Data Reductions and Combinatorial Bounds for Improved Approximation Algorithms

Faisal N. Abu-Khzam[a], Cristina Bazgan[b], Morgan Chopin[c], Henning Fernau[d,*]

[a]*Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon*
[b]*PSL, University of Paris-Dauphine, LAMSADE UMR 7243, France; and Institut Universitaire de France*
[c]*Université Pierre et Marie Curie, LIP6 UMR 7606, France*
[d]*Fachbereich 4, Abteilung Informatikwissenschaften, Universität Trier, Germany*

Abstract

Kernelization algorithms in the context of Parameterized Complexity are often based on a combination of data reduction rules and combinatorial insights. We will expose in this paper a similar strategy for obtaining polynomial-time approximation algorithms. Our method features the use of approximation-preserving reductions, akin to the notion of parameterized reductions. We exemplify this method to obtain the currently best approximation algorithms for Harmless Set, Differential and Multiple Nonblocker, all of them can be considered in the context of securing networks or information propagation.

Keywords: Reduction rules, maximization problems, polynomial-time approximation, domination problems

1. Introduction

It is well-known that most interesting combinatorial problems are hard from a computational point of view. More technically speaking, they mostly turn out to be NP-hard. As many of these combinatorial problems have some importance for practical applications, several techniques have been developed to deal with them. From a more mathematical angle, the two most interesting and widespread approaches are (polynomial-time) approximation and fixed-parameter algorithms. Both areas have developed their own set of tools over the years. For instance, methods related to Linear Programming are prominent in the area of Approximation Algorithms [4]. Conversely, data reduction rules are the method of choice to obtain kernelization results, which is central to Parameterized Algorithms [24]. Another essential ingredient to kernelization algorithms

*Corresponding author; Tel. +49 651 201 2827.
Email addresses: faisal.abukhzam@lau.edu.lb (Faisal N. Abu-Khzam), bazgan@lamsade.dauphine.fr (Cristina Bazgan), morgan.chopin@lip6.fr (Morgan Chopin), fernau@uni-trier.de (Henning Fernau)

is a collection of combinatorial insights to the specific problem, often (already) supplied by mathematicians working in Combinatorics. It is quite natural to try to employ certain tools from one area to the other one. For example, the title of the paper in [39] nicely indicates the intended use of Linear Programming to obtain FPT algorithms. In this paper, we take the opposite approach and show how to use data reduction rules and (constructive) combinatorial insights to obtain approximation algorithms, in particular for maximization problems. Notice that data reduction rules are often used in heuristic approaches, well-established in practical implementations. So, our approach also brings the often more theoretical findings closer to practice.

For the purpose of illustrating our method, we will mainly deal with maximization problems that are obtained from domination-type graph problems. Notice that this set of problems already delivers a wealth of computationally hard problems with interesting related combinatorial questions, as testified by the books [31, 32]. We first describe these problems, using standard graph-theoretic terminology.

Let $G = (V, E)$ be an undirected graph and $D \subseteq V$.

1. D is called a *dominating set* if, for all $x \in V \setminus D$, there is a $y \in D \cap N(x)$. $V \setminus D$ is known as an *enclaveless set* [44] or as a *nonblocker set* [23].
2. D is called a *total dominating set* if, for all $x \in V$, there is a $y \in D \cap N(x)$. $V \setminus D$ has been introduced as a *harmless set* or *robust set* (with unaminity thresholds) in [7].
3. If D can be partitioned as $D = D_1 \cup D_2$ such that, for all $x \in V \setminus D$, there is a $y \in D_2 \cap N(x)$, then (D_2, D_1) defines a *Roman domination function* $f_{D_1,D_2} : V \to \{0, 1, 2\}$ such that $f_{D_1,D_2}(V) = \Sigma_{w \in V} f_{D_1,D_2}(w) = 2|D_2| + |D_1|$. According to [11], $|V| - f_{D_1,D_2}(V)$ is also known as the *differential* of a graph (as introduced in [37]) if $f_{D_1,D_2}(V)$ is smallest possible.
4. If for all $x \in V \setminus D$, there are k elements in $D \cap N(x)$, then D is a *k-dominating set*, see [18, 22, 28]. We will call $V \setminus D$ a *k-nonblocker set*.

The maximization problems derived from these four definitions are: Nonblocker, Harmless Set, Differential, and k-Nonblocker. Actually, Nonblocker has been looked into by the approximation algorithm community quite a lot in recent years [2, 20, 40], where it is known as the Maximum Star Forest problem. Although these problems are all better known from the minimization perspective, there is a good reason to study them in this complementary way: All of these minimization problems do not possess constant-factor approximations under reasonable complexity assumptions (the reduction shown in [21] for (Total) Dominating Set starts from Set Cover), while the complementary problems can be treated in this favorable way. For Roman Domination, observe that the reduction shown in [27] works from Set Cover, so that again (basically) the same lower bounds follow. This move is related to *differential approximation* [3]. Notice that this comes along with similar properties from the perspective of Parameterized Complexity: While natural parameterizations of the minimizations lead to W[2]-hard problems [24, 27], the

natural parameterizations of the maximization counterparts are fixed-parameter tractable. However, as this is more customary as a combinatorial entity, let us refer (as usual) by $\gamma(G)$ to the size of the smallest dominating set of G, by $\gamma_t(G)$ to the size of the smallest total dominating set, by $\gamma_R(G)$ to the Roman domination number of G, i.e., the smallest value of a Roman domination function of G, and by $\gamma_k(G)$ to the size of the smallest k-dominating set of G.

Some graph-theoretic notations. Let $G = (V, E)$ be a simple undirected graph. We denote by $N(x)$ the set of neighbors of vertex x; the cardinality of $N(x)$ is the *degree* of x. The *closed neighborhood* of x is the set $N[x] = N(x) \cup \{x\}$. A vertex of degree zero is known as an *isolated vertex*, and a vertex of degree one as a *leaf*. The number of vertices of a graph is called its *order*. We say that $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given $U \subseteq V$, $G[U]$ denotes the subgraph of $G = (V, E)$ induced by U with vertex set U and edge set $\{xy \mid x, y \in U \wedge xy \in E\}$. A subgraph G' of G is called *induced* if there is a vertex subset U such that $G' = G[U]$. A repetition-free sequence $x_1, \ldots, x_k$ of vertices is a *path* in G (of *length* $k-1$) if $x_i x_{i+1} \in E$ for $i = 1, \ldots, k-1$. Then, we also say that the path $x_1, \ldots, x_k$ *connects* its *endpoints* x_1 and x_k. A graph is *connected* if between every pair of vertices, there is a path connecting them. A graph that is not connected is called *disconnected*. A connected subgraph $G' = (V', E')$ of $G = (V, E)$ with the property that a V'' with $V' \subseteq V'' \subseteq V$ that induces a connected subgraph of G satisfies $V' = V''$ is also known as a *(connected) component* of G. The *diameter* of a connected graph G is the greatest length of a shortest path in G.

We also employ some non-standard terminology on special paths that we introduce now. A *chain* is an induced path whose interior vertices are of degree two in G. A chain with one leaf endpoint is a *pendant chain*. A *floating chain* is a chain with two leaves. In other words, this is a path component. A *support vertex* is a non-leaf endpoint of a pendant chain. Support vertices may have more than one pendant chain.

Main Results. We introduce a notion of approximation-preserving reductions analogous to parameter-preserving reductions known in Parameterized Complexity in order to obtain new approximation algorithms. We introduce a general methodology to obtain constant-factor approximations for various problems. For instance, along with an algorithmic version of the upper bound obtained in [34] on the size of a total dominating set, we present a factor-two approximation algorithm for Harmless Set, beating the previously known factor of three [7]. Moreover, we are deriving a factor-$\frac{11}{3}$ approximation algorithm for Differential, which was set up as an open problem in [10], where this approximability question could be only settled for bounded-degree graphs; our approach also improves on the factor-4 approximation exhibited in [8]. However, as in [10] APX-completeness was shown for the degree-bounded case, nothing better than constant-factor approximations can be expected for general graphs. Finally, we present constant-factor approximation algorithms for k-Nonblocker.

Organization of the paper. Section 2 explains the use of reduction rules within maximization problems. It also exhibits the general method. Section 3 shows how to employ our general method to one specific problem in a non-trivial way. Sections 4 and 5 show that the same method can be also applied to other problems. We conclude with discussing further research directions.

2. Approximation preserving reductions for maximization problems

Specializing standard terminology from [4], we can express the following. A maximization problem $\mathcal{P}$ can be specified by a triple $(I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}})$, where

1. $I_{\mathcal{P}}$ is the set of input instances of $\mathcal{P}$;
2. $\mathrm{SOL}_{\mathcal{P}}$ is a function that associates to $x \in I_{\mathcal{P}}$ the set $\mathrm{SOL}_{\mathcal{P}}(x)$ of feasible solutions of x;
3. $m_{\mathcal{P}}$ provides on (x, y), where $x \in I_{\mathcal{P}}$ and $y \in \mathrm{SOL}_{\mathcal{P}}(x)$, a positive integer which is the value of the solution y.

An optimum solution y^* to x satisfies: (i) $y^* \in \mathrm{SOL}_{\mathcal{P}}(x)$, and (ii) $m_{\mathcal{P}}(y^*) = \max\{m_{\mathcal{P}}(y) \mid y \in \mathrm{SOL}_{\mathcal{P}}(x)\}$. The value $m_{\mathcal{P}}(y^*)$ is also referred to as $m^*_{\mathcal{P}}(x)$ for brevity.

Given a maximization problem $\mathcal{P}$, a *factor-α approximation*, $\alpha \geq 1$, associates to each $x \in I_{\mathcal{P}}$ some $y \in \mathrm{SOL}_{\mathcal{P}}(x)$ such that $\alpha \cdot m_{\mathcal{P}}(x, y) \geq m^*_{\mathcal{P}}(x)$. A solution $y \in \mathrm{SOL}_{\mathcal{P}}(x)$ satisfying $\alpha \cdot m_{\mathcal{P}}(x, y) \geq m^*_{\mathcal{P}}(x)$ is also called an *α-approximate solution* for x.

We are now going to present a first key notion for this paper.

Definition 1. Let $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}})$ be a maximization problem. An *α-preserving reduction*, with $\alpha \geq 1$, is a pair of mappings $\mathtt{inst}_{\mathcal{P}} : I_{\mathcal{P}} \to I_{\mathcal{P}}$ and $\mathtt{sol}_{\mathcal{P}}$ which, given $y' \in \mathrm{SOL}_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x))$, produces some $y \in \mathrm{SOL}_{\mathcal{P}}(x)$ such that there are constants $a, b \geq 0$ satisfying the following inequalities:

1. $a \leq \alpha \cdot b$,
2. $m^*_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x)) + a \geq m^*_{\mathcal{P}}(x)$, and
3. $\forall y' \in \mathrm{SOL}_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x)) : m_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x), y') + b \leq m_{\mathcal{P}}(x, \mathtt{sol}_{\mathcal{P}}(y'))$.

When referring to this definition, we mostly explicitly specify the constants a and b for ease of verification. An important trivial example is given by a pair of identity mappings that are α-preserving for any $\alpha \geq 1$. Notice that a similar notion has been introduced, or implicitly used, in the context of minimization problems in [16, 17, 29].

Let us illustrate this notion with the following small examples.

Example 1. ***Isolate Reduction for Nonblocker****. Delete an isolated vertex. This rule is α-preserving for any $\alpha \geq 1$ with $a = b = 0$. Namely, observe that no isolated vertex ever goes into a nonblocker set.*

For the following example, recall that Independent Set refers to the problem of finding the maximum set of vertices in a graph such that no two of them are adjacent.

Example 2. *We are going to discuss three rules.*

Isolate Reduction for Independent Set*. Delete an isolated vertex.*
This rule is α-preserving for any $\alpha \geq 1$ with $a = b = 1$. Namely, observe that an isolated vertex can always go into an independent set.

Leaf Reduction for Independent Set*. If G contains some leaf u with neighbor v, then reduce G to a graph G' by deleting u and v.*
This rule is α-preserving for any $\alpha \geq 1$ with $a = b = 1$. Namely, observe that the edge a leaf is incident to can be covered by v, never leading to worse results than taking u. Hence, u can go into the independent set without loss of generality.

Folding Reduction for Independent Set*. If G contains some chain $x-u-v$ where x and v are not neighbors. Then reduce G to a graph G' by deleting u and merging x and v.*

This rule was probably first shown (as a parameterized reduction rule for the related Vertex Cover *problem) in [25].*

This rule is α-preserving for any $\alpha \geq 1$ with $a = b = 1$.
Let us argue first why $b = 1$. If solution I' to the graph G' obtained from G by applying the Folding Reduction contains the vertex w obtained by merging x and v, then $I := (I' \setminus \{w\}) \cup \{x, v\}$ is an independent set in G. If $w \notin I'$, then $I := I' \cup \{u\}$ is an independent set of G.
To see why $a = 1$, consider some optimum solution (independent set) I of G. If $\{x, v\} \subseteq I$ or $\{x, v\} \cap I = \emptyset$ (and hence $u \in I$, as I is optimum), then we could easily get an independent set I' with $|I'| = |I| - 1$ by reversing the argument leading to conclude that $b = 1$. If, without loss of generality, $x \in I$ but $v \notin I$, then $(I \setminus \{x\}) \cup \{u\}$ is also an (optimum) independent set of G, and we can continue our argument as above.

Following the idea of the previous example, we can define a more elaborate rule for Nonblocker that can deal with certain vertices of degree two:

Example 3. ***Long Chain Reduction for Nonblocker****. If G contains some chain $x - u - v - y$ connecting x and y, then reduce G to a graph G' by deleting u, v, y and introducing edges from x to all vertices from $N_G(y)$ (if not already there).*

This rule is α-preserving for any $\alpha \geq 1$. This is testified by the pair of numbers $(2, 2)$. The proof that this claim is correct is straightforward yet non-trivial, as several cases have to be considered. Since we do not make use of this rule in the following, we omit the proof here and refer the reader to the proof of a similar statement for Harmless Set *in Theorem 15. The proof of correctness of the corresponding parameterized reduction can be found in [41]; in [23], this reduction rule is only mentioned as "The Degree Two Rule". Furthermore, we illustrate in Figure 1 the case when x belongs to the dominating set D that is considered, while y does not belong to D, showing that such a solution can be locally changed to another one D' that is not worse than D. In this picture, cycled nodes belong to D (or D', resp.), while boxed nodes do not belong to the*

corresponding nonblocker set. So, we see an exchange argument that is similar to what we explained with the Folding Reduction for Independent Set.

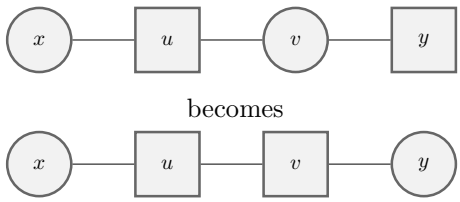

Figure 1: Dealing with $x \in D$, $y \notin D$.

Theorem 4. *Let $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}})$ be some maximization problem. If the pair $(\mathtt{inst}_{\mathcal{P}}, \mathtt{sol}_{\mathcal{P}})$ describes an α-preserving reduction and if, given some instance x, $y' \in \mathrm{SOL}_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x))$ is an α-approximate solution for $\mathtt{inst}_{\mathcal{P}}(x)$, then $y = \mathtt{sol}_{\mathcal{P}}(y')$ is an α-approximate solution for x.*

Proof. We have to prove that $\alpha \cdot m_{\mathcal{P}}(x, y) \geq m^*_{\mathcal{P}}(x)$. Now,

$$\frac{m^*_{\mathcal{P}}(x)}{m_{\mathcal{P}}(x,y)} \leq \frac{m^*_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x)) + a}{m_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x), y') + b} \leq \frac{\alpha m_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x), y') + \alpha b}{m_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x), y') + b} = \alpha$$

as required. □

This shows that an α-preserving reduction leads to a special AP-reduction as defined in [4]. But there, these reductions were mainly used to prove hardness results, as it is also the case of [29] that we already mentioned. However, we use this notion to obtain approximation algorithms.

The notion of an α-preserving reduction was coined following the successful example of kernelization reductions known from Parameterized Complexity [24]. One of the nice features of those is that they are usually compiled from simpler rules that are often based on some applicability conditions. In the following, we describe that this also works out for approximation. We need two further notions to make this precise.

We call an α-preserving reduction $(\mathtt{inst}_{\mathcal{P}}, \mathtt{sol}_{\mathcal{P}})$ *strict* if $|\,\mathtt{inst}_{\mathcal{P}}(x)| < |x|$ for all $x \in I_{\mathcal{P}}$, and it is called *polynomial-time computable* if the two mappings comprising the reduction can be computed in polynomial time.

The following lemma is relatively straightforward but technical to prove. Yet, it contains an important message: reduction rules can be composed so that the composition preserves certain properties.

Lemma 5. *If $(\mathtt{inst}_{\mathcal{P}}, \mathtt{sol}_{\mathcal{P}})$ and $(\mathtt{inst}'_{\mathcal{P}}, \mathtt{sol}'_{\mathcal{P}})$ are two α-preserving reductions , then the composition $(i, s) := (\mathtt{inst}_{\mathcal{P}} \circ \mathtt{inst}'_{\mathcal{P}}, \mathtt{sol}'_{\mathcal{P}} \circ \mathtt{sol}_{\mathcal{P}})$ is also an α-preserving reduction. If both $(\mathtt{inst}_{\mathcal{P}}, \mathtt{sol}_{\mathcal{P}})$ and $(\mathtt{inst}'_{\mathcal{P}}, \mathtt{sol}'_{\mathcal{P}})$ are strict (polynomial-time computable, resp.), then the composition (i, s) is strict (polynomial-time computable, resp.).*

PROOF. Consider a situation described as in the lemma, where the pair of numbers (a, b) shows that $(\mathtt{inst}_{\mathcal{P}}, \mathtt{sol}_{\mathcal{P}})$ is α-preserving and the pair of numbers (a', b') shows that $(\mathtt{inst}'_{\mathcal{P}}, \mathtt{sol}'_{\mathcal{P}})$ is α-preserving. Clearly, if $x \in I_{\mathcal{P}}$, then $\mathtt{inst}_{\mathcal{P}}(x) \in I_{\mathcal{P}}$ and hence $(\mathtt{inst}_{\mathcal{P}} \circ \mathtt{inst}'_{\mathcal{P}})(x) = \mathtt{inst}'_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x)) \in I_{\mathcal{P}}$, as well. A similar observation applies to the solutions (in the reversed order). We now prove that the composition (i, s) is α-preserving, testified by the pair of numbers $(a + a', b + b')$.

$$\begin{aligned} m^*_{\mathcal{P}}((\mathtt{inst}_{\mathcal{P}} \circ \mathtt{inst}'_{\mathcal{P}})(x)) + (a + a') &= (m^*_{\mathcal{P}}(\mathtt{inst}'_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x))) + a') + a \\ &\geq m^*_{\mathcal{P}}(\mathtt{inst}_{\mathcal{P}}(x)) + a \\ &\geq m^*_{\mathcal{P}}(x) \end{aligned}$$

The computation for the bounds on the solution is similar and hence omitted. The claims on composability of the strictness and the polynomial-time computability are easy to see. □

By a trivial induction argument, the previous lemma generalizes to any finite number of reductions that we like to compose. To continue Example 1, we could conclude that the following Generalized Isolate Reduction rule for Nonblocker, described as *Delete all isolated vertices*, is a strict, polynomial-time computable reduction which is α-preserving for any $\alpha \geq 1$. Namely, it can be implemented by:

As long as possible, apply the Isolate Reduction for Nonblocker.

Remark 1. Let us briefly discuss the case when we consider α-preserving reductions (as in our definition) with $a, b < 0$. Strictly speaking, such reductions do not satisfy Definition 1. However, Theorem 4 still holds true in this case. The reason why we excluded such reductions from our discussions in general is that one crucial argument would fail with such a notion: namely, such reductions cannot be processed until they are no longer applicable in a general loop, as (often) the instance will (at least potentially) expand and not shrink as it is the case with our definition, which means that such a loop might not terminate.

However, occasionally we will use such a reduction rule as the very last one (in a chain of rules that we apply). Due to the validity of Theorem 4, this still leads to a good approximation algorithm. We will call such a reduction (for clarity) a *weak α-preserving reduction*.

In actual fact, this peculiarity resembles the discussions in the area of Parameterized Complexity, or better said in the subarea of Kernelization, about the question whether a kernelization mapping should be allowed to increase the value of the parameter or not. If such an increase is allowed, again infinite loops are possible when arbitrarily composing kernelizations. Hence, for several technical results, non-increasing the parameter value is required. We will make use of the possibility of having a weak α-preserving reduction applied in the end only once, see Observation 34. The interested reader is invited to interpret the according reduction rules as data reduction rules in the parameterized sense. One can then observe that it is exactly this weak α-preserving rule that corresponds to a kernelization rule which increases the parameter.

Conditional reductions. In the realm of Kernelization, reductions are often described in some conditional form:

if condition **then do** action

Our previous considerations apply also for this type of conditioned reductions, apart from the fact that an instance may not change, assuming that the reduction was not applicable, which means that the condition was not true for that instance.

First, we have to make clear what the notions of "strictness" and "polynomial-time computations" refer to in the context of reduction rules with conditions. "Strictness" now means that the input will be shortened if the condition is met, and "polynomial time" means two things: a) the condition can be checked in polynomial time and b) the possibly triggered action can be performed in polynomial time. Moreover, often there is a finite collection of conditioned reductions. These can be combined in quite a natural way into a single conditioned reduction. This is formally described in the following lemma.

Lemma 6. *Assume that, for each* $1 \leq i \leq n$*,*

$$\textbf{\textit{if}}\ \textit{condition}_i\ \textbf{\textit{then do}}\ \textit{action}_i$$

is a conditioned α*-preserving reduction. Then, these can be combined into a single conditioned* α*-preserving reduction* $\langle \textit{combi-condition}, \textit{combi-action} \rangle$ *as follows:*

$$\textbf{\textit{if}}\ \exists i(\textit{condition}_i)\ \textbf{\textit{then do}}\ \textit{perform some applicable action}_i$$

If all original conditioned reductions are strict (polynomial-time computable, resp.), then the combined reduction is strict (polynomial-time computable, resp.).

Now, we can present a general recipe how to obtain a polynomial-time factor-α approximation based on α-preserving reductions. The previous lemma shows that the use of a single reduction in the formulation of the next theorem does not lose any generality.

Theorem 7. *Assume that* $\mathcal{P}$ *is some maximation problem. Suppose that*

$$\textbf{\textit{if}}\ \textit{condition}(x)\ \textbf{\textit{then do}}\ \textit{action}(x)$$

is some conditioned α*-preserving, strict, polynomial-time computable reduction. Further assume that there is some polynomial-time computable factor-*α *approximation algorithm* A *for* $\mathcal{P}$*, restricted to instances from* $\{x \in I_{\mathcal{P}} \mid \neg \textit{condition}(x)\}$*. Then, there is a polynomial-time computable factor-*α *approximation algorithm for all instances.*

Proof. The desired algorithm should work as follows. Given an instance x:

1. As long as possible, some α-preserving reductions are performed. This yields the sequence of instances $x = x_0, x_1, \ldots, x_n$.

2. Then, A is applied to the reduced instance $x' := x_n$.
3. As $y_n := y' := A(x')$ is an α-approximate solution for $x' = x_n$, we can successively construct α-approximate solutions y_{n-1} for x_{n-1}, $\ldots$, y_1 for x_1 and finally $y := y_0$ for $x = x_0$.
4. Return y as approximate solution for x.

As compositions of α-preserving reductions yield α-preserving reductions (maintaining some desirable properties), as shown in Lemma 5, any α-approximate solution to x_n can be turned into an α-approximate solution for x. Hence, all claimed properties directly follow by our previous considerations, apart from the polynomial-time claim. Here, observe as the reductions are strict, $n \leq |x|$, so that the loop in Step 1 terminates after a polynomial number of steps. □

The general strategy that we follow can be sketched as follows:

1. Apply (strict, poly-time computable) α-preserving reduction rules as long as possible.
2. Possibly modify the resulting graph so that it meets some requirements from known combinatorial results on the graph parameter of interest.
3. Compute some solution for the modified graph that satisfies the mentioned combinatorial bounds.
4. Construct from this solution a good approximate solution for the original instance.

In order to illustrate the use of this strategy, let us elaborate on Nonblocker, matching a result from [40].

Example 8. *Actually, conceptually this algorithm is even simpler than the one we present for* Harmless Set *in particular in the following sections. This goes along the lines of the kernelization result by Dehne* et al. *[23], but kernelization needs no constructive proof of the combinatorial backbone result; the non-constructive proof of [13, 38] is hence sufficient.*

1. *Delete all isolated vertices.*
   This is just the Generalized Isolate Reduction rule for Nonblocker discussed above. Hence, this rule is α-preserving for any $\alpha \geq 1$ (with $a = b = 0$).
   If the resulting graph is of minimum degree at least two, we are ready to directly apply the algorithm of Nguyen et al. *[40]. Otherwise, we continue as described in the following.*
2. *Merge all leaf neighbors into a single vertex.*
   Again, this rule is α-preserving for any $\alpha \geq 1$ (with $a = b = 0$ for a single merge and hence also for a finite sequence of merges).
3. *Delete all leaves but one, which is x.*
   Again, this rule is α-preserving for any $\alpha \geq 1$ (with $a = b = 1$ for a single deletion; hence it is also α-preserving for a finite sequence of deletions).
4. *Steps 1, 2 and 3 yield the graph G of order n_G.*

5. *Create a copy G' of the graph G; call the vertices in the new graph by priming the names of vertices of G. Let H be the graph union of G and G' plus the edge xx'. H is of minimum degree at least two by construction.*
6. *Take the algorithm of Nguyen* et al. *[40] to obtain a dominating set D_H of H satisfying $|D_H| \leq \frac{2}{5}n_H$. Should the solution D_H contain x or x', it is not hard to modify it to contain the leaf neighbors y or y', instead.*
7. *Hence, $D_G = V_G \cap D_H$ is a dominating set for G with $|D_G| \leq \frac{2}{5}n_G$. Trivially, $N_G = V_G \setminus D_G$ is a nonblocker solution for G that is $\frac{5}{3}$-approximate.*
8. *As the merging and deletion reductions are α-preserving for each $\alpha \geq 1$, we can safely undo them and hence obtain a $\frac{5}{3}$-approximate solution for the original graph instance.*

Better approximation algorithms for Nonblocker *have been obtained by Chen* et al. *[20] (with a factor of 1.41) and by Athanassopoulos* et al. *[2] (with a factor of 1.244). It would be interesting to learn of more proper combinatorial approaches that could improve on the factor 1.67 exhibited above. For instance, possibly the Long Chain Reduction for* Nonblocker *shown in Example 3 could be helpful, as better combinatorial bounds are known for nonblocker sets (dominating sets) in the case of graphs of minimum degree three; see [43]. Unforunately, also Reed's proof is non-constructive and would first have to be turned into a polynomial-time algorithm to make use of this result.*

Nguyen et al. *used a slightly different way to obtain their approximation algorithm. Let us reformulate and sketch the result from [40] within our framework. As a reduction rule, they only remove isolated vertices; these would be put into the dominating set anyways. The combinatorial result aimed at is the one exhibited by Blank [13] and (independently) by McCuaig and Shepherd [38] that shows that any graph (except for seven exceptional graphs) of order n with minimum degree of at least two has a dominating set with at most $\frac{2}{5}n$ vertices. This result is used by first modifying the graph by deleting all leaves and then interconnecting the leaf neighbors so that the minimum degree two requirement is met. It is then shown that it is possible to construct a nonblocker set for G, given a dominating set D_H satisfying $|D_H| \leq \frac{2}{5}n_H$ for the modified graph H. An essential ingredient is a new proof of the mentioned result from [38] that is in fact a polynomial-time algorithm to compute D_H within H.This result was also used by our version of this algorithm given above.*

3. Harmless Set

We are now turning towards Harmless Set as the most elaborate example of our methodology. First, we are going to present the combinatorial backbone of our result. Let $S_2(G)$ be the set all vertices of degree two within G. By courtesy to the authors, let us say that *G satisfies the Lam-Wei property* if it has minimum degree at least two and $G[S_2(G)]$ decomposes into K_1- and K_2-components. This property is equivalent to requiring that G has no isolates, no leaves and no three consecutive vertices of degree two.

Theorem 9. *(Lam and Wei [34]) Let G be a graph of order n_G that satisfies the Lam-Wei property. Then, $\gamma_t(G) \leq n_G/2$.*

The proof of this theorem is non-constructive, as it uses tools from extremal combinatorics. Now, we show how to obtain a polynomial-time algorithm that actually computes a total dominating set (TDS) D with $|D| \leq n_G/2$ under the assumptions of Theorem 9.

As connected components can be dealt with consecutively, we can assume that G is connected in the following discussion.

First, we greedily remove edges, as long as the graph still satisfies the Lam-Wei property. A TDS computed for the resulting graph is also a TDS for the original graph. For simplicity, we can hence further assume that no edges can be removed from G without violating the Lam-Wei property. This is a technical condition needed for applying some of the Lemmas from [34].

We now differentiate two main cases:

- If $S_2(G)$ is an independent set in G, i.e., if $G[S_2(G)]$ has no edges, then we have to differentiate further cases when the shortest cycle in G is of length 3, 4, 5, 6, or larger. In each of the cases, Lam and Wei show how to construct a graph G' smaller than G that also satisfies the Lam-Wei property.

- Otherwise, $G[S_2(G)]$ contains a K_2-component. Starting out from such a path of length one, the proof of [34, Lemma 6] shows how to construct a set Q of vertices such that the graph $G' = G[V \setminus Q]$ also satisfies the Lam-Wei property and, moreover, $\gamma_t(G) \leq \gamma_t(G') + \frac{|Q|}{2}$ is satisfied.

As some optimum TDS can be surely easily computed for small graphs, the sketched procedure allows to recursively compute some TDS solution for G. Notice in particular that the proofs of Lam and Wei show how to construct a solution for the calling instance from the one obtained for the called instance (in the recursion). Also, it is shown (as explicitly indicated in the second case above) that the claimed bound on the solution size easily follows by induction.

Hence, we can state the following constructive version of the combinatorial result of Lam and Wei:

Theorem 10. *For a given graph $G = (V, E)$ of order n_G that satisfies the Lam-Wei property, one can compute a TDS $D \subseteq V$ with $|D| \leq \frac{n_G}{2}$ in polynomial time.*

Observe that a quick analysis of the sketched algorithm indicates a bound of $O(n_G^8)$ for the running time, as one has to actually verify that there are no short cycles in G to match the case analysis. Supposedly, a complete re-analysis of the combinatorial argument could reveal better algorithms, but for the proof of concept of our methodology, this analysis is sufficient here.

Our approximation algorithm for Harmless Set is based on obtaining a (small enough) TDS in a graph H obtained from the input G after a number

of modifications (mainly vertex deletions). In the reduction from G to H, we distinguish between the number of deleted vertices d (to get from G to H) and the number of vertices a added to convert the TDS D_H to D_G.

Theorem 11. *Let G be a graph of order n_G and let H be a graph of order n_H obtained from G by deleting d vertices and possibly adding some edges. Let D_G and D_H be TDS solutions of G and H, respectively, such that $a = |D_G| - |D_H| \leq d$. If $|D_H| \leq c \cdot n_H$ for some $c < 1$ and if $d \leq \gamma_t(G)$, then $V(G) \setminus D_G$ is a harmless set of G whose size $n_G - |D_G|$ is within a factor of $(1-c)^{-1}$ from optimum.*

Proof. As $n_H = n_G - d$, $|D_G| = |D_H| + a \leq c(n_G - d) + a = cn_G + (a - cd) \leq cn_G + d - cd = cn_G + (1-c)d \leq cn_G + (1-c)\gamma_t(G)$. Hence, $n_G - |D_G| \geq n_G - cn_G - (1-c)\gamma_t(G) = (1-c)(n_G - \gamma_t(G))$. This immediately yields an approximation factor of $(1-c)^{-1}$. □

In the following section, we will present reduction rules that produce a graph G with the property (**) that each vertex of degree bigger than one has at most one leaf neighbor. The surgery that produces a graph H from G as indicated in Theorem 11 includes removing all d leaves and adding edges to ensure that H has minimum degree of two and satisfies that each component of $H[S_2(H)]$ has diameter at most one. Notice that all leaf neighbors in G belong to some optimum TDS of G without loss of generality. Due to (**), $\gamma_t(G) \geq d$ as required. Moreover, given some TDS solution D_H for H, we can produce a valid TDS solution D_G for G by adding all d leaf neighbors to D_H. Notice that Theorem 11 leads to a factor-2 approximation algorithm for Harmless Set based on Theorem 10.

In the following part of the section, we are going to describe the reduction rules necessary to produce a graph to which we could apply the mentioned combinatorial results.

Reduction Rules for Harmless Set

Now, we list α-preserving reductions for Harmless Set. We start with two very simple rules.

Isolate Reduction. If there is some isolated vertex, produce the instance $(\{x\}, \emptyset)$ that has trivially no solution. If there is some isolated edge xy, produce the instance $G[V \setminus \{x, y\}]$ from $G = (V, E)$.

Leaf Reduction. If there are two leaf vertices u, v with common neighbor w, then delete u. (It would go into the harmless set.)

Observation 12. *The two reduction rules are correct. The Isolate Reduction (for edges) and the Leaf Reduction are α-preserving for any $\alpha \geq 1$.*

Proof. For the correctness of the Isolate Reduction rule (for vertices), observe that a graph with isolated vertices has no total dominating set at all. The correctness of the other rules is easily seen. The Isolate Reduction (for edges)

is α-preserving by setting $a = b = 0$ in the definition. In other words, endpoints of isolated edges must belong to any TDS solution. The Leaf Reduction is α-preserving by setting $a = b = 1$ in the definition. This is because leaves do not belong to some optimum TDS solution, except when there is a K_2-component in the graph. □

Hence from now on, no vertex can have two leaf neighbors.

Actually, we could generalize the Leaf Reduction towards the following rule:

Twin Reduction. Recall that vertices u and v are said to be *true twins* if $N[u] = N[v]$ and *false twin* if $N(u) = N(v)$.

- If there are two vertices u and v such that $N[u] = N[v]$, i.e., they form true twins, then delete v.
- If there are two vertices u and v such that $N(u) = N(v)$, i.e., they form false twins, then delete v.

In fact, vertices deleted by the Twin Reduction would go into the harmless set, unless they are isolates. As we are not using this rule in some crucial manner in what follows, we present the following result without proof.

Theorem 13. *The Twin Reduction is α-preserving for any $\alpha \geq 1$.*

We shall reduce the length of pendant chains to at most two, based on the following reduction rules. The first one actually generalizes the Isolate Reduction.

Floating Chain Reduction. Delete all floating chains.

Observation 14. *The Floating Chain Reduction is α-preserving for any $\alpha \geq 1$.*

Proof. G' is obtained from G by deleting a floating chain. For the chain, the numbers $a = b$ can be computed (optimally) in linear time; they correspond to the size of optimum solutions for the floating chain component. □

Long Chain Reduction. Assume that G is a graph that contains a path $x - u - v - w - y$, where u, v, w are three consecutive vertices of degree two, where $|N(y)| \geq 2$. Then, construct the graph G' by

- deleting x, u, v, w and
- connecting y to all vertices in $N(x) \setminus \{u\}$ (without creating double edges).

This corresponds to merging x and y and deleting u, v, w. This Long Chain Reduction resembles the folding rule known for Vertex Cover (in Parameterized Complexity, see [24]).

Theorem 15. *The Long Chain Reduction is α-preserving for any $\alpha \geq 1$.*

PROOF. Let G be the original graph and G' the graph obtained from G by deleting the path u, v, w and merging x and y as described by the rule. We show in the following that $a = 2$ (in part (a)) and $b = 2$ (in part (b)) by considering several cases.

(a) Let C be a maximum harmless set (HS) for G. Let us first briefly discuss what happens if $N(x) = \{u\}$, i.e., if x is a leaf. Then, it is not hard to see that an optimum solution C would contain x and w, but not u and v. Merging x and y and deleting u, v, w is now equivalent to deleting the whole pending chain $x - u - v - w$. As $w \in C$, it does not dominate y, so that $C' = C \setminus \{x, w\}$ is a harmless set for G'.

In the following discussion, we can hence assume that x has at least two neighbors. Hence, $\min\{|N(x)|, |N(y)|\} \geq 2$. We now consider cases whether or not $x \in C$ or $y \in C$.

- Assume that $x \in C$ and $y \in C$. Hence, u, w are not dominated neither by x nor by y. As C is maximum, we can assume $|C \cap \{u, v, w\}| = 1$. Indeed, suppose that it is not the case, that is all of u, v and w are in $V \setminus C$. Since $\min\{|N(x)|, |N(y)|\} \geq 2$, we can define the new solution $C \cup \{w\} \setminus \{v_y\}$ of same size where $v_y \in N(y) \setminus \{w\}$. Hence, $C' = C \setminus \{x, u, v, w\}$ is a HS of G', with $|C'| = |C| - 2$.

- Assume that $x \notin C$ and $y \notin C$. First, let us discuss the possibility that $u \notin C$ and $w \notin C$. As C is maximum, the purpose of this is to dominate (i) v and (ii) x and y. To accomplish (i), either $u \notin C$ or $w \notin C$ would suffice. However, as C is maximum, condition (ii) means that $N(x) \setminus C = \{u\}$ and that $N(y) \setminus C = \{w\}$. By our assumptions, $\min\{|N(x)|, |N(y)|\} \geq 2$. Hence, there is a vertex $z \in N(y)$, $z \neq w$. Now, $\tilde{C} = (C \setminus \{z\}) \cup \{w\}$ is also a maximum HS satisfying $\{v, w\} \subseteq \tilde{C}$. From now on, we assume that $|C \cap \{u, v, w\}| = 2$ and that $|((N(x) \cup N(y)) \setminus (\{u, w\} \cup C)| \geq 1$ (in other words, at least one of x and y has a neighbor in $V \setminus C$ other than u and w, respectively). Hence, $C' = C \setminus \{u, v, w\}$ is a HS of G' with $|C'| = |C| - 2$.

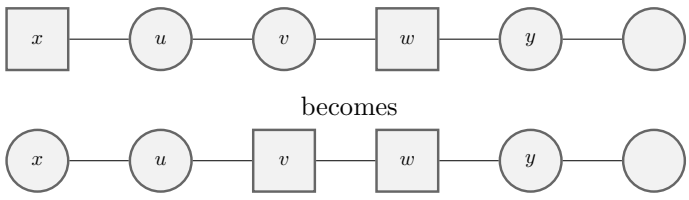

Figure 2: Dealing with $x \in C$, $y \notin C$, with y dominated.

- Assume now that $x \in C$ and $y \notin C$. (As $\min\{|N(x)|, |N(y)|\} \geq 2$, the case that $x \notin C$ and $y \in C$ is symmetric.) We argue in the following that we can obtain another solution no worse than C that matches one of the previously considered cases, by locally exchanging some vertices between

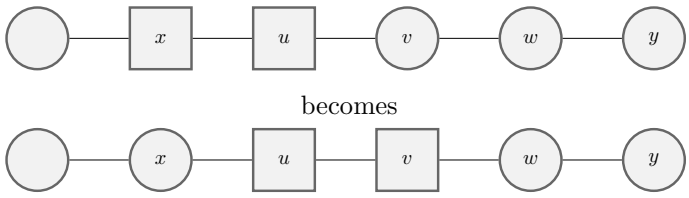

Figure 3: Dealing with $x \in C$, $y \notin C$, with x dominated.

C and its complement. These cases are depicted in Figures 2 and 3. There, squared nodes refer to members of the HS, and round nodes to vertices in the corresponding TDS.

As u is not dominated by x, either (i) $\{u, v\} \subseteq V \setminus C$ or (ii) $\{v, w\} \subseteq V \setminus C$. In case (i), x is dominated by u, but y must (still) be dominated by some vertex from $N(y) \setminus \{w\}$. In case (ii), symmmetrically y is dominated by w, but x must be dominated by some vertex from $N(x) \setminus \{u\}$. In both cases, $\tilde{C} = (C \setminus \{x\}) \cup \{v\}$ is another maximum harmless set of G. This leads us back to the previous item (i.e., $|C'| = |C| - 2$.)

Summarizing, we have shown that from C we can construct a harmless set C' for G' with $|C'| = |C| - 2$. Thus the maximum harmless set C'^* for G' satisfies $|C'^*| \geq |C'| = |C| - 2$. Therefore, the inequality 1 of Definition 1 is satisfied with $a = 2$.

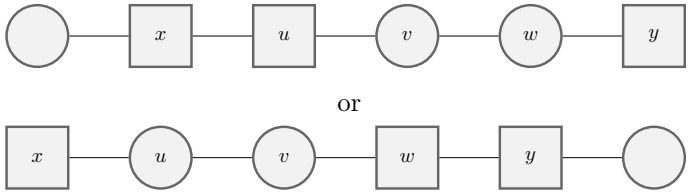

Figure 4: How to regain a solution from C' when $y \in C'$.

(b) Conversely, assume C' is some harmless set for G'. We distinguish two cases:

- Assume that $y \in C'$. Then, y is dominated by some z in its neighborhood (in G'). We consider two cases according to the situation in G. (i) If $z \in N(x)$, then $C = C' \cup \{x, u\}$ is a HS in G. (ii) If $z \in N(y) \setminus N(x)$, then $C = C' \cup \{x, w\}$ is a HS in G. In both cases, $|C| = |C'| + 2$. We refer to Figure 4.

- If $y \notin C'$, then again y is dominated by some z in its neighborhood (in G'). We perform the same case distinction as in the previous case: (i) If $z \in N(x)$, then $C = C' \cup \{u, v\}$ is a HS in G. (ii) If $z \in N(y) \setminus N(x)$, then

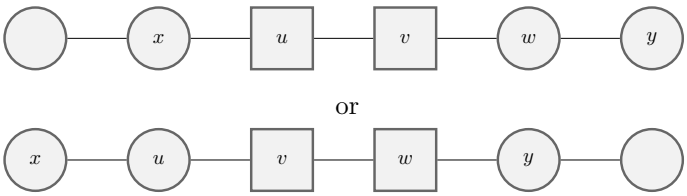

Figure 5: How to regain a solution from C' when $y \notin C'$.

$C = C' \cup \{v, w\}$ is a HS in G. In both cases, $|C| = |C'| + 2$. We refer to Figure 5.

Thus the inequality 2 of Definition 1 is satisfied with $b = 2$. □

Similarly, one sees the correctness of the following rule.

Cycle Chain Reduction. If G is a graph that contains a cycle $x-u-v-w-x$, where u, v, w are three consecutive vertices of degree two, then construct the graph G' by deleting u.

Observation 16. *The cycle chain reduction is α-preserving for any $\alpha \geq 1$.*

Proof. An optimum harmless set for G will put exactly two out of the three vertices u, v, w into the harmless set. W.l.o.g., let these be u and v. Conversely, w and x would go into the total dominating set. Also, in the reduced graph, v will be in the harmless set, while x and w will be in the total dominating set. This shows the claim with constants $a = b = 1$. □

Hence, a graph reduced by the reduction rules mentioned so far may only contain chains of length four, with at most two consecutive vertices of degree two.

Finally, we deal with support vertices with multiple pendant chains. Assuming the Long Chain Reduction has been applied, any pendant chain is of length two or less. Accordingly, a support vertex where two of more pendant chains meet does belong to some optimum solution. The following rule makes this idea more precise.

Pendant Chain Reduction. Assume that $G = (V, E)$ is a graph that contains two pendant chains with common endpoint v of which at least one path is of length two. Then, construct the graph $G' = (V', E')$ by deleting one of the two pendant chains, keeping one which is of length two.

Theorem 17. *The Pendant Chain Reduction is α-preserving for any $\alpha \geq 1$.*

Proof. Let $v - x - y$ and (a) $v - z - t$ (or (b) just $v - t$) be two pendant paths of G. Sketches to these two cases can be found in Figure 6. Then y belongs to some maximum harmless set C of G while x and v belong to $V(G) \setminus C$. Similarly,

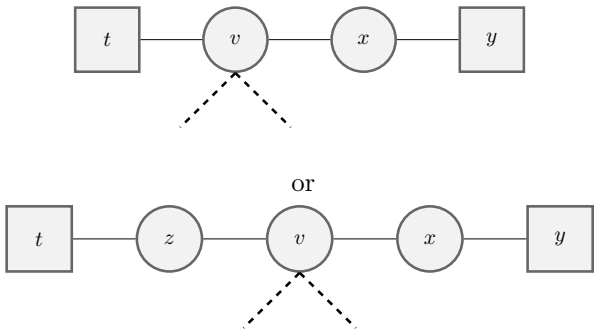

Figure 6: Sketches of the two cases in the Pendant Chain Reduction.

z (if it exists) belongs to $V(G) \setminus C$, while t belongs to C. It follows that the definition of α-preserving reduction can be applied with $a = b = 1$. Notice that, because we keep $v - x - y$, neither v nor x will belong to any harmless set solution for the reduced graph. Hence, adding t to the harmless set solution of the reduced graph is always possible, resulting in a valid harmless set for the original graph. □

We are now in the position to apply Theorem 11.

Observation 18. *Assume the graph $G = (V, E)$ is reduced according to the reduction rules described so far. Then, G satisfies the following properties:*

- *G contains no chain of three vertices of degree two.*
- *By the Leaf Reduction rule, any vertex has at most one leaf neighbor.*

We can summarize our findings as follows:

Theorem 19. Harmless Set *is factor-2 polynomial-time approximable.*

Proof. We will first describe how the approximation algorithm works. Let us assume that $G = (V, E)$ is an instance of Harmless Set that is reduced according to the previously presented α-preserving reduction rules. In particular, G satisfies the properties listed in Observation 18. Let G' be a graph isomorphic to G so that each vertex v of G corresponds to a vertex v' of G', under the assumed isomorphism $f : V(G) \longrightarrow V(G')$. Let L be the set of leaves of G (and correspondingly L' the set of leaves of G'). Let N be the set of leaf neighbors of G and N' be the set of leaf neighbors of G'. By our properties of G and by construction, $|N'| = |N| = |L| = |L'|$. Let $\tilde{G} := G[V(G) \setminus L]$ and $\tilde{G'} := G'[V(G') \setminus L']$. We construct a graph H obtained from the graph union of $\tilde{G}$ and $\tilde{G'}$ simply by adding edges between each pair of corresponding leaf neighbors $v \in N$ of G and $v' = f(v) \in N'$.

Due to the application of the Pendant Chain Reduction rule to G (and G'), the addition of edges between corresponding leaf neighbors in G and G' does not introduce induced cycles with more than two consecutive degree-two vertices.

Observe that the resulting graph H might contain chains with more than two vertices of degree two. This happens if G contains a pendant chain with two vertices of degree two. Let $z-y-x-v$ be such a path, with z being a leaf and v having degree at least three. Connecting the two leaf neighbors y and y' yields the chain $v-x-y-y'-x'-v'$ in H that contains four consecutive vertices of degree two. In actual fact, this is the only way how H could contain such long chains. To the resulting graph H, apply the Long Chain Reduction rule as long as possible. Notice that an application of this rule does never decrease degrees, adds two vertices to the solution and removes four vertices of the graph. In our little example, $v-x-y-y'-x'-v'$ would be converted into $v-v'$.

This results in a graph H' of order $n_{H'}$ with minimum degree at least two containing no chain of three vertices of degree two. Hence, we can apply the polynomial-time algorithm stated in Theorem 10 that returns a TDS $D_{H'}$ for H' with $2|D_{H'}| \leq n_{H'}$. Undoing a certain number of Long Chain Reductions, say, c, that we applied, we obtain a TDS D_H for H with $2|D_H| = 2(|D_{H'}| + 2c) \leq n_{H'} + 4c = n_H$.

Now, we are going to separate H again into G and G'. Let $D := D_H \cup N \cup N'$. Notice that it is possible that some vertices of $N \cup N'$ already belong to D_H.

We will describe how to add further vertices to D in what follows in order to turn D into a total dominating set $D_{G \cup G'}$ for the graph union $G \cup G'$.

Namely, the following needs to be done for each pair of vertices x, x' where $x \in N$ and $x' \in N'$. It should become clear by the analysis that for each such pair, at most two vertices are added to D_H to produce the final $D_{G \cup G'}$, so that we can conclude that $|D_{G \cup G'} \setminus D_H| \leq 2|L| = 2|N|$. This is formally seen, as we define a function $f : N \cup N' \to V \cup V'$ such that $D_{G \cup G'}$ is a total dominating set of $G \cup G'$ that can be described as

$$D_{G \cup G'} = D_H \cup f(N \cup N').$$

Notice that the vertices in $L \cup L'$ are the only ones that are potentially not yet dominated in $G \cup G'$; this is why we can restrict our attention to these vertices. So, let $x \in N$ in the following.

- If $x \in D_H$ and $x' \in D_H$, then choose some $y \in N_G(x)$ and put both y and y' into D. Define $f(x) = y$ and $f(x') = y'$. Hence, $\{x, x', y, y'\} \setminus D_H \subseteq \{f(x), f(x')\}$.
- If $x \in D_H$ but $x' \notin D_H$, then x is already dominated by some y in H that belongs to G. Only x' needs to be fixed; for simplicity, add y' to D. Define $f(x) = x'$ and $f(x') = y'$. Here, $\{x, x', y, y'\} \setminus D_H \subseteq \{f(x), f(x')\}$.
- If neither x nor x' belong to D_H, then x is dominated (in H) by some y that belongs to G and x' is dominated by some z' from G'. Define $f(x) = x$ and $f(x') = x'$. Now, $\{x, x', y, y', z, z'\} \setminus D_H \subseteq \{f(x), f(x')\}$.

Now, we want to apply Theorem 11 to the graphs $G \cup G'$ and H, and to the TDS solutions $D_{G\cup G'}$ and D_H that we obtained for these graphs so far. We have to check if the conditions of that theorem are met. Observe that H is obtained from $G \cup G'$ by deleting $d = 2|L|$ (leaf) vertices and adding some edges, although this is not the original description of the construction of H (from G). Notice that $d = 2|L| \leq \gamma_t(G \cup G') = 2\gamma_t(G)$. Also, for $a = |D_{G\cup G'}| - |D_H|$, we have shown that $a = |D_H \cup f(N \cup N')| - |D_H| = |(D_H \cup f(N \cup N')) \setminus D_H| \leq |f(N \cup N')| \leq 2|N| = d$.

By our reduction rules and by our construction, the graph H' obtained from H satisfies the Lam-Wei property. Hence, we can apply Theorem 10 and obtain (in polynomial time) a total dominating set $D_{H'}$ for H' and, as explained above, from $D_{H'}$ we can construct (in polynomial time) a TDS D_H for H that satisfies $2|D_H| \leq n_H$. Theorem 11 allows us hence to conclude that $V(G \cup G') \setminus D_{G\cup G'}$ is a 2-approximate harmless set solution for $G \cup G'$ that can be computed in polynomial time.

By symmetry, we can assume that $|D_{G\cup G'} \cap V(G)| \leq |D_{G\cup G'} \cap V(G')|$. Hence, $D_G := D_{G\cup G'} \cap V(G)$ is a TDS for G, such that $V(G) \setminus D_G$ is a 2-approximate harmless set solution for G. Finally, notice that the graph G we are talking about in the last paragraphs is already reduced with respect to the reduction rules that we presented. However, as all these reduction rules are α-preserving for any $\alpha \geq 1$, we can also obtain a 2-approximate harmless set solution for any graph $\hat{G}$. Namely, if the graph G was obtained from $\hat{G}$ by exhaustively applying our reduction rules, we also know how to construct a 2-approximate harmless set solution $HS_{\hat{G}}$ for $\hat{G}$ from $V(G) \setminus D_G$ in polynomial time. □

4. The differential of a graph

Let us start with an alternative presentation of this notion. Let $G = (V, E)$ be a graph. For $X \subseteq V$, let

$$\partial(X) := \left| \left(\bigcup_{x \in X} N(x) \right) \setminus X \right| - |X|.$$

$\partial(X)$ is called the *differential* of the set X, and our aim is to find a vertex set that maximizes this quantity. This maximum quantity is known as the differential of G, written $\partial(G)$. We introduced above the *Roman domination number* $\gamma_R(G)$ as the smallest value of any Roman domination function on V. As was shown in [11], the following Gallai-type equality holds: $\gamma_R(G) + \partial(G) = |V|$ for any graph $G = (V, E)$. This is the same type of relationship as we saw for domination (and nonblocker) and for total domination (and harmless set).

The following combinatorial results are known:

Theorem 20. *[9, 11, 19] Let G be a connected graph of order n.*

- *If $n \geq 3$, then $\partial(G) \geq n/5$.*

- *If G has minimum degree at least two, then $\partial(G) \geq \frac{3n}{11}$, apart from five exceptional graphs, none of them having more than seven vertices.*

It is not hard to turn the first combinatorial result into a kernelization result, yielding a kernel bound of $5k$, where k is the natural parameterization of the DIFFERENTIAL. Along the lines of [7], we can obtain a factor-5 approximation by first computing a spanning tree $T = (V, E_T)$ for G and then computing an optimum differential set D_T in T by dynamic programming, and then observing that D_T is a factor-5 approximation for G. In [8], this result was improved to a kernel whose order is bounded by $4k$. Along those lines, we can also get a factor-4 approximation. However, the second item of Theorem 20 suggests a possible improvement to a factor of $\frac{11}{3}$ if we employ our framework. This is what we are going to endeavor in this section.

1. **Leaf Reduction.** If there are two leaves adjacent to the same vertex, then connect these leaves by an edge.
2. **Hair Reduction.** If there are two hairs connected to the same vertex, then remove the two hair leaves.
3. **Hair-Leaf Reduction.** If there is a leaf and a hair connected to the same vertex, then remove the hair leaf.
4. **Long Hair Reduction.** If there is a long hair $u - v - w - \cdots$, then remove u, v, w.
5. **Neighbor Hair Reduction.** If there is a hair $u - v - \cdots$ connected to a vertex w and another hair $u' - v' - \cdots$ connected to a neighbor w' of w, then remove the edge ww'.

Table 1: Reductions for DIFFERENTIAL.

First, we show that the reduction rules presented in [8] as kernelization rules (and exhibited in adapted form in Table 1) can be also interpreted as α-preserving rules. We use some non-standard terminology for stating the rules. A *hair* is a sequence of two vertices uv, where u is a leaf and v has degree two. Then, u is also called a *hair leaf*. We use the following simple notation for a hair uv for reasons of clarity: $u-v-\cdots$. Notice that a hair corresponds to a pendant chain of length two. Accordingly, a pendant chain of length three corresponds to a *long hair*, referring more precisely to three vertices $u-v-w-\cdots$, where v and w are of degree two and u is a leaf. The first three reduction rules from Table 1 are illustrated in Figures 7 and 8. There, round nodes indicate vertices that belong (without loss of generality) to a set X delivering the biggest differential. Likewise, they would receive a value of two by an optimum Roman domination function. Squared nodes denote vertices that are not in X.

In the reasoning given for the rules in [8], only for the Long Hair Reduction, the natural parameter k upperbounding the differential of the graph changes (decreases by one). The argument shows (for the other cases) that even if a set of vertices is produced for the reduced graph that is not a valid solution for the original graph, still another solution can be constructed that is not

Figure 7: Sketch of the Leaf Reduction.

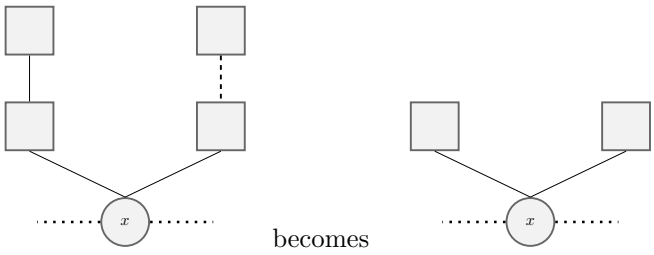

Figure 8: Sketch of the Hair and the Hair-Leaf Reductions.

worse (smaller) than the one that was obtained, so that approximation factors are clearly preserved. Hence, all rules but the Long Hair Reduction are α-preserving as testified by the numbers $a = b = 0$ in the definition. The Long Hair Reduction can be seen to be α-preserving when setting $a = b = 1$. Hence, we can summarize:

Observation 21. *The five reduction rules listed in Table 1 are α-preserving for any $\alpha \geq 1$.*

Lemma 22. *[8] Let G be a graph where none of the previous five reductions applies. Then, G has the following properties:*

(1) *To each vertex, at most one leaf or one hair is attached, but not both together.*
(2) *If we remove all leaves and all hairs from G, then the remaining graph, henceforth called* nucleus*, has minimum degree of at least two.*
(3) *If a hair is attached to a vertex u in the nucleus, then no hair is attached to any neighbor of u within the nucleus.*

Notice that the properties listed in Lemma 22 ensure that when obtaining the nucleus H from the reduced graph G by deleting d vertices, $d \leq \gamma_R(G)$ is verified. In order to verify that a sufficiently big solution for the nucleus can be found in polynomial time, observe the proof strategy of [9]: there, the differential of a graph is modeled by so-called big star packings. It is possible to start with a greedily obtained big star packing and then further modify the solution, using the local (and hence easy-to-check) criteria exhibited in various lemmas of that paper, up to the point when no further improvements are possible. The

big star packing obtained in this way corresponds to a differential set D with $\partial(D) \geq \frac{3}{11}n$, where n is the order of the graph.

As the proof in [9] uses extremal combinatorial arguments, it is (at least at first glance) non-constructive. Let us give some more details of the algorithm that is hidden within these combinatorial arguments in the following.

The greedy selection of a big star packing. We will (from now on) work on a mixed graph (i.e., a graph that has both directed and undirected edges; directed edges are also called arcs) such that each vertex has at most one outgoing arc, and no vertex with an incoming arc has an outgoing arc. We will call a vertex incident to some directed arc *marked*. As we start with an undirected graph $G = (V, E)$, at the beginning all vertices are unmarked. We proceed as follows:

As long as possible:

- Pick some unmarked vertex x with at least two unmarked neighbors.
- Direct all edges connecting x to any unmarked neighbor towards x.

Now, consider the set D of vertices to which some arcs point to, and let $B(D)$ denote the remaining marked vertices. Clearly, $\left(\bigcup_{x\in D} N(x)\right) \setminus D = B(D)$. Hence, $\partial(D) = |B(D)| - |D|$. Moreover, due to the directions of the edges, we can view each $x \in D$ as the center of a star to which at least two arcs (rays) are pointing. So, we have defined a collection $\mathcal{S}(D)$ of stars that can be viewed as a star packing. As each star has at least two rays, we call them *big stars*. Due to our greedy approach, we hence arrive at $\mathcal{S}(D)$ as being a maximal big star packing. Moreover, $|B(D)|$ is also the number of directed edges (or rays) in total. Let $C(D) := V \setminus (B(D) \cup D)$.

By definition of the partition $(D, B(D), C(D))$ of V we find:

Observation 23. *No edge connects vertices from D with vertices from $C(D)$.*

As we obtain a maximal big star packing, we conclude:

Observation 24. *The induced graph $G[C(D)]$ is undirected and decomposes into K_1- and K_2-components.*

First local improvement. We are now going to improve the solution found so far.

As long as possible:

- Pick some vertex x from $B(D)$ that has two or more neighbors in $C(D)$.
- Let $y \in D$ be such that the edge xy is directed towards y.
- Replace the arc from x to y by an undirected edge again.
- If there is now (only) one arc zy directed to y, remove y from D and render zy an undirected edge again. (This will increase the number of unmarked vertices.)

- Direct all edges that connect x to some unmarked vertex towards x and put x into D.

If no further improvements are possible, one might want to ensure that the (new) big star packing $\mathcal{S}(D)$ is still maximal. If not, obvious further improvements are possible. However, after a finite number of steps, this will end.

The new set D (and the related star packing $\mathcal{S}(D)$ that can be read off from the directed edges) satisfies Observation 24 and the following one.

Observation 25. *Every $x \in B(D)$ has at most one neighbor in $C(D)$.*

Second local improvement. By a procedure similar to the previous case, we can create new stars if some $x \in B(D)$ is part of a star with at least three rays and neighbor of some K_2-component in $G[C(D)]$. Leaving out details in this case, we can observe for the (new) differential set D:

Observation 26. *If $x \in B(D)$ is neighbor of some K_2-component in $C(D)$, then it belongs to some star with at most two rays.*

Some simple computations (as undertaken in [9], following Lemma 3.9) show that the set D satisfies our desired bound, i.e., $|D| \geq \frac{3}{11}|V|$, if the packing $\mathcal{S}(D)$ only contains stars with at least three rays. It could well be that the (valid) differential set D satisfies the bound and we can stop here.

Further local improvements on smaller stars $K_{1,2}$. If the packing $\mathcal{S}(D)$ contains smaller stars with two rays, then we have to make further local improvements on these smaller stars, considering them in groups. The details can be found in Lemmas 3.10 through 3.17 in [9], but this should make clear that finally we can obtain a sufficiently big differential in polynomial time.

Having obtained such differential set for the nucleus of a graph, this solution can be easily lifted to a solution of the reduced graph; Theorem 27 (below) allows us to conclude with Theorem 28.

We are going to use the idea of computing a sufficiently big solution for the nucleus, based on the following variant of Theorem 11. In order to state this variant similar to the previous ones, we use two disjoint sets of vertices with subscripts one and two to (implicitly) define a Roman domination function that maps the vertices in the set with subscript i onto the number i. To further strengthen the analogy, we call the union of these two sets a *Roman dominating set*, or Roman DS for short.

Theorem 27. *Let G be a graph of order n_G and let H be a graph of order n_H obtained from G by deleting d vertices. Let $D_G = D_{G,1} \cup D_{G,2}$ and $D_H = D_{H,1} \cup D_{H,2}$ be Roman DS solutions of G and H, respectively, such that $D_{H,2} = D_{G,2}$ and $a = |D_{G,1}| - |D_{H,1}| \leq d$. If $|D_{H,1}| + 2|D_{H,2}| \leq c \cdot n_H$ for some $c < 1$ and if $d \leq \gamma_R(G)$, then $\partial(V(G) \setminus D_G) = n_G - 2|D_{G,2}| - |D_{G,1}|$ is within a factor of $(1-c)^{-1}$ from optimum.*

PROOF. As $n_H = n_G - d$, $|D_{G,1}| + 2|D_{G,2}| = |D_{H,1}| + a + 2|D_{H,2}| \leq c(n_G - d) + a = cn_G + (a - cd) \leq cn_G + d - cd = cn_G + (1-c)d \leq cn_G + (1-c)\gamma_R(G)$. Hence, $n_G - 2|D_{G,2}| - |D_{G,1}| \geq n_G - cn_G - (1-c)\gamma_t(G) = (1-c)(n_G - \gamma_R(G)) = (1-c)\partial(G)$. This immediately yields an approximation factor of $(1-c)^{-1}$. □

We described that we can turn the (non-constructive) combinatorial reasoning of [9] into a polynomial-time algorithm. Moreover, we can lift a Roman domination function $f_H : V(H) \to \{0, 1, 2\}$ on the nucleus H to a Roman domination function $f_G : V(G) \to \{0, 1, 2\}$ on the (reduced) graph G by setting $f_G(x) = f_H(x)$ for $x \in V(H)$ and $f_G(x) = 1$ for $x \in V(G) \setminus V(H)$. Also, if G satisfies the properties listed in Lemma 22, the number $d = |V(G) \setminus V(H)|$ of deleted vertices is upperbounded by $\gamma_R(G)$, as it is best to assign 2 to the leaf neighbor. Furthermore, accounting for each leaf neighbor, at most two vertices are deleted, namely when a hair is attached to some vertex. This allows us to conclude within our framework:

Theorem 28. DIFFERENTIAL *is factor-*$\frac{11}{3}$ *polynomial-time approximable.*

Notice that better combinatorial bounds for Roman Domination (and hence for Differential) have been obtained for 2-connected graphs and for graphs of minimum degree at least three by Liu and Chang [36, 35]. It would be interesting to see if the corresponding proofs could be turned into better kernelization or approximation algorithms for the computational problems.

5. Multiple Nonblocker sets

We are first going to explain why neither some nice approximation algorithm nor some FPT algorithm (with the standard parameterization) yields useful results. We shall assume $k > 1$ in this section.

Theorem 29. k-DOMINATING SET, $k > 1$, *is not approximable within a factor better than* $c \log |V|$ *for some* $c > 0$ *unless* $P = NP$ *[42]. Moreover, the (standard) parameterized version is W[2]-hard.*

PROOF. We show that DOMINATING SET is reducible to k-DOMINATING SET. Given an instance G of DOMINATING SET, we introduce (in total) k copies of each vertex, say, $v[1], \ldots, v[k]$ of vertex v, and introduce a complete bipartite graph $K_{k,k}$ in $\{u[1], \ldots, u[k]\} \cup \{v[1], \ldots, v[k]\}$ whenever there is an edge uv in G. Then, the new graph has a k-dominating set of size kt if and only if the original graph G has a dominating set of size t. □

We consider now a combinatorial upper bound on the size of some feasible solution of the minimization problem.

Theorem 30 ([22]). *Let G be a graph of order n_G and a minimum degree at least k. Then $\gamma_k(G) \leq \frac{k}{k+1} n_G$.*

The known non-constructive proof can be turned into a polynomial-time algorithm obtaining the following result.

Theorem 31. *For a given graph G of order n_G and minimum degree at least k, one can compute a k-dominating set D with $|D| \leq \frac{k}{k+1} n_G$ in polynomial time.*

PROOF. First, we greedily remove edges between vertices of degree greater than k obtaining a graph G' of minimum degree (exactly) k. Let $S = \{v \in V : |N(v)| > k\}$. By construction, S is an independent set in G'. We build a maximal independent set T that contains S. Then $V \setminus T$ is a k-dominating set.

If $|V \setminus T| \leq kn_G/(k+1)$, then $D := V \setminus T$ is also a k-dominating set in the supergraph G of G'. Otherwise, while $|V \setminus T| > kn_G/(k+1)$, construct a maximal independent set T' of $G[V \setminus T]$ and set $T = T'$. We show in the following that the algorithm terminates.

Let $r = |T|$. When $|V \setminus T| > kn_G/(k+1)$, we get $n_G = r + |V \setminus T| > r + kn_G/(k+1) = r + k(r + |V \setminus T|)/(k+1)$, thus $|V \setminus T| > kr$. Since T is a maximal independent set (and hence a dominating set) and every element of $V \setminus T$ is of degree k in G', every vertex of $V \setminus T$ has degree at most $k-1$ in $G'[V \setminus T]$. It follows that any maximal independent set of $G'[V \setminus T]$ contains at least $r+1$ vertices (otherwise, $|V \setminus T| \leq r + r(k-1) = rk$). Compute any maximal independent set T' of $G'[V \setminus T]$. Now $|T'| > r = |T|$ and $V \setminus T'$ is a k-dominating set that is smaller than $V \setminus T$; namely, because the minimum degree is at least k, all elements of any independent set are k-dominated by its complement. □

Our approximation algorithm is based on obtaining a k-dominating set in a graph H obtained from the input G after a number of modifications (mainly vertex deletions and insertions).

Theorem 32. *Let G be a graph of order n_G and let H be a graph of order n_H obtained from G by deleting d vertices and adding $2k$ new vertices, with $d > k$. Let D_G and D_H be k-dominating set solutions of G and H such that $a = |D_G| - |D_H| = d - k$. If $|D_H| \leq c \cdot n_H$ for some $c < 1$ and $d \leq \gamma_k(G)$, then $V(G) \setminus D_G$ is a k-nonblocker of G whose size $n_G - |D_G|$ is within a factor of $(1-c)^{-1}$ from optimum (modulo an additive constant less than k).*

PROOF. As $n_H = n_G - d + 2k$, $|D_G| = |D_H| + a \leq c(n_G - d + 2k) + d - k = cn_G + (1-c)d + 2ck - k \leq cn_G + (1-c)\gamma_k(G) + k(2c-1)$. Hence, $n_G - |D_G| \geq n_G - cn_G - (1-c)\gamma_k(G) - k(2c-1) = (1-c)(n_G - \gamma_k(G)) - k(2c-1)$. This immediately yields an approximation factor of $(1-c)^{-1}$ (modulo the additive constant $k(2c-1) < k$). □

In the rest of this section, we present reduction rules that produce a graph G with minimum degree at least k. Our reduction rules mainly deal with vertices of degree $k-1$ or less. Each such vertex must be in any k-dominating set. We shall refer to such vertices by *low-degree* vertices in the sequel.

Low-Degree Vertex Deletion Reduction. If a low-degree vertex v has only low-degree neighbors, then delete v. If there is a vertex u with at least $k+1$ low-degree neighbors, then delete the edge between u and one low-degree neighbor of u.

Observation 33. *The Low-Degree Vertex Deletion Reduction is α-preserving for any $\alpha \geq 1$.*

PROOF. The soundness of Low-Degree Vertex Deletion is rather straightforward. A low-degree vertex that is not a neighbor of a high-degree vertex can be placed (safely) in any k-dominating set. If the number of low-degree neighbors of a vertex u is $t > k$, then u is dominated by any k of these neighbors. Therefore we can safely delete $t - k$ edges connecting u to low-degree neighbors. We keep k neighbors to make sure that u remains dominated in any subsequent solution. This reduction is α-preserving with constants $a = b = 0$. □

Low-Degree Merging Reduction. Let $G = (V, E)$ be an instance of k-NONBLOCKER that has been subject to the Low-Degree Vertex Deletion Reduction rule. Then we add a complete bipartite graph $K_{k,k}$ with new vertices $u_1, \ldots, u_k$, $v_1, \ldots, v_k$. For every vertex $v \in V$ of degree at least k, having q low-degree neighbors $w_1, \ldots, w_q$, with $q \leq k$, merge w_i with v_i for every $i = 1, \ldots, q$.

Observation 34. *The Low-Degree Merging Reduction rule is weakly α-preserving for any $\alpha \geq 1$.*

PROOF. For the soundness of this rule, observe that, as low-degree vertices will end up in the dominating set, the only purpose of these could be to help dominate other vertices. As every vertex has no more than k low-degree vertices, we can mimick their effect by the $K_{k,k}$ gadget that we insert instead.

We are going to verify the definition of α-preserving reductions; to this end, we show that $a = b = -k$ works out in our case. In fact, we must refer to the variant sketched in Remark 1 to understand this concept.

Let G be the original graph and G' the graph obtained from G after applying the reduction rule.

(a) Let C be a maximum k-nonblocker for G. Observe that C does not contain any low-degree vertices since they are part of any k-dominating set. Consider $C' = C \cup \{u_1, \ldots, u_k\}$. Then C' is a maximum k-nonblocker set for G' of size $|C'| = |C| + k$.

(b) Consider now the converse. Let C' be some k-nonblocker set for G'. We can suppose that C' contains $u_1, \ldots, u_k$, otherwise we remove $v_1, \ldots, v_k$ and add $u_1, \ldots, u_k$. Thus $C = C' \setminus \{u_1, \ldots, u_k\}$ is a k-nonblocker set for G of size $|C| = |C'| - k$. □

The reductions above take polynomial time, so that Theorem 32 allows us to conclude:

Theorem 35. *k-NONBLOCKER is factor-$(k+1)$ polynomial-time approximable (modulo an additive constant less than k).*

Combinations with the previous section as indicated in the definitions of [30] should be possible. We leave this for future research, similar to variants like Liar's Domination; see [12] and the literature quoted therein.

6. Conclusions

We presented a framework for obtaining approximation algorithms for maximization problems, inspired by similar reasonings for obtaining kernelization results. We see five major directions from this approach:

- Paraphrasing [26], we might say that not only FPT, but also polynomial-time maximization is *P-time extremal structure*. This should inspire mathematicians working in graph theory (and other areas of combinatorics) to work out useful combinatorial bounds on different graph parameters. We started on domination-type parameters, and this might be a first venue of continuation, for example, along the lines sketched in [14, 15, 33].

- Conversely, approximation algorithms that stay within the combinatorial grounds of their problem tend to reveal (combinatorial) insights into the problem that might get lost when moving, for instance, into the area of Mathematical Programming.

- The notion of α-preserving reduction is similar to the local ratio techniques [5] that allowed to re-interpret many (e.g., primal-dual) approximation algorithms (for minimization problems) in a purely combinatorial fashion; see [6]. We see some hope for similar developments using α-preserving reduction for maximization problems.

- The fact that α-preserving reductions are inspired by FPT techniques should allow to adapt these notions for obtaining new and faster parameterized approximation algorithms.

- Reductions are often close to practical heuristics and hence allow for fast implementations.

Acknowledgements. We are grateful for having had the chance of discussing this paper at the Bertinoro Workshop on Parameterized Approximation in May 2014. This work was supported by the bilateral research cooperation CEDRE between France and Lebanon (grant number 30885TM). Further support of this research by visiting professorships granted to the first and the last of the authors by the University of Paris-Dauphine is gratefully acknowledged, which helped us meet in Paris for preparing this manuscript together. A major part of this work was done when the third author was affiliated with the *Institut für Optimierung und Operations Research, Universität Ulm*, Germany. An extended abstract of this paper has been presented at ISAAC 2014, see [1].

Note added in proof. Recently, we became aware of a new framework for approximative kernelizations contained in a manuscript entitled *Lossy-Kernelization*, written by Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. The notions developed there are partly similar to the ones that we present in this paper. However, the focus of that manuscript is on developing a notion of kernelization (and hence of reduction rules) that is tailored towards parameterized approximation, while we are deliberately focussing on making use of reduction rules for obtaining (classical) polynomial-time approximations.

References

[1] F. N. Abu-Khzam, C. Bazgan, M. Chopin, and H. Fernau. Approximation algorithms inspired by kernelization methods. In H.-K. Ahn and C.-S. Shin, editors, *Algorithms and Computation — 25th International Symposium, ISAAC 2014*, volume 8889 of *LNCS*, pages 479–490. Springer, 2014.

[2] S. Athanassopoulos, I. Caragiannis, C. Kaklamanis, and M. Kyropoulou. An improved approximation bound for spanning star forest and color saving. In R. Královic and D. Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS*, volume 5734 of *LNCS*, pages 90–101. Springer, 2009.

[3] G. Ausiello, C. Bazgan, M. Demange, and V. Th. Paschos. Completeness in differential approximation classes. *International Journal of Foundations of Computer Science*, 16(6):1267–1295, 2005.

[4] G. Ausiello, P. Creczenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation; Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[5] R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz. Local ratio: a unified framework for approximation algorithms. *ACM Surveys*, 36(4):422–463, 2004.

[6] R. Bar-Yehuda and D. Rawitz. On the equivalence between the primal-dual schema and the local-ratio technique. *SIAM Journal of Discrete Mathematics*, 19:762–797, 2005.

[7] C. Bazgan and M. Chopin. The robust set problem: Parameterized complexity and approximation. In B. Rovan, V. Sassone, and P. Widmayer, editors, *Mathematical Foundations of Computer Science 2012 – 37th International Symposium, MFCS*, volume 7464 of *LNCS*, pages 136–147. Springer, 2012.

[8] S. Bermudo and H. Fernau. Combinatorics for smaller kernels: The differential of a graph. *To appear in Theoretical Computer Science*.

[9] S. Bermudo and H. Fernau. Lower bounds on the differential of a graph. *Discrete Mathematics*, 312:3236–3250, 2012.

[10] S. Bermudo and H. Fernau. Computing the differential of a graph: hardness, approximability and exact algorithms. *Discrete Applied Mathematics*, 165:69–82, 2014.

[11] S. Bermudo, H. Fernau, and J. M. Sigarreta. The differential and the Roman domination number of a graph. *Applicable Analysis and Discrete Mathematics*, 8:155–171, 2014.

[12] A. Bishnu, A. Ghosh, and S. Paul. Parameterized complexity of k-tuple and liar's domination. *CoRR*, abs/1309.5461, 2013.

[13] M. Blank. An estimate of the external stability number of a graph without suspended vertices (in Russian). *Prikl. Math. i Programmirovanie Vyp.*, 10:3–11, 1973.

[14] M. Borowiecki and D. Michalak. Generalized independence and domination in graphs. *Discrete Mathematics*, 191:51–56, 1998.

[15] A. Bouchou, M. Blidia, and M. Chellali. Relations between the Roman k-domination and Roman domination numbers in graphs. *Discrete Mathematics, Algorithms and Applications*, 6(3), 2014.

[16] L. Brankovic and H. Fernau. Parameterized approximation algorithms for Hitting Set. In R. Solis-Oba and G. Persiano, editors, *Approximation and Online Algorithms — 9th International Workshop, WAOA 2011*, volume 7164 of *LNCS*, pages 63–76. Springer, 2012.

[17] L. Brankovic and H. Fernau. A novel parameterised approximation algorithm for minimum vertex cover. *Theoretical Computer Science*, 511:85–108, 2013.

[18] Y. Caro and Y. Roditty. A note on the k-domination number of a graph. *International Journal of Mathematics and Mathematical Sciences*, 13(1):205–206, 1990.

[19] E. W. Chambers, B. Kinnersley, N. Prince, and D. B. West. Extremal problems for Roman domination. *SIAM Journal of Discrete Mathematics*, 23:1575–1586, 2009.

[20] N. Chen, R. Engelberg, C. T. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved approximation algorithms for the spanning star forest problem. *Algorithmica*, 65(3):498–516, 2013.

[21] M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206:1264–1275, 2008.

[22] E. J. Cockayne, B. Gamble, and B. Shepherd. An upper bound for the k-domination number of a graph. *Journal of Graph Theory*, 9:533–534, 1985.

[23] F. Dehne, M. Fellows, H. Fernau, E. Prieto, and F. Rosamond. Nonblocker: parameterized algorithmics for minimum dominating set. In J. Štuller, J. Wiedermann, G. Tel, J. Pokorný, and M. Bielikova, editors, *Software Seminar SOFSEM*, volume 3831 of *LNCS*, pages 237–245. Springer, 2006.

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

[25] R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 49–99. 1999.

[26] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-time extremal structure I. In H. Broersma, M. Johnson, and S. Szeider, editors, *Algorithms and Complexity in Durham ACiD 2005*, volume 4 of *Texts in Algorithmics*, pages 1–41. King's College Publications, 2005.

[27] H. Fernau. Roman Domination: a parameterized perspective. *International Journal of Computer Mathematics*, 85:25–38, 2008.

[28] J. F. Fink and M. S. Jacobson. n-domination in graphs. In *Graph Theory and Its Applications to Algorithms and Computer Science*, pages 283–300. Wiley, 1985.

[29] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 470–479. IEEE Computer Society, 2012.

[30] A. Hansberg and L. Volkmann. Upper bounds on the k-domination number and the k-Roman domination number. *Discrete Applied Mathematics*, 157:1634–1639, 2009.

[31] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*, volume 208 of *Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker, 1998.

[32] M. Hennings and A. Yeo. *Total Domination in Graphs*. Springer, 2013.

[33] K. Kämmerling and L. Volkmann. Roman k-domination in graphs. *Journal of the Korean Mathematical Society*, 46:1309–1318, 2009.

[34] P. C. B. Lam and B. Wei. On the total domination number of graphs. *Utilitas Mathematica*, 72:223–240, 2007.

[35] C.-H. Liu and G. J. Chang. Roman domination on 2-connected graphs. *SIAM Journal of Discrete Mathematics*, 26(1):193–205, 2012.

[36] C.-H. Liu and G. J. Chang. Upper bounds on Roman domination numbers of graphs. *Discrete Mathematics*, 312(7):1386–1391, 2012.

[37] J. L. Mashburn, T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and P. J. Slater. Differentials in graphs. *Utilitas Mathematica*, 69:43–54, 2006.

[38] B. McCuaig and B. Shepherd. Domination in graphs of minimum degree two. *Journal of Graph Theory*, 13:749–762, 1989.

[39] N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. LP can be a cure for parameterized problems. In C. Dürr and T. Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 14 of *LIPIcs*, pages 338–349. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[40] C. T. Nguyen, J. Shen, M. Hou, Li Sheng, W. Miller, and L. Zhang. Approximating the spanning star forest problem and its application to genomic sequence alignment. *SIAM Journal on Computing*, 38(3):946–962, 2008.

[41] E. Prieto. *Systematic Kernelization in FPT Algorithm Design*. PhD thesis, The University of Newcastle, Australia, 2005.

[42] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484. ACM Press, 1997.

[43] B. Reed. Paths, stars, and the number three. *Combinatorics, Probability and Computing*, 5:277–295, 1996.

[44] P. J. Slater. Enclaveless sets and MK-systems. *Journal of Research of the National Bureau of Standards*, 82(3):197–202, 1977.