Linear-Time Computation of a Linear Problem Kernel for Dominating Set on Planar Graphs

René van Bevern[1,⋆], Sepp Hartung[1], Frank Kammer[2], Rolf Niedermeier[1], and Mathias Weller[1,⋆⋆]

[1] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
{rene.vanbevern, sepp.hartung, rolf.niedermeier, mathias.weller}@tu-berlin.de
[2] Institut für Informatik, Universität Augsburg, Augsburg, Germany
kammer@informatik.uni-augsburg.de

Abstract. We present a linear-time kernelization algorithm that transforms a given planar graph G with domination number $\gamma(G)$ into a planar graph G' of size $\mathrm{O}(\gamma(G))$ with $\gamma(G) = \gamma(G')$. In addition, a minimum dominating set for G can be inferred from a minimum dominating set for G'. In terms of parameterized algorithmics, this implies a linear-size problem kernel for the NP-hard Dominating Set problem on planar graphs, where the kernelization takes linear time. This improves on previous kernelization algorithms that provide linear-size kernels in cubic time.

1 Introduction

This work lies in the intersection of two active lines of research:

1. NP-hard problems on planar graphs and the exploitation of their structural properties to obtain better algorithms (approximation or fixed-parameter) [4, 5, 14], and
2. polynomial-time data reduction and problem kernelization [6, 14], an important subfield of parameterized complexity analysis.

Indeed, planar graph problems played an important role in the development of several lines of research in parameterized complexity analysis. More specifically, the topic of subexponential time fixed-parameter algorithms was first studied for the Dominating Set problem on planar graphs [1, 10]. The linear-size problem kernel for Dominating Set on planar graphs [2] may be considered as a nucleus for the recent, rapid growth of results on problem kernels [6, 14], and planar graph problems

⋆ Supported by the DFG, project AREG, NI 369/9.
⋆⋆ Supported by the DFG, project DARE, NI 369/11.

Proceedings of the 6th International Symposium on Parameterized and Exact Computation (IPEC'11), Saarbrücken, Germany, September 2011, volume 7112 in Lecture Notes in Computer Science, pp. 194–206.

led to the first tractability results for the local search paradigm in the context of parameterized complexity [11]. In our work, again Dominating Set serves as a starting example, now for studying the issue of *linear-time* kernelizability as a natural goal within polynomial-time data reduction.

In a nutshell, a kernelization algorithm transforms in polynomial time an instance of a (typically NP-hard) problem into an equivalent instance whose size is bounded from above by a function of a problem-specific parameter. Nowadays, it has become a standard challenge to minimize the size of problem kernels [6, 14]. As to Dominating Set on planar graphs (given an undirected planar graph G and a positive integer k, select at most k vertices such that each unselected vertex in G has at least one selected neighbor[1]), there first was a $335k$-vertex problem kernel [2], which was further refined into a $67k$-vertex problem kernel [8], both computable in cubic time.[2] In this previous work, the focus was on "engineering" data reduction rules in order to gain a small provable kernel size. Here, we aim at engineering the usage of known (and "established") ones in terms of improved time complexity instead of heading for new data reduction rules.

Following up the work on problem kernels for Dominating Set on planar graphs, we shift the focus from improving the kernel size to improving the running time (from cubic to linear) while still maintaining a linear-size problem kernel. Since this turns out to be a demanding task; for the sake of improving readability, we do not measure the constant factor for the problem kernel size.[3] In this sense, our work parallels other classification work related to Dominating Set, where the goal was to extend the considered graph class and/or class of problems [7, 12, 13]. Finally, we conjecture and already found some evidence that our data reduction approach also extends to other problems on planar graphs [13], again leading to linear-time linear-size problem kernels.

A similar result was achieved with a different approach by Hagerup [15]. He presents a linear problem kernel that can be computed in linear time by

[1] In what follows, knowledge of the parameter value k, that is, the maximum allowed size of a dominating set, will not be explicitly used in our algorithms. However, to formulate our results in a parameterized algorithmics setting, we need the parameter k.

[2] Experimental work showed that the corresponding data reduction rules are useful on several real-world data sets [3].

[3] A more or less standard analysis leads to large constants—however, on the one hand, with a more refined analysis they can be significantly improved and on the other hand, it is just a worst-case bound saying little about the effect on real-world instances. The goal of this paper is clearly of classification nature, affirmatively answering a question posed independently by Jiong Guo and Saket Saurabh at *WorKer'2010* held in Leiden, Netherlands; that is, a linear-time linear-size kernel for Dominating Set in planar graphs is possible.

providing completely new reduction rules that are tailored for DOMINATING SET. On the one hand, this may lead to smaller constants in the running time or the kernel bound and less complex analysis thereof. On the other hand, our approach of recycling the old reduction rules of Alber et al. [2] has the advantage of greater versatility: It can likely be applied to a variety of NP-hard problems on planar graphs (see [13]) and bears the possibility of improving the kernel-size in analogy to the work of Chen et al. [8] and Wang et al. [16].

Our Contributions. Revisiting previous data reduction rules for DOMINATING SET on planar graphs [2], we shift focus to the execution time of data reduction, improving it from cubic to linear time. To this end, we "rework" the known rules and their mathematical analysis and carefully analyze their interaction. Our central observation is that one can significantly gain efficiency by a non-exhaustive application of data reduction rules. More specifically, implementing the known data reduction rules [2] in the natural and straightforward way would "unavoidably" lead to cubic running time: The reason for this is that one has to inspect a quadratic number of so-called (potential) regions that a planar embedding of the graph may have. Thus, one of our major technical contributions is to restrict the region decomposition concept in such a way that the inspection (and, thus, the data reduction) can be done much faster. In this way, we achieve an $\mathrm{O}(k)$-vertex problem kernel for DOMINATING SET on planar graphs in linear time. Notably, our kernel size analysis is not as fine-grained as the previous ones [2, 8], meaning that we did not analyze the constant factor for the upper bound on the number of problem kernel vertices. Note, however, since multiple kernelization algorithms can be run on top of each other (this makes them quite different from approximation algorithms), using our algorithm as spear-head in combination with Chen et al.'s algorithm [8], for an n-vertex planar graph we can trivially achieve a problem kernel with $67k$ vertices in $\mathrm{O}(n + k^3)$ time, somewhat attenuating the quest for a sharper kernel size analysis. Due to the lack of space, most proofs are deferred to a full version of the paper.

Notation. We only consider undirected graphs $G = (V, E)$, where $V(G) := V$ is the set of vertices and $E(G) := E$ is the set of edges. Furthermore, let $n := |V|$ and $m := |E|$. The *open neighborhood* $N^G(v)$ of a vertex $v \in V$ in G is the set of vertices that are adjacent to v in G. The *closed neighborhood* $N^G[v]$ is $N^G(v) \cup \{v\}$. For a vertex set $S \subseteq V$ we set $N^G(S) := \bigcup_{v \in S} N^G(v)$. We use the *joint neighborhood* $N^G(v, w)$ of two

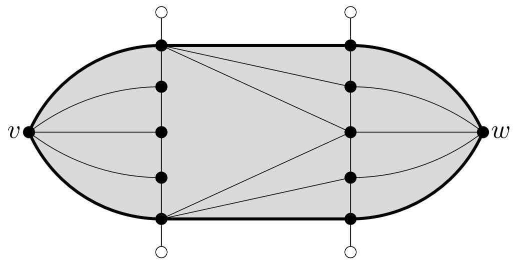

Fig. 1. Illustration of a region. The boundary path is shown as path of bold edges.

vertices to denote $(N^G(v) \cup N^G(w)) \setminus \{v, w\}$ and the *closed joint neighborhood* $N^G[v, w] := N^G[v] \cup N^G[w]$. A v_1-v_ℓ*-path* in G is a sequence $\mathcal{P} := (v_1, v_2, \ldots, v_\ell) \in V^\ell$ of vertices with $\{v_i, v_{i+1}\} \in E$ for $i \in \{1, \ldots, \ell - 1\}$ and $v_i \neq v_j$ for $i \neq j$, where $\ell - 1$ is the *length* of the path. We use $V(\mathcal{P})$ to denote the set of vertices of the path $\mathcal{P}$. We call two vertices v and w *connected* in G if there is a v-w path in G. We use $\operatorname{dist}^G(v, w)$ to denote the length of a shortest path between v and w in G, also called *distance*. The superscript G is omitted if G is clear from the context. The *domination number* $\gamma(G)$ of a graph G is the size of a smallest dominating set of G.

For a language $L \subseteq \Sigma^* \times \mathbb{N}$, a *kernelization algorithm* takes as input an instance (x, k), where k is called *parameter* and, computes in time polynomial in $|x|+k$ an instance (x', k') such that $(x', k') \in L \Leftrightarrow (x, k) \in L$, $|x'| \leq f(k)$, and $k' \leq k$. Here, f is a computable function solely depending on k which measures the *size* of the *problem kernel* (x', k').

2 Comparison to Previous Kernelizations

To obtain a linear-size problem kernel for Dominating Set on planar graphs, we employ a framework developed by Alber et al. [2]. They showed that a planar graph G with domination number $\gamma(G)$ can be decomposed into $\mathrm{O}(\gamma(G))$ so-called "regions". Data reduction ensures that each of these regions has constant size and that $\mathrm{O}(\gamma(G))$ vertices are not contained in any region. We follow a similar approach, modifying data reduction rules to run in linear time.

Basically, a region R of a planar graph G is a part of an embedding of G into the plane. Each region contains two vertices v and w such that $N[v, w]$ contains all vertices of R and a boundary that separates R from the rest of G, as illustrated in Figure 1.

Definition 1. Let G be a plane graph.[4] A *region* $R(v, w)$ between two vertices v and w is a closed bounded subset S of the plane such that:

1. the boundary of $R(v, w)$ is formed by two simple paths[5] between v and w, each of which has length at most three and
2. all vertices inside S are from $N[v, w]$.

We denote by $R(v, w)$ also the set of vertices in a region $R(v, w)$ and by $\partial R(v, w)$ the set of vertices on the boundary paths of a region $R(v, w)$. The vertices in $R(v, w) \setminus \partial R(v, w)$ are the *inner vertices* of $R(v, w)$.

Alber et al. [2] showed that each dominating set D of a planar graph G yields a so-called maximal D-region decomposition with $\mathrm{O}(|D|)$ regions.

Definition 2. For a plane graph G and $D \subseteq V(G)$, a *D-region decomposition* of G is a set $\mathcal{R}$ of regions between pairs of vertices in D such that

1. $\forall v, w \in D$ and $R(v, w) \in \mathcal{R}$, it holds that $D \cap R(v, w) = \{v, w\}$ and
2. for two distinct regions $R_1, R_2 \in \mathcal{R}$, it holds that $(R_1 \cap R_2) \subseteq (\partial R_1 \cap \partial R_2)$.

For a D-region decomposition $\mathcal{R}$, we define $V(\mathcal{R}) := \bigcup_{R \in \mathcal{R}} R$. A D-region decomposition $\mathcal{R}$ is *maximal* if there is no region $R \notin \mathcal{R}$ such that $\mathcal{R}' := \mathcal{R} \cup \{R\}$ is a D-region decomposition with $V(\mathcal{R}) \subsetneq V(\mathcal{R}')$.

Using data reduction, Alber et al. [2] shrink to constant size all regions that may potentially be part of a D-region decomposition for a minimum dominating set D of the input graph G. Since $|D| = \gamma(G)$, such a region decomposition comprises $\mathrm{O}(\gamma(G))$ regions. Together with an $\mathrm{O}(\gamma(G))$-bound on the number of vertices that are not in regions, this shows the linear size of the kernel.

Our goal is to modify the data reduction rules of Alber et al. [2] so that they can be applied in linear time instead of cubic time. Unfortunately, we have to make some sacrifices regarding the effectiveness of the data reduction rules, which we explain in Section 3. Alber et al. [2] employ two data reduction rules: one that shrinks the neighborhood of vertices, and one that shrinks regions. To shrink the neighborhood of vertices, we use a slightly modified version of Reduction Rule 1 of Alber et al. [2] that can be applied exhaustively in linear time. To shrink regions, we show that we can find all regions whose inner vertices have at most a constant number of neighbors. Furthermore, we provide means to ensure that regions do not contain vertices with more neighbors. This enables us to find and shrink regions in linear time.

[4] A plane graph is a particular embedding of a planar graph.
[5] This also includes degenerated cases where the two paths have common vertices.

3 Data Reduction Rules

In this section, we first describe two data reduction rules and show that they are *correct*, that is, they maintain planarity and do not change the domination number of the input graph. Then, we show how to execute them in linear time. Whenever we introduce new vertices into a graph, we call them *dummy vertices*. Moreover, we assume that our data reduction rules can check in $O(1)$ time whether a vertex is a dummy vertex. This can be achieved by marking dummy vertices accordingly. Note that these marks will be removed from the final output graph in order to obtain a proper Dominating Set instance (where unmarked graphs are required as input).

3.1 Private Neighborhood Rule

As Alber et al. [2], we partition the neighborhood of a vertex v in a graph G into three subsets:

$$\begin{aligned}
N_1^G(v) &:= \{u \in N^G(v) \mid N^G(u) \setminus N^G[v] \neq \emptyset\},\\
N_2^G(v) &:= \{u \in N^G(v) \setminus N_1^G(v) \mid N^G(u) \cap N_1^G(v) \neq \emptyset\},\\
N_3^G(v) &:= N^G(v) \setminus (N_1^G(v) \cup N_2^G(v)).
\end{aligned}$$

We now give our variant of Rule 1 of Alber et al. [2] for planar graphs.

Reduction Rule 1. Let $v \in V(G)$ be a vertex such that $|N_3^G(v)| > 1$ or $|N^G(N_3^G(v))| > 1$. Then, remove $N_3^G(v)$ from G and attach a new degree-one dummy vertex v' to v.

The correctness of Reduction Rule 1 follows from the correctness of Reduction Rule 1 of Alber et al. [2], as we only delete a subset of the vertices of which each was shown to be safely removable by Alber et al. [2]. By removing only a subset of the removable vertices, we can show that, for an exhaustive application of Reduction Rule 1, it is sufficient to apply Reduction Rule 1 once for every vertex. In this way, we can prove Lemma 1 below. In the following, we say that a graph G is *reduced* with respect to Reduction Rule 1 if Reduction Rule 1 is not applicable to G.

Lemma 1. *For planar graphs, Reduction Rule 1 can be applied exhaustively in* $O(n)$ *time.*

Proof. Let G be a planar graph, and let $v \in V(G)$ be a vertex that does not satisfy the conditions of Reduction Rule 1, that is, neither $|N_3^G(v)| > 1$ nor $|N^G(N_3^G(v))| > 1$. We show that, in the graph G' that results from applying Reduction Rule 1 to a vertex $u \in V(G) \setminus \{v\}$, the vertex v still does not satisfy the conditions of Reduction Rule 1. This implies that, in order to apply Reduction Rule 1 exhaustively to G, it is sufficient to apply Reduction Rule 1 at most once to each vertex. As shown by Alber et al. [2, Lemma 2], this can be done in $\mathrm{O}(n)$ time for planar graphs.

Towards a contradiction, assume that Reduction Rule 1 is applicable to v in G'. Then, because Reduction Rule 1 does not add edges between vertices in $V(G)$, it must hold that $N_3^{G'}(v) \cap (N_1^G(v) \cup N_2^G(v)) \neq \emptyset$. However, we show that the contrary is true. To this end, recall that each vertex in $N_2^G(v)$ is adjacent to a vertex in $N_1^G(v)$. Thus, in order to show $N_3^{G'}(v) \cap (N_1^G(v) \cup N_2^G(v)) = \emptyset$, it is sufficient to show that for each vertex $x \in N_1^G(v)$ it holds that $N^G[x] \cap N_3^{G'}(v) = \emptyset$. We distinguish the following two cases:

First, assume that $x \in N_2^{G'}(v) \cup N_3^{G'}(v)$. This is only true if Reduction Rule 1, when applied to u, deletes all neighbors of x that are nonadjacent to v. Let $y \in N_3^G(u) \cap N^G(x) \setminus N^G[v]$ be one such neighbor. Since $y \in N_3^G(u)$, we know that $x \in N^G[u] \setminus N_1^G(u)$ and thus $N^{G'}[x] \subseteq N^{G'}[u]$. Hence, u is adjacent to v. Because u has a degree-one dummy neighbor in G', no vertex from $N^{G'}[u]$ is contained in $N_3^{G'}(v)$. Since $N^G(x) \cap V(G') \subseteq N^{G'}[u]$, this implies $N^G[x] \cap N_3^{G'}(v) = \emptyset$.

In the second case, assume that $x \notin N_2^{G'}(v) \cup N_3^{G'}(v)$. Thus, we only have to show that the vertices in $N^G(x)$ are not in $N_3^{G'}(v)$. If $x \in N_1^{G'}(v)$, then, obviously, none of the vertices in $N^G(x)$ is in $N_3^{G'}(v)$. Hence, consider the subcase where $x \notin N_1^{G'}(v)$. This implies $x \notin N^{G'}(v)$ and thus x would have been deleted by Reduction Rule 1. Therefore, we have $x \in N_3^G(u)$, that is, $N^G[x] \subseteq N^G[u]$, implying $v \in N^G(u)$. Again, because u has a degree-one dummy neighbor in G', no vertex from $N^{G'}[u]$ is contained in $N_3^{G'}(v)$, implying $N^G[x] \cap N_3^{G'}(v) = \emptyset$. $\square$

3.2 Joint Neighborhood Rule

In this section, we present a data reduction rule that shrinks regions to constant size. The presented data reduction rule is based on Reduction Rule 2 by Alber et al. [2]. However, we modify it as follows: Reduction Rule 2 of Alber et al. [2] removes certain vertices from the sets $N(v, w)$ for vertices $v, w \in V$. Since we cannot compute $N(v, w)$ for all vertex pairs $v, w \in V$ in linear time, we will show that it is sufficient to only remove vertices from efficiently-computable subsets $N_0(v, w) \subseteq N(v, w)$ for a linear number of vertex pairs. More specifically, $N_0(v, w)$ contains vertices on short low-degree v-w-paths.

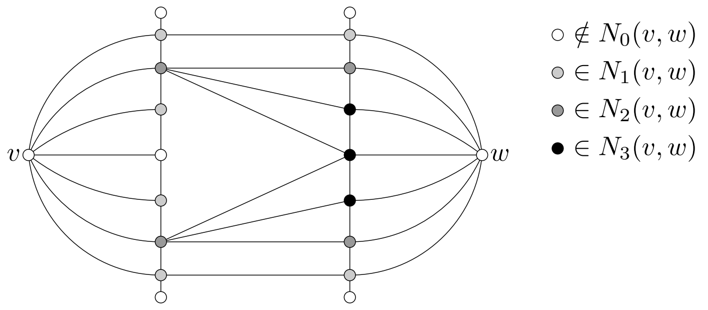

Fig. 2. An illustration of Definition 4.

Definition 3. A vertex v with $\deg(v) \leq 78$ is called *low-degree vertex.*[6] A v-w-path consisting only of v, w, and low-degree vertices is called *low-degree path.*

Note that all low-degree paths of constant length c starting at some vertex v can be found in $\mathrm{O}(\deg(v))$ time by starting a breadth-first search at v, only descending on low-degree vertices and stopping at depth c. It can be checked in $\mathrm{O}(1)$ time whether a vertex is a low-degree vertex by iterating over its adjacency list, aborting when a 79th neighbor is found.

Observation 1. For a vertex v, all constant-length low-degree paths starting at v can be listed in $\mathrm{O}(\deg(v))$ time.

To present our data reduction rule, we need the following definition of joint neighborhoods. It strongly resembles the definition used by Alber et al. [2, Section 2.2]. However, our sets $N_i^G(v,w)$ for $i \in \{1,2,3\}$ are defined with respect to $N_0^G(v,w)$ instead of $N^G(v,w)$. The following definition is illustrated in Figure 2.

Definition 4. Let v, w be vertices in a planar graph G. We define

$$\begin{aligned}
N_0^G(v,w) &:= \{u \in N^G(v,w) \mid u \text{ is on a low-degree } v\text{-}w\text{-path of length at most four that only consists of vertices in } N^G[v,w]\},\\
N_1^G(v,w) &:= \{u \in N_0^G(v,w) \mid N^G(u) \setminus N_0^G[v,w] \neq \emptyset\},\\
N_2^G(v,w) &:= \{u \in N_0^G(v,w) \setminus N_1^G(v,w) \mid N^G(u) \cap N_1^G(v,w) \neq \emptyset\},\\
N_3^G(v,w) &:= N_0^G(v,w) \setminus (N_1^G(v,w) \cup N_2^G(v,w)).
\end{aligned}$$

[6] Herein, the constant 78 results from the mathematical analysis and can be improved using a more intricate analysis.

```
Algorithm 1: Reduce Vertices in Regions.
   Input: A planar graph $G = (V, E)$.
   Output: A planar graph $G' = (V', E')$.
1  compute $N_0^G(v, w)$ for all $v, w \in V$ with $N_0^G(v, w) \neq \emptyset$;
2  $G' \leftarrow G$;
3  foreach $(v, w)$ such that $v$ and $w$ are non-dummy vertices and $N_0^G(v, w) \neq \emptyset$ do
4      EnsurePaths($G', v, w, N_0^G(v, w)$);
5      $N_3 \leftarrow N_3^{G'}(v, w) \cap N_0^G(v, w)$;
6      if $|N_3| \geq 4$ and $N_3$ cannot be dominated by a single vertex $u \notin \{v, w\}$ then
7          if $N_3 \subseteq N^{G'}(v) \cap N^{G'}(w)$ then remove the vertices in $N_3 \setminus \{z, z'\}$
           from $G'$, for arbitrary $z, z' \in N_3$ with $(N^{G'}(z) \cap N^{G'}(z')) \setminus N_3 \subseteq \{v, w\}$
           and $\{z, z'\} \notin E(G')$;
8          else if $N_3 \subseteq N^{G'}(v)$ and $N_3 \nsubseteq N^{G'}(w)$ then remove $N_3$ from $G'$ and
           (unless already done before) attach a degree-one dummy vertex $v'$ to $v$;
9          else if $N_3 \nsubseteq N^{G'}(v)$ and $N_3 \subseteq N^{G'}(w)$ then remove $N_3$ from $G'$ and
           (unless already done before) attach a degree-one dummy vertex $w'$ to $w$;
10         else if $N_3 \nsubseteq N^{G'}(v)$ and $N_3 \nsubseteq N^{G'}(w)$ then remove $N_3$ from $G'$ and
           (unless already done before) attach a degree-one dummy vertex $v'$ to $v$
           and a new degree-one dummy vertex $w'$ to $w$;
11 return G'
```

Using Definition 4, we present our second data reduction rule in form of Algorithm 1, which we now explain. Algorithm 1 basically corresponds to Reduction Rule 2 of Alber et al. [2]. For each pair $v, w \in V$ with $N_0^G(v, w) \neq \emptyset$, Algorithm 1 removes a subset of $N_0^G(v, w)$ from the graph G and, if applicable, attaches degree-one dummy vertices to v or w. We first explain the data reduction between lines 5 and 10 and then explain the purpose of the EnsurePaths procedure called in line 4. The set N_3 introduced in line 5 of Algorithm 1 is the set of vertices that may possibly be removed. We will see that N_3 can be efficiently computed and updated. Moreover, the choice of N_3 ensures the correctness of the data

```
Procedure EnsurePaths($G', v, w, N_0^G(v, w)$)
   // for $x, u \in V(G')$, let $B(x, u) := \{u' \in V(G') \setminus \{x\} \mid \text{dist}^{G'-\{x\}}(u, u') \leq 2\}$
1  $N_3^{\text{ldv}}(v) \leftarrow \{u \in N_3^{G'}(v) \cap N_0^G(v, w) \mid B(v, u)$ only has low-degree vertices$\}$;
2  $N_3^{\text{ldv}}(w) \leftarrow \{u \in N_3^{G'}(w) \cap N_0^G(v, w) \mid B(w, u)$ only has low-degree vertices$\}$;
3  if $|N^{G'}(v)| > 1 \wedge N_3^{\text{ldv}}(v) \neq \emptyset$ then remove $N_3^{\text{ldv}}(v)$ from $G'$ and (unless
   $v$ already has one) attach a new degree-one dummy vertex $v'$ to $v$;
4  if $|N^{G'}(w)| > 1 \wedge N_3^{\text{ldv}}(w) \neq \emptyset$ then remove $N_3^{\text{ldv}}(w)$ from $G'$ and (unless
   $w$ already has one) attach a new degree-one dummy vertex $w'$ to $w$;
```

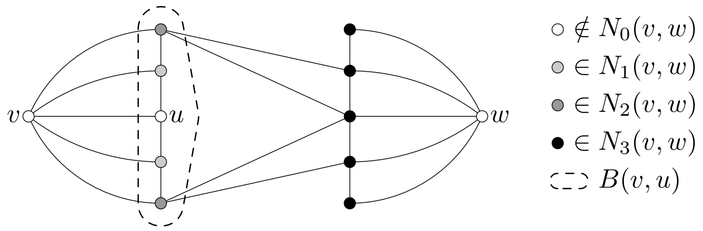

Fig. 3. The vertex u is not on a v-w-path of length at most four. Therefore, u and its neighbors are not deleted by a call of Algorithm 1. However, $u \in N_3(v)$. Hence, Reduction Rule 1 would delete u. Moreover, since $B(v,u)$ (surrounded by the dashed line) only contains low-degree vertices, EnsurePaths will delete u as well.

reduction executed by Algorithm 1, which can be seen by comparing it to Reduction Rule 2 of Alber et al. [2]: it is straightforward to observe that, if the condition in line 6 is satisfied, then the corresponding condition for Reduction Rule 2 of Alber et al. [2] is also satisfied. Moreover, in this case, we remove only a subset of vertices that Alber et al. [2] show to be safely removable. Note that the vertices z, z' chosen in line 7 exist:

Lemma 2. *Let v, w be vertices of a planar graph G' and let $N_3 \subseteq N_3^{G'}(v,w) \cap N^{G'}(v) \cap N^{G'}(w)$ with $|N_3| \geq 4$. Then, N_3 contains vertices z, z' with $\{z, z'\} \notin E(G')$ and $(N^{G'}(z) \cap N^{G'}(z')) \setminus N_3 \subseteq \{v, w\}$.*

We now explain the EnsurePaths procedure called in line 4 for a vertex pair (v,w). Observe that the graph G' considered in the for-loop in line 3 of Algorithm 1 is not necessarily reduced with respect to Reduction Rule 1 since previous iterations of the loop might have deleted vertices. However, an application of Reduction Rule 1 might become necessary: it might happen that some vertex $u \in N_0^G(v,w)$ is not in $N_0^{G'}(v,w)$ when the pair (v,w) is considered in line 3. Such a situation is illustrated in Figure 3 and could arise if all vertices on u's low-degree v-w-paths are deleted by data reduction executed for some other vertex pair. As in Figure 3, this could prevent u or some of its neighbors from being removed from G'. In the situation shown, Reduction Rule 1 would delete u. Hence, in order to ensure that a vertex in G' that is in $N_0^G(v,w)$ has a low-degree v-w-path in G', it could help to apply Reduction Rule 1 to v and w. However, doing so for each considered pair (v,w) might be too time-consuming. This is the reason why EnsurePaths is employed, which in lines 3 and 4 deletes a subset of $N_3^{G'}(v)$ and $N_3^{G'}(w)$ in a case where Reduction Rule 1 would completely delete $N_3^{G'}(v)$ and $N_3^{G'}(w)$. Namely, those vertices $u \in N_3^{G'}(v)$ (or $N_3^{G'}(w)$) are deleted for which $B(v,u)$ (or $B(w,u)$, respectively) only contains low-degree vertices—a condition which merely ensures that we can

Algorithm 2: Compute $N_0^G(v,w)$ for all $v,w \in V(G)$ with $N_0^G(v,w) \neq \emptyset$.

```
**Input**: A planar graph $G = (V, E)$ with vertices numbered $V := \{1, \dots, n\}$.
**Output**: $N_0^G(v,w)$ for all $v,w \in V$ with $N_0^G(v,w) \neq \emptyset$.
1  $\mathcal{D} \leftarrow$ empty list;   /* $(v,w,u) \in \mathcal{D}$ will be equivalent to $u \in N_0^G(v,w)$ */
2  for $v \in V$ and each low-degree path $p$ of length at most four starting at $v$ do
3      $w \leftarrow$ ending vertex of $p$;
4      if all vertices of $p$ are in $N^G[v,w]$ then
5          foreach vertex $u \in V \setminus \{v,w\}$ of $p$ do append $(v,w,u)$ to $\mathcal{D}$;
6  sort $\mathcal{D}$ in lexicographical order using radix sort;
7  foreach $(v,w,u) \in \mathcal{D}$ in lexicogr. order do   /* collect $N_0^G(v,w)$ from $\mathcal{D}$ */
8      $(v',w',u') \leftarrow$ previous element in $\mathcal{D}$;
9      if $v \neq v' \vee w \neq w'$ then   /* we encounter the pair $(v,w)$ the first time */
10         new set $N_0^G(v,w) \leftarrow \{u\}$;
11     else if $u \neq u'$ then add $u$ to $N_0^G(v,w)$;   /* avoids duplicates */
12 return $N_0^G(v,w)$ for all $v,w \in V$ with $N_0^G(v,w) \neq \emptyset$;
```

efficiently check whether $u \in N_3^{G'}(v)$ or $u \in N_3^{G'}(w)$. Since EnsurePaths deletes only vertices which Reduction Rule 1 deletes, EnsurePaths is correct. Moreover, observe that, in Figure 3, the vertex u would be deleted from G' by EnsurePaths.

To execute Algorithm 1 one can compute all sets $N_0^G(v,w)$ in linear time using Algorithm 2. Also, one frequently has to check whether a vertex is in $N_3^{G'}(v,w)$. The following lemma shows that this can be done efficiently.

Lemma 3. *For a planar graph G and vertices u, v, w of G, it is $\mathrm{O}(1)$-time-decidable whether $u \in N_3^G(v,w)$. Moreover, all sets $N_0^G(v,w)$ can be enumerated in $\mathrm{O}(n)$ total time for all vertices $v, w \in V(G)$ with $N_0^G(v,w) \neq \emptyset$.*

Since we can efficiently check membership of a vertex in $N_3^G(v,w)$ and how to compute all sets $N_0^G(v,w)$, we have all ingredients to prove the running time of Algorithm 1.

Lemma 4. *On planar graphs, Algorithm 1 can be executed in $\mathrm{O}(n)$ time.*

4 Problem Kernel

This section presents our kernelization algorithm based on the data reduction rules shown in the previous section. We explain how, given a planar graph G, the algorithm computes a graph G' with $\gamma(G) = \gamma(G')$ whose size is linear in $\gamma(G)$. The kernelization algorithm runs in three phases. Each phase applies Reduction Rule 1 or Algorithm 1 to finally output G'.

Phase 1. Exhaustively apply Reduction Rule 1 to G. Let G_1 denote the resulting graph. By Lemma 1, G_1 is computable in $\mathrm{O}(n)$ time and is reduced with respect to Reduction Rule 1.

Phase 2. Apply Algorithm 1 to G_1, then Reduction Rule 1 exhaustively, then again Algorithm 1 and, finally, Reduction Rule 1 exhaustively. Let the result be denoted by G_2. By Lemmas 1 and 4, G_2 is computable in $\mathrm{O}(n)$ time. We will see that most vertices in G_2 have degree at most 78.

Phase 3. Apply Algorithm 1 and exhaustively apply Reduction Rule 1 to G_2, resulting in a graph G_3. Using the fact that in G_2 most vertices have at most 78 neighbors and that G_2 is reduced with respect to Reduction Rule 1, we can show that, using Algorithm 1, Phase 3 removes enough parts from G_2 to obtain a linear-size problem kernel.

To show that G_3 has size $\mathrm{O}(\gamma(G))$, we transfer structural observations from G_1 to the graphs G_2 and G_3 using a maximal D-region decomposition of G_1 with respect to an arbitrary embedding. Herein, we choose D as a dominating set of size at most $2\gamma(G)$ for G_1 that contains the set D' of vertices in $V(G_1) \cap V(G_3)$ to which dummy vertices have been attached by data reduction rules, that is, $D' := \{v \in V(G_1) \cap V(G_3) \mid N^{G_3}(v) \setminus V(G_1) \neq \emptyset\}$. Note that we may not be able to efficiently compute D. However, D and the embedding of G_1 are only required for the analysis. We first show that D exists and, thereafter, we show how a maximal D-region decomposition helps us to transfer structure from G_1 to G_2 and G_3.

Lemma 5. *There is a dominating set $D \supseteq D'$ for G_1 with $|D| \leq 2\gamma(G)$.*

For the remainder of this section, we base our reasoning on a fixed maximal D-region decomposition $\mathcal{R}$ of G_1 with $D \supseteq D'$. Notice that if a degree-one dummy vertex is attached to a vertex v, then neither Reduction Rule 1 nor Algorithm 1 can delete v. Instead, v is in G_3, where it also has a (possibly different) dummy neighbor. The choice of $\mathcal{R}$ and the definition of a D-region-decomposition for G_1 then leads to the following:

Observation 2. Let $R(v, w) \in \mathcal{R}$ be a region. Reduction Rule 1 and Algorithm 1 do not attach dummy vertices to any vertex in $R(v, w) \setminus \{v, w\}$.

Observation 2 ensures that, in none of the graphs G_1, G_2, and G_3, the inner vertices of a region $R \in \mathcal{R}$ have neighbors that are not in R. In this way, Observation 2, to a certain extent, preserves the region structure of G_1 in G_2 and G_3.

As shown by Alber et al. [2, Proposition 1], $|\mathcal{R}| \in \mathrm{O}(|D|) = \mathrm{O}(\gamma(G))$. The proof of the problem kernel size now works as follows: Proposition 1 shows that Phase 2 shrinks the number of vertices that are not in any region of $\mathcal{R}$ to $\mathrm{O}(\gamma(G))$.

Proposition 1. $|(V(G_1) \setminus V(\mathcal{R})) \cap V(G_2)| \in \mathrm{O}(\gamma(G))$.

Lemma 6 shows that, as a result of Phase 2, inner vertices of regions in G_1 that are also present in G_2 have constant degree in G_2.

Lemma 6. *Let $R(v,w) \in \mathcal{R}$, and let $u \in V(G_2) \setminus \{v,w\}$. If $\{v,w\} \subseteq V(G_2)$, then $|N^{G_2}(u) \cap R(v,w)| \leq 78$.*

Exploiting Lemma 6, Proposition 2 shows that, using Algorithm 1, Phase 3 shrinks each region $R \in \mathcal{R}$ to $\mathrm{O}(1)$ vertices.

Proposition 2. *Let $R(v,w) \in \mathcal{R}$. Then, $|V(G_3) \cap R(v,w)| \in \mathrm{O}(1)$.*

Finally, we bound the number of vertices added by our kernelization algorithm (dummy vertices). Since $|D'| \in \mathrm{O}(\gamma(G))$ is shown by Lemma 5 and to each such vertex at most one dummy vertex is added, it follows that there are at most $\mathrm{O}(\gamma(G))$ vertices in $V(G_3) \setminus V(G_1)$. We conclude that G_3 consists of $\mathrm{O}(\gamma(G))$ vertices, yielding our central theorem:

Theorem 1. *On planar graphs, a linear-size problem kernel for* DOMINATING SET *is computable in linear time.*

5 Conclusion

Our work is meant to provide a first case study, using the well-known kernelization of DOMINATING SET on planar graphs, on how known kernelization algorithms can be tuned for better time performance by carefully analyzing the interaction and costs of the underlying data reduction rules. Clearly, on the practical side it is important to further improve on the upper bound for the kernel size to be achieved in linear time.

Informally speaking, the analysis of our problem kernel shows that "knowing when to stop" may be a source of performance gain. This is particularly true for kernelization algorithms since faster algorithms with worse kernel bounds can be combined with slower algorithms yielding better kernel bounds in order to achieve an overall improvement.

As to future challenges, there are a lot of possibilities in revisiting known kernelization results and re-engineering them in terms of algorithmic efficiency. As to DOMINATING SET on planar graphs, we left open whether there is a linear-time problem kernel that can be achieved by an *exhaustive* application of data reduction rules (and not stopping their application for reasons of efficiency). In this work, we computed an $\mathrm{O}(\gamma(G))$-size problem kernel in $\mathrm{O}(n)$ time. For other problems, it might be helpful to alleviate the quest from $\mathrm{O}(n)$-time computability to $\mathrm{O}(p(k)+n)$-time or

O$(p(k) \cdot n)$-time computability, where p is a polynomial solely depending on a parameter k. For instance, Chor et al. [9] presented an almost linear-time kernelization for a variant of the Clique Covering problem. Finally, we leave it as a challenge for future research to investigate the complexity classes of fixed-parameter tractable problems that allow for kernels of *any* size computable in (almost) linear running time.

Bibliography

[1] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

[2] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[3] J. Alber, N. Betzler, and R. Niedermeier. Experiments on data reduction for optimal domination in networks. *Ann. Oper. Res.*, 146(1):105–117, 2006.

[4] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[5] M. Bateni, M. Hajiaghayi, and D. Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. In *Proc. 42th STOC*, pages 211–220. ACM Press, 2010.

[6] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

[7] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) kernelization. In *Proc. 50th FOCS*, pages 629–638. IEEE, 2009.

[8] J. Chen, H. Fernau, I. A. Kanj, and G. Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. *SIAM J. Comput.*, 37(4):1077–1106, 2007.

[9] B. Chor, M. R. Fellows, and D. W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proc. 30th WG*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004.

[10] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.

[11] M. R. Fellows, F. A. Rosamond, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? In *Proc. 21st IJCAI*, pages 486–491, 2009.

[12] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proc. 21st SODA*, pages 503–510. ACM/SIAM, 2010.

[13] J. Guo and R. Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proc. 34th ICALP*, volume 4596 of *LNCS*, pages 375–386. Springer, 2007.

[14] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

[15] T. Hagerup. Linear-time kernelization for planar dominating set. In *Proc. 6th IPEC*, LNCS. Springer, 2011. to appear.

[16] J. Wang, Y. Yang, J. Guo, and J. Chen. Linear problem kernels for planar graph problems with small distance property. In *Proc. 36th MFCS*, volume 6907 of *LNCS*, pages 592–603. Springer, 2011.