice versa. Thirdly, we present a modification of the above algorithm, which trades the generality for improved running time and solves Co-MaxIR in time $\mathcal{O}^*(3.069^k)$.

Although all the algorithms are surprisingly simple, a major effort is required to prove their running time using a non-standard measure and a Measure & Conquer (M&C) approach. While nowadays M&C is a standard technique for the analysis of moderately exponential time algorithms (see, e.g., [15]), it is still seldom used in parameterized algorithmics.

Finally, as a direct consequence, we obtain algorithms that compute the irredundance numbers in time $\mathcal{O}^*(1.99914^n)$ and even $\mathcal{O}^*(1.9601^n)$ in case of Co-MaxIR. These are the first exact exponential time algorithms breaking the trivial 2^n enumeration barrier for computing the irredundance numbers on arbitrary graphs with n vertices, a well-known open question (see, e.g., [16]). Recently, this has independently been achieved by a group consisting of M. Cygan, M. Pilipczuk and J. O. Wojtaszczyk [17].

Due to space limits, some proofs are omitted in this extended abstract.

2 Preliminaries and Linear Kernels

The following alternative definition of *irredundance* is more descriptive and eases understanding the results 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 garden, and all additional 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 fix one of these vertices as a unique garden and the other vertices 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 the 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*.

Our first theorem makes use of the inequality $\text{ir}(G) \leq \gamma(G) \leq n/2$ in graphs without isolated vertices for Co-MinMaxIR, and of crown reductions [18, 19] for Co-MaxIR. This already shows that Co-MinMaxIR and Co-MaxIR allow fixed-parameter tractable algorithms with a running time exponential in k, a new contribution. In contrast, computing $\text{IR}(G)$ is W[1]-complete [14].

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

3 A Simple Algorithm For Computing The Irredundance Numbers

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 is is easy to see that $|\mathcal{K}_e| = |\mathcal{K}\setminus\mathcal{G}| \leq k$ 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 $v \in \mathcal{W}$. Additionally, 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 all $v \in \mathcal{K}_i$, $N(v) \subseteq \mathcal{W}$,
3. for all $v \in \mathcal{K}_e$, $N(v) \cap (\mathcal{G}_e \cup \overline{V}) \neq \emptyset$,
4. for all $v \in \mathcal{K}_e$, $|N(v) \cap \mathcal{G}_e| \leq 1$,
5. for all $v \in \mathcal{G}_e$, $N(v) \cap (\mathcal{K}_e \cup \overline{V}) \neq \emptyset$,
6. for all $v \in \mathcal{G}_e$, $|N(v) \cap \mathcal{K}_e| \leq 1$.

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:

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

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 $v \in \mathcal{W}$, 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 $v \in \mathcal{K}_e$ with $N(v) \cap \mathcal{G}_e = \emptyset$ and $N(v) \cap \overline{V} = \{w\}$, then set $\mathcal{G}_e := \mathcal{G}_e \cup \{w\}$.
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 $\mathcal{K}_e := \mathcal{K}_e \cup \{w\}$.
R_4 For every $v \in$ Not$\mathcal{G} \cap$ 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. Since the algorithm uses exhaustive branching, we easily obtain:

Algorithm 1 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 $\mathcal{K}_e$, $\mathcal{K}_i$, $\mathcal{G}_e$, $\mathcal{W} \subseteq V$

01: Compute the sets Not$\mathcal{G}$, Not$\mathcal{K}$.
02: Apply the reduction rules exhaustively, updating Not$\mathcal{G}$ and Not$\mathcal{K}$.
03: if the current labeling is invalid then return NO.
04: if $\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) < 0$ then return NO.
05: if $|\mathcal{K}_e| + |\mathcal{W}| = k$ or $|\mathcal{G}_e| + |\mathcal{W}| = k$ or all vertices are labeled then
06:     return whether $V \setminus (W \cup \mathcal{G}_e)$ is a solution.
07: if Not$\mathcal{G} \neq \emptyset$ (or analogously, Not$\mathcal{K} \neq \emptyset$) then
08:     choose $v \in$ Not$\mathcal{G}$;
09:     return CO-IR($G, k, \mathcal{K}_e \cup \{v\}, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W}$) or CO-IR($G, k, \mathcal{K}_e, \mathcal{K}_i, \mathcal{G}_e, \mathcal{W} \cup \{v\}$)
10: 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.
11: 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($G, k, \mathcal{K}_e \cup \{u\}, \mathcal{K}_i, \mathcal{G}_e \cup \{v\}, \mathcal{W}$)
```

Lemma 1. *Algorithm 1 correctly solves* Co-MaxIR *(when initially called with empty labels).*

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

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. Since all recursive calls only require polynomial time, the running time of Algorithm 1 is bounded by $\mathcal{O}^*(T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}))$. Let our measure be:

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

Lemma 2. $T(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}) \leq \alpha^{\varphi(k, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W})}$ *with* $\alpha \leq 3.841$.

Theorem 2. Co-MaxIR *and* Exact Co-MinMaxIR *can be solved in time* $\mathcal{O}^*(3.841^k)$. *Consequently, the irredundance numbers of a graph* G *with* n *vertices can be computed in time* $\mathcal{O}^*(1.99914^n)$.

4 Measure & Conquer Tailored To The Problems

In this section, we tailor the general Algorithm 2 to the needs of the Co-MaxIR problem. To this end, we use a more precise annotation of vertices: In the course of the algorithm, they will be either unlabeled $\mathcal{U}$, kings with internal gardens

$\mathcal{K}_i$, kings with external gardens $\mathcal{K}_e$, (external) gardens $\mathcal{G}_e$, wilderness $\mathcal{W}$, not being kings Not$\mathcal{K}$, or not being gardens Not$\mathcal{G}$. We furthermore partition the set of vertices V into *active* vertices

$$V_a = \mathcal{U} \cup \mathrm{Not}\mathcal{G} \cup \mathrm{Not}\mathcal{K} \cup \{\, v \in \mathcal{K}_e \mid N(v) \cap \mathcal{G}_e = \emptyset \,\} \cup \{\, v \in \mathcal{G}_e \mid N(v) \cap \mathcal{K}_e = \emptyset \,\}$$

that have to be reconsidered, and *inactive* vertices $V_i = V \setminus V_a$. Now, a labeling is called *complete* if $V_a = \emptyset$. This means that the inactive vertices are either from $\mathcal{W}$, $\mathcal{K}_i$ or paired-up external kings and gardens. Define $\mathcal{K}_{ea} = \mathcal{K}_e \cap V_a$ and $\mathcal{K}_{ei} = \mathcal{K}_e \cap V_i$ (and analogously $\mathcal{G}_{ea}$, $\mathcal{G}_{ei}$).

We use a new measure $\varphi(k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a) = k - |\mathcal{W}| - |\mathcal{G}_{ei}| - \omega_\ell(|\mathcal{K}_{ea}| + |\mathcal{G}_{ea}|) - \omega_n(|\mathrm{Not}\mathcal{G}| + |\mathrm{Not}\mathcal{K}|)$, where Not$\mathcal{G}$ and Not$\mathcal{K}$ are taken into account. We will later determine the weights ω_ℓ and ω_n to optimize the analysis, where $0 \le \omega_n \le 0.5 \le \omega_\ell \le 1$ and $\omega_n + \omega_\ell \le 1$. We will describe in words how the measure changes in each case. Let us first present the reduction rules that we employ in Table 1. They shall be applied in the given order, one at a time. Note that the reduction rules are sound and do not increase the measure.

Lemma 3. *In a reduced instance, a vertex $v \in \mathrm{Not}\mathcal{K} \cup \mathrm{Not}\mathcal{G}$ has at most one neighbor $u \in \mathcal{G}_e \cup \mathcal{K}_e$; more precisely, if such u exists, then $u \in \mathcal{G}_e$ if and only if $v \in \mathrm{Not}\mathcal{G}$ and $\deg(v) \ge 2$ (Thus, v must have a neighbor z that is not in $\mathcal{G}_e \cup \mathcal{K}_e$).*

Proof. Consider, w.l.o.g., $v \in \mathrm{Not}\mathcal{G}$. Assume that $u \in N(v) \cap (\mathcal{G}_e \cup \mathcal{K}_e)$ exists. Note that the alternative $u \in \mathcal{K}_e$ is resolved by Reduction Rule 7. Hence, $u \in \mathcal{G}_e$. If v had no other neighbor but u, then Reduction Rule 12 would have triggered. So, $\deg(v) \ge 2$. Let $z \in N(v) \setminus \{u\}$. If the claim were false, then $z \in \mathcal{G}_e \cup \mathcal{K}_e$. The case $z \in \mathcal{K}_e$ is ruled out by Reduction Rule 7. The case $z \in \mathcal{G}_e$ is dealt with by Reduction Rule 13. Hence, $z \notin \mathcal{G}_e \cup \mathcal{K}_e$. □

Now consider Algorithm 2.

Lemma 4. *In each labeled graph which is input of a recursive call of* CO-IR *there are no two neighbors u, v such that $u \in \mathcal{K}_{ea}$ and $v \in \mathcal{G}_{ea}$.*

Lemma 5. *Whenever our algorithm encounters a reduced instance, a vertex $v \in \mathcal{G}_e$ obeys $N(v) \subseteq \mathcal{U} \cup \mathrm{Not}\mathcal{G}$. Symmetrically, if $v \in \mathcal{K}_e$, then $N(v) \subseteq \mathcal{U} \cup \mathrm{Not}\mathcal{K}$.*

Although we are looking for a maximal irredundant set, we can likewise look for a *complete labeling* $L = ({\mathcal{K}_i}^L, {\mathcal{G}_{e_i}}^L, {\mathcal{K}_{e_i}}^L, \mathcal{W}^L)$ that partitions the whole vertex set $V = {\mathcal{K}_i}^L \uplus {\mathcal{G}_{e_i}}^L \uplus {\mathcal{K}_{e_i}}^L \uplus \mathcal{W}^L$ into internal kings, external kings and gardens, as well as wilderness. Having determined L, $I_L = {\mathcal{K}_{e_i}}^L \uplus {\mathcal{K}_i}^L$ should be an irredundant set, and conversely, to a given irredundant set I, one can compute in polynomial time a corresponding complete labeling. However, during the course of the algorithm, we deal with (incomplete) labelings $L = (\mathcal{K}_i, \mathcal{G}_e, \mathcal{K}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a)$, a tuple of subsets of V that also serve as input to our algorithm, preserving the invariant that $V = \mathcal{K}_i \uplus \mathcal{G}_e \uplus \mathcal{K}_e \uplus \mathrm{Not}\mathcal{G} \uplus \mathrm{Not}\mathcal{K} \uplus \mathcal{W} \uplus \mathcal{U}$. A complete labeling corresponds to a labeling with $\mathrm{Not}\mathcal{G} = \mathrm{Not}\mathcal{K} = \mathcal{U} = V_a = \emptyset$.

1. If V contains a vertex $x \in \mathcal{K}_i$ and a neighbor $u \in \mathcal{G}_e \cup \mathcal{K}_e \cup \mathcal{K}_i$, then return NO.
2. If V contains a vertex x with two neighbors u, v where $x \in \mathcal{K}_e$ and $u, v \in \mathcal{G}_e$, then return NO. Exchanging the roles of kings and gardens, we obtain a symmetric rule.
3. If V contains an isolated vertex $v \in (\mathcal{G}_e \cup \mathcal{K}_e)$, then return NO.
4. If V contains an isolated vertex $v \in (\mathrm{Not}\mathcal{K} \cup \mathrm{Not}\mathcal{G})$, then put v into $\mathcal{W}$, decreasing the measure by $1 - \omega_n$.
5. If V contains an isolated vertex $u \in \mathcal{U}$, then put u into $\mathcal{K}_i$ and set $V_a = V_a \setminus \{u\}$.
6. Delete an edge between two external kings or two external gardens.
7. Delete an edge between a $\mathcal{K}_e$- and a $\mathrm{Not}\mathcal{G}$-vertex. Exchanging the roles of kings and gardens, we obtain a symmetric rule.
8. Remove any edges incident to vertices in $\mathcal{W}$.
9. *a*) Delete an edge between two $\mathrm{Not}\mathcal{K}$-vertices.
   b) Delete an edge between two $\mathrm{Not}\mathcal{G}$-vertices.
10. If $u \in \mathcal{U}$ such that $N(u) = \{v\}$ for some $v \in \mathcal{U}$, then put u into $\mathcal{K}_i$ and set $V_a = V_a \setminus \{u\}$.
11. If $u \in \mathcal{K}_i$, then put its neighbors $N(u)$ into $\mathcal{W}$ and set $V_a = V_a \setminus N(u)$; this decreases the measure by $|N(u)|$.
12. If V contains two neighbors u, v such that $u \in \mathcal{G}_{e\,a}$ and $v \in \mathcal{U} \cup \mathrm{Not}\mathcal{G}$ with either $\deg(u) = 1$ or $\deg(v) = 1$, then put v into $\mathcal{K}_e$, and render both, u and v inactive; this decreases the measure by $1 - \omega_\ell$ if $v \in \mathcal{U}$ and otherwise by $1 - \omega_\ell - \omega_n$. Exchanging the roles of kings and gardens, we obtain a symmetric rule.
13. If V contains a vertex v with two neighboring gardens such that $v \in \mathcal{U}$, then set $v \in \mathrm{Not}\mathcal{K}$; if $v \in \mathrm{Not}\mathcal{G}$, then set $v \in \mathcal{W}$. This decreases the measure by ω_n or by $(1 - \omega_n)$, respectively. Exchanging the roles of kings and gardens, we obtain a symmetric rule.
14. Assume that V contains two inactive neighbors u, v where $u \in \mathcal{K}_e$ and $v \in \mathcal{G}_e$, then put all $x \in (N(u) \cap \mathcal{U})$ into $\mathrm{Not}\mathcal{G}$, all $x \in (N(u) \cap \mathrm{Not}\mathcal{K})$ into $\mathcal{W}$, all $x \in (N(v) \cap \mathcal{U})$ into $\mathrm{Not}\mathcal{K}$ and all $x \in (N(v) \cap \mathrm{Not}\mathcal{G})$ into $\mathcal{W}$. The measure decrease for each vertex x is ω_n if $x \in ((N(u) \cup N(v)) \cap \mathcal{U})$ and $1 - \omega_n$ if $x \in (N(u) \cap \mathrm{Not}\mathcal{K}) \cup (N(v) \cap \mathrm{Not}\mathcal{K})$.

Table 1. Extensive list of reduction rules.

Since $(\mathrm{Not}\mathcal{K} \cup \mathrm{Not}\mathcal{G}) \subseteq V_a$, we have obtained a complete labeling once we leave our algorithm in Line 4, returning YES. We say that a labeling $L' = (\mathcal{K}_i{}', \mathcal{G}_e{}', \mathcal{K}_e{}', \mathrm{Not}\mathcal{G}', \mathrm{Not}\mathcal{K}', \mathcal{W}', V_a')$ *extends* the labeling $L = (\mathcal{K}_i, \mathcal{G}_e, \mathcal{K}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a)$ if $\mathcal{K}_i \subseteq \mathcal{K}_i{}'$, $\mathcal{G}_e \subseteq \mathcal{G}_e{}'$, $\mathcal{K}_e \subseteq \mathcal{K}_e{}'$, $\mathrm{Not}\mathcal{G} \subseteq \mathcal{W}' \cup \mathcal{K}_e{}' \cup \mathrm{Not}\mathcal{G}'$, $\mathrm{Not}\mathcal{K} \subseteq \mathcal{W}' \cup \mathcal{G}_e{}' \cup \mathrm{Not}\mathcal{K}'$, $\mathcal{W} \subseteq W'$, $V_a' \subseteq V_a$. We also write $L \prec_G L'$ if L' extends L. Notice that reduction rules and recursive calls only extend labelings (further).

Notice that $\prec_G$ is a partial order on the set of labelings of a graph $G = (V, E)$. The maximal elements in this order are precisely the complete labelings. Hence, the labeling L_I corresponding to a maximal irredundant set I is maximal, with $\varphi(k, L_I) \leq 0$ iff $|I| \geq |V| - k$. Conversely, given a graph $G = (V, E)$, the labeling $L_G = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, V)$ is the smallest element of $\prec_G$; this is also the initial labeling that we start off with when first calling Algorithm 2. If L, L' are labelings corresponding to the parameter lists of nodes n, n' in the search tree such that n is ancestor of n' in the search tree, then $L \prec_G L'$. The basic strategy of

Algorithm 2 A faster algorithm for CO-MAXIR.

```
Algorithm CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a$):
Input: Graph $G = (V, E)$, $k \in \mathbf{N}$, labels $\mathcal{K}_i$, $\mathcal{K}_e$, $\mathcal{G}_e$, $\mathrm{Not}\mathcal{G}$, $\mathrm{Not}\mathcal{K}$, $\mathcal{W}$, $V_a \subseteq V$

01: Consecutively apply the procedure CO-IR to components containing $V_a$-vertices.
02: Apply all the reduction rules exhaustively.
03: if $\varphi(k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, V_a) < 0$ then return NO.
04: if $V_a = \emptyset$ then return YES.
05: if maxdegree($G[V_a]$) $\leq 2$ then solve by dynamic programming [20].
06: if $\mathrm{Not}\mathcal{G} \neq \emptyset$ (and analogously, $\mathrm{Not}\mathcal{K} \neq \emptyset$) then
07:     choose $v \in \mathrm{Not}\mathcal{G}$; if $\exists z \in N(v) \cap \mathcal{G}_e$ then $I := \{v, z\}$ else $I := \emptyset$.
08:     return CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathrm{Not}\mathcal{G} \setminus \{v\}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus I$) or
            CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G} \setminus \{v\}, \mathrm{Not}\mathcal{K}, \mathcal{W} \cup \{v\}, V_a \setminus \{v\}$);
09: if there is an unlabeled $v \in V$ with exactly two neighbors $u, w$ in $G[V_a]$,
        where $u \in \mathcal{G}_{ea}$ and $w \in \mathcal{K}_{ea}$ then
10:     return CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{v, u\}$) or
            CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e \cup \{v\}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{v, w\}$);
11: if $\mathcal{K}_{ea} \cup \mathcal{G}_{ea} \neq \emptyset$ then
12:     Choose some $v \in \mathcal{K}_{ea} \cup \mathcal{G}_{ea}$ of maximum degree.
13:     if $v \in \mathcal{K}_{ea}$ (and analogously, $v \in \mathcal{G}_{ea}$) then return whether
14:         $\exists u \in N(v)$ : CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e \cup \{u\}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{u, v\}$)
15: Choose $v \in \mathcal{U}$ of maximum degree, preferring $v$ with some $u \in N(v)$ of degree two.
16: return CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W} \cup \{v\}, V_a \setminus \{v\}$)
        or CO-IR($G, k, \mathcal{K}_i \cup \{v\}, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{v\}$)
        or $\exists u \in N(v)$ such that
            CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e \cup \{v\}, \mathcal{G}_e \cup \{u\}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{u, v\}$)
        or $\exists u \in N(v)$ such that
            CO-IR($G, k, \mathcal{K}_i, \mathcal{K}_e \cup \{u\}, \mathcal{G}_e \cup \{v\}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a \setminus \{u, v\}$)
```

Algorithm 2 is to exhaustively consider all complete labelings (only neglecting cases that cannot be optimal). This way, also all important maximal irredundant sets are considered.

Lemma 6. *If* $\varphi(k, \mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathcal{W}, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, V_a) < 0$*, then for weights* $0 \leq \omega_n \leq 0.5 \leq \omega_\ell \leq 1$ *with* $\omega_n + \omega_\ell \leq 1$*, for any complete labeling* $L = ({\mathcal{K}_i}^L, {\mathcal{G}_{ei}}^L, {\mathcal{K}_{ei}}^L, \mathcal{W}^L)$ *extending* $\Lambda := (\mathcal{K}_i, \mathcal{K}_e, \mathcal{G}_e, \mathrm{Not}\mathcal{G}, \mathrm{Not}\mathcal{K}, \mathcal{W}, V_a)$ *we have* $\varphi(k, {\mathcal{K}_i}^L, {\mathcal{G}_{ei}}^L, {\mathcal{K}_{ei}}^L, \emptyset, \emptyset, \mathcal{W}^L, \emptyset) < 0$.

1	$\mathcal{K}_{ea} \to {\mathcal{K}_{ei}}^L$	$-\omega_l$
2	$\mathcal{G}_{ea} \to {\mathcal{G}_{ei}}^L$	$1 - \omega_l$
3	$\mathrm{Not}\mathcal{G} \to {\mathcal{K}_{ei}}^L$	$-\omega_n$
4	$\mathrm{Not}\mathcal{G} \to \mathcal{W}^L$	$1 - \omega_n$
5	$\mathrm{Not}\mathcal{K} \to {\mathcal{G}_{ei}}^L$	$1 - \omega_n$
6	$\mathrm{Not}\mathcal{K} \to \mathcal{W}^L$	$1 - \omega_n$
7	$\mathcal{U} \to \mathcal{W}^L$	1
8	$\mathcal{U} \to {\mathcal{K}_i}^L$	0
9	$\mathcal{U} \to {\mathcal{K}_{ei}}^L$	0
10	$\mathcal{U} \to {\mathcal{G}_{ei}}^L$	1

Proof. We give a table for every possible label transition from Λ to its extension L. Note that Algorithm CO-IR only computes such solutions. All entries except two cause a non-increase of φ. The entries number 1 and 3 expose an increase in φ. By the problem definition, there exists a bijection $f : {\mathcal{K}_e}^L \to {\mathcal{G}_e}^L$. So for a vertex v in ${\mathcal{K}_{ei}}^L \cap \mathcal{K}_{ea}$ we must have $f(v) \in \mathcal{U} \cup \mathrm{Not}\mathcal{K}$. By Lemma 4 $f(v) \notin \mathcal{G}_{ea}$. Taking now into account the label transition of $f(v)$ which must be of the form $\mathcal{U} \to {\mathcal{G}_e}_i^L$ or Not$\mathcal{K} \to {\mathcal{G}_e}_i^L$, we see that a total decrease with respect to v and $f(v)$ of at least $1 - \omega_n - \omega_l \geq 0$ can be claimed. If $v \in$ Not$\mathcal{G} \cap {\mathcal{K}_e}_i^L$ then by arguing analogously we get a total decrease of at least $1 - \omega_n - \omega_l \geq 0$. $\square$

Lemma 7. *Assume that there is an unlabeled vertex v, which has exactly two neighbors $v_G \in \mathcal{G}_e$ and $v_K \in \mathcal{K}_e$. In the corresponding branching process, we may then omit the case $v \in \mathcal{W}$.*

Proof. We are looking for an inclusion-maximal irredundant set. Hence, only the positions of the kings matter, not the positions of the gardens. So, in particular we cannot insist on the garden of v_K being placed on some neighbor u of v_K different from v. In this sense, any solution that uses v as wilderness can be transformed into a no worse solution with $v \in \mathcal{G}_e$: Simply pair up v and v_K, turning the hitherto garden of v_K into wilderness. So, no optimum solution is lost by omitting the case $v \in \mathcal{W}$ in the branching. $\square$

Lemma 8. *Algorithm 2 correctly solves* Co-MaxIR.

Proof. The algorithm correctly answers NO in line 3 by Lemma 6. If $V_a = \emptyset$ in line 4 we can deduce $k \geq |\mathcal{W}| + |\mathcal{G}_{ei}|$. Thus, we can correctly answer YES. The recursive calls in lines 8, 14 and 16 are all exhaustive branchings and therefore no optimum solution is neglected. In the call in line 10 we do not consider the possibility of setting $v \in \mathcal{W}$ which is justified by Lemma 7. $\square$

Theorem 3. Co-MaxIR *can be solved in time $\mathcal{O}^*(3.069^k)$, and consequently in time $\mathcal{O}^*(1.9601^n)$.*

Proof. The correctness of the algorithm has been reasoned above already (in particular, Lemma 6 concerning the correctness of the abort condition). For the running time, we now provide a partial analysis leading to recurrences that estimate an upper bound on the search tree size $T_\varphi(\mu, h)$, where μ denotes the measure and h the height of the search tree. The claimed running time would then formally follow by an induction over h.

1. Assume that the algorithm branches on some vertex $v \in$ Not$\mathcal{G}$, the case $v \in$ Not$\mathcal{K}$ being completely analogous. By reduction rules, $N(v) \subseteq \mathcal{U} \cup \mathcal{G}_{ea} \cup$ Not$\mathcal{K}$.

a) If $N(v) \subseteq \mathcal{U} \cup$ Not$\mathcal{K}$, we derive the following branch in the worst case: $T_\varphi(\mu, h) \leq T_\varphi(\mu - (1 - \omega_n), h - 1) + T_\varphi(\mu - (\omega_\ell - \omega_n), h - 1)$. This follows from a simple branching analysis considering the cases that v becomes wilderness or that v becomes a king. See also Table 2 where the entries $(1a)\#1$ and $(1a)\#2$ correspond to the reduction of the first and second branch.

b) Assume now that $N(v) \cap \mathcal{G}_{ea} \neq \emptyset$ and let $u \in N(v) \cap \mathcal{G}_{ea}$. Lemma 3 ensures that there can be at most one element in $N(v) \cap \mathcal{G}_e$ and that $d(v) \geq 2$. Due to Reduction Rule 12, $\deg(u) \geq 2$.

First assume that $\deg(u) = 2$, i.e., $N(u) = \{v, x\}$. Then, we arrive at the following recursion: $T_\varphi(\mu, h) \leq T_\varphi(\mu - (2 - \omega_\ell - \omega_n), h - 1) + T_\varphi(\mu - (1 - \omega_\ell + \omega_n), h - 1)$, see entries $(1bi)\#1$ and $(1bi)\#2$ of Table 2. This is seen as follows. By

Weight	1	1	ω_ℓ	ω_ℓ	ω_n	ω_n	
Case	$\mathcal{W}$	$\mathcal{G}_{ei}$	$\mathcal{K}_{ea}$	$\mathcal{G}_{ea}$	Not$\mathcal{G}$	Not$\mathcal{K}$	potent. diff. $\geq$
$(1a)\#1$	v				$-v$		$1-\omega_n$
$(1a)\#2$			v		$-v$		$\omega_\ell-\omega_n$
$(1bi)\#1$	v	u		$-u$	$-v$		$2-\omega_n-\omega_\ell$
$(1bi)\#2$		u		$-u$	$-v,+z$	x	$1+\omega_n-\omega_\ell$
$(1bii)\#1$	v				$-v$		$1-\omega_n$
$(1bii)\#2$		u		$-u$	$-v,+z$	$\{x_1,x_2\}$	$1+2\omega_n-\omega_\ell$
$(2a)\#1$		v_G		$-v_G$	$N(v_G)\setminus\mathcal{G}_e$		$1-\omega_\ell+2\omega_n$
$(2a)\#2$		v	$-v_K$			$N(v_K)\setminus\mathcal{K}_e$	$1-\omega_\ell+2\omega_n$
$(2b)\#1$		$\{v_G,x\}$	$-v_K$	$-v_G$	$N(v_G)\setminus\{v\}$	$-x\in N(v_K)\setminus\{v\}$	$2-2\omega_\ell+\omega_n$
$(2b)\#2$		$\{v,v_G\}$	$-v_K$	$-v_G$	$-x\in N(v_G)\setminus\{v\}$	$N(v_K)\setminus\{v\}$	$2-2\omega_\ell+\omega_n$
$(2c)\#1$		$\{v_G,x\}$	$-v_K$	$-v_G$	$N(v_G)\setminus\{v\}$	$-x\in N(v_K)\setminus\{v\}$	$2-2\omega_\ell+\omega_n$
$(2c)\#2$		v	$-v_K$			$N(v_K)\setminus\mathcal{K}_e$	$1-\omega_\ell+\omega_n$
$(3)\#j$		v		$-v$	$N(u)\setminus\{v\}$	$N(v)\setminus\{u\}$	$1-\omega_\ell+\deg(v)\omega_n$
$(4a)\#1$	$N(w)$						2
$(4a)\#2$	$N(v)$						$\deg(v)$
$(4a)\#j$		u			$N(v)\setminus\{u\}$	$N(u)\setminus\{v\}$	$1+\deg(v)\omega_n$
$(4a)\#j$		v			$N(u)\setminus\{v\}$	$N(v)\setminus\{u\}$	$1+\deg(v)\omega_n$
$(4b)\#1$	v						1
$(4b)\#2$	$N(v)$						$\deg(v)$
$(4b)\#j$		u			$N(v)\setminus\{u\}$	$N(u)\setminus\{v\}$	$1+(\deg(v)+1)\omega_n$
$(4b)\#j$		v			$N(u)\setminus\{v\}$	$N(v)\setminus\{u\}$	$1+(\deg(v)+1)\omega_n$

Table 2. Overview over the potential gained in different branchings; symmetric branches due to exchanging roles of kings and gardens are not displayed. Neither are possibly better branches listed.

setting $v \in \mathcal{W}$, due to Reduction Rule 8, u will be of degree one and hence will be paired with its neighbor x due to Reduction Rule 12. If $x \in \mathcal{U}$, the measure decreases by $2-\omega_l-\omega_n$. If $x \in \text{Not}\mathcal{G}$, it decreases by $2-\omega_l-2\omega_n$. But then by Lemma 3 there is $y \in N(x)\setminus\{u\}$ such that $y \in \text{Not}\mathcal{K}\cup\mathcal{U}$. Then by Reduction Rule 12 y is moved to $\mathcal{W}\cup\text{Not}\mathcal{G}$ giving some additional amount of at least ω_n. Note that $y \neq v$ as $v \in \text{Not}\mathcal{G}$.
If we set $v \in \mathcal{K}_e$, then u and v will be paired by Reduction Rule 14. Thereafter, the other neighbor x of u will become a member of Not$\mathcal{G}$ or of $\mathcal{W}$, depending on its previous status. Moreover, there must be a further neighbor $z \in \mathcal{U}\cup\text{Not}\mathcal{K}$ of v (by Lemma 3 and the fact that u is the unique $\mathcal{G}_{ea}$ neighbor) that will become member of Not$\mathcal{G}$ or $\mathcal{W}$. This yields the claimed measure change if $z \neq x$. If $z = x$, then z is in $\mathcal{U}$ and the vertex will be put into $\mathcal{W}$. Thus, the recursively considered instance has complexity $T_\varphi(\mu-(2-\omega_\ell-\omega_n), h-1) \leq T_\varphi(\mu-(1-\omega_\ell+\omega_n), h-1)$.

Secondly, assume that $\deg(u) \geq 3$ (keeping the previous scenario otherwise). This yields the following worst-case branch: $T_\varphi(\mu, h) \leq T_\varphi(\mu-(1-\omega_n), h-1) + T_\varphi(\mu-(1-\omega_\ell+2\omega_n), h-1)$, see entries $(1bii)\#1$ and $(1bii)\#2$ of Table 2.

This is seen by a similar (even simpler) analysis. Note that all $z \in N(v)\cap N(u) \subseteq \mathcal{U}$ get labeled $\mathcal{W}$ in the second branch.

We will henceforth not present the recurrences for the search tree size in this explicit form, but rather point to Table 2 that contains the same information. There, cases are differentiated by writing Bj for the jth branch.

2. Assume that all active vertices are in $\mathcal{U} \cup \mathcal{G}_e \cup \mathcal{K}_e$, with $\mathcal{G}_{ea} \cup \mathcal{K}_{ea} \neq \emptyset$. Then, the algorithm would pair up some $v \in \mathcal{G}_{ea} \cup \mathcal{K}_{ea}$. Assume that there is an unlabeled vertex v that has exactly two neighbors $v_G \in \mathcal{G}_e$ and $v_K \in \mathcal{K}_e$. Observe that we may skip the possibility that $v \in \mathcal{W}$ due to Lemma 7. Due to space constraints, we omit further details. For the resulting branching vectors, please see Table 2.

3. Assume that all active vertices are in $\mathcal{U} \cup \mathcal{G}_e \cup \mathcal{K}_e$, with $\mathcal{G}_{ea} \cup \mathcal{K}_{ea} \neq \emptyset$. Then, the algorithm tries to pair up some $v \in \mathcal{G}_{ea} \cup \mathcal{K}_{ea}$ of maximum degree. There are $\deg(v)$ branches for the cases labeled (3)#j. Since the two possibilities arising from $v \in \mathcal{G}_{ea} \cup \mathcal{K}_{ea}$ are completely symmetric, we focus on $v \in \mathcal{G}_{ea}$. First note that by Lemma 5 and the fact that Not$\mathcal{G} \cup$ Not$\mathcal{K} = \emptyset$ we have $N(v) \subseteq \mathcal{U}$. Exactly one neighbor u of $v \in \mathcal{G}_{ea}$ will be paired with v in each step, i.e., we set $u \in \mathcal{K}_e$. Pairing the king on u with the garden from v will inactivate both u and v. Then, reduction rules will label all other neighbors of v with Not$\mathcal{K}$ (they can no longer be kings), and symmetrically all other neighbors of u with Not$\mathcal{G}$. Note that $N(u) \setminus (\mathcal{K}_e \cup \{v\}) \neq \emptyset$, since otherwise a previous branching case or Reduction Rules 13 or 12 would have triggered. Thus, there must be some $q \in N(u) \cap \mathcal{U}$. From q, we obtain at least a measure decrease of ω_n, even if $q \in N(v)$. This results in a set of recursions depending on the degree of v as given in Table 2.

4. Finally, assume $V_a = \mathcal{U}$. Since an instance consisting of paths and cycles can be easily seen to be optimally solvable in polynomial time, we can assume that we can always find a vertex v of degree at least three to branch at. There are two cases to be considered: (a) either v has a neighbor w of degree two or (b) this is not the case. Details of the analysis are contained in Table 2.

Finally, to show the claimed parameterized running time, we set $\omega_\ell = 0.7455$ and $\omega_n = 0.2455$ in the recurrences. If the measure drops below zero, then we argue that we can safely answer NO, see Lemma 6. The exponential running time is a straightforward consequence using the aforementioned "win-win" approach. □

5 Conclusions

We presented a parameterized route to the solution of yet unsolved questions in exact algorithms. More specifically, we obtained algorithms for computing the irredundance numbers running in time less than $\mathcal{O}^*(2^n)$ by devising appropriate parameterized algorithms (where the parameterization is via a bound k on the co-irredundant set) running in time less than $\mathcal{O}^*(4^k)$.

It would be interesting to see this approach used for other problems, as well. Some of the vertex partitioning parameters discussed in [21] seem to be appropriate. We believe that the M&C approach could also be useful to find better algorithms for computing the lower irredundance number.

References

1. Favaron, O., Haynes, T.W., Hedetniemi, S.T., Henning, M.A., Knisley, D.J.: Total irredundance in graphs. Discrete Mathematics **256**(1-2) (2002) 115–127
2. Allan, R.B., Laskar, R.: On domination and independent domination numbers of a graph. Discrete Mathematics **23**(2) (1978) 73–76
3. Favaron, O.: Two relations between the parameters of independence and irredundance. Discrete Mathematics **70**(1) (1988) 17–20
4. Fellows, M.R., Fricke, G., Hedetniemi, S.T., Jacobs, D.P.: The private neighbor cube. SIAM J. Discrete Math. **7**(1) (1994) 41–47
5. Hedetniemi, S.T., Laskar, R., Pfaff, J.: Irredundance in graphs: a survey. Congr. Numer. **48** (1985) 183–193
6. Bollobás, B., Cockayne, E.J.: Graph-theoretic parameters concerning domination, independence, and irredundance. J. Graph Theory **3** (1979) 241–250
7. Cockayne, E.J., Grobler, P.J.P., Hedetniemi, S.T., McRae, A.A.: What makes an irredundant set maximal? J. Combin. Math. Combin. Comput. **25** (1997) 213–224
8. Chang, M.S., Nagavamsi, P., Rangan, C.P.: Weighted irredundance of interval graphs. Information Processing Letters **66** (1998) 65–70
9. Cockayne, E.J., Hedetniemi, S.T., Miller, D.J.: Properties of hereditary hypergraphs and middle graphs. Canad. Math. Bull. **21**(4) (1978) 461–468
10. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of Domination in Graphs. Volume 208 of Monographs and Textbooks in Pure and Applied Mathematics. Marcel Dekker (1998)
11. Colbourn, C.J., Proskurowski, A.: Concurrent transmissions in broadcast networks. In: Proc. of 11th ICALP. Number 172 in LNCS (1984) 128–136
12. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
13. Raman, V., Saurabh, S.: Parameterized algorithms for feedback set problems and their duals in tournaments. Theoretical Computer Science **351**(3) (2006) 446–458
14. Downey, R.G., Fellows, M.R., Raman, V.: The complexity of irredundant sets parameterized by size. Discrete Applied Mathematics **100** (2000) 155–167
15. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. Journal of the ACM **56**(5) (2009)
16. Fomin, F.V., Iwama, K., Kratsch, D., Kaski, P., Koivisto, M., Kowalik, L., Okamoto, Y., van Rooij, J., Williams, R.: 08431 Open problems – Moderately exponential time algorithms. In: Moderately Exponential Time Algorithms. Number 08431 in Dagstuhl Seminar Proceedings (2008)
17. Cygan, M., Pilipczuk, M., Wojtaszczyk, J.O.: Irredundant set faster than $O(2^n)$. In: these proceedings
18. Chor, B., Fellows, M., Juedes, D.: Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In: Proc. of 30th WG. Number 3353 in LNCS, Springer (2004) 257–269
19. Fellows, M.R.: Blow-ups, win/win's, and crown rules: Some new directions in FPT. In: Proc. of 29th WG. Number 2880 in LNCS, Springer (2003) 1–12
20. Telle, J.A.: Complexity of domination-type problems in graphs. Nordic J. of Comp. **1** (1994) 157–171
21. Telle, J.A.: Vertex Partitioning Problems: Characterization, Complexity and Algorithms on Partial k-Trees. PhD thesis, Department of Computer Science, University of Oregon, USA (1994)