Parameterized Complexity of Finding Regular Induced Subgraphs[1]

Hannes Moser [a,2] Dimitrios M. Thilikos [b,3]

[a] *Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany*

[b] *Department of Mathematics, National and Capodistrian University of Athens Panepistimioupolis, GR-15784, Athens, Greece.*

Abstract

The r-Regular Induced Subgraph problem asks, given a graph G and a non-negative integer k, whether G contains an r-regular induced subgraph of size at least k, that is, an induced subgraph in which every vertex has degree exactly r. In this paper we examine its parameterization k-Size r-Regular Induced Subgraph with k as parameter and prove that it is $W[1]$-hard. We also examine the parameterized complexity of the dual parameterized problem, namely, the k-Almost r-Regular Graph problem, which asks for a given graph G and a non-negative integer k whether G can be made r-regular by deleting at most k vertices. For this problem, we prove the existence of a problem kernel of size $O(kr(r+k)^2)$.

1 Introduction

Regular graphs as well as regular subgraphs have been intensively studied from a structural point of view (e.g., [1]). An interesting problem related to regular graphs is to decide whether a given graph contains a regular subgraph. One of the first problems of this kind was stated by Garey and Johnson: Cubic

Email addresses: moser@minet.uni-jena.de (Hannes Moser), sedthilk@math.uoa.gr (Dimitrios M. Thilikos).

[1] A preliminary version of this paper appeared in the Proceedings of the 2nd Algorithms and Complexity in Durham (ACiD'06), Durham, England, September 2006, Volume 7 in Texts in Algorithmics, pages 107–118, College Publications.

[2] Supported by the EC Research Training Network HPRN-CT-2002-00278 (COMBSTRU) and by the Deutsche Forschungsgemeinschaft, project ITKO, NI 369/5.

[3] Supported by the Spanish CICYT project TIN-2004-07925 (GRAMMARS).

Subgraph, that is, the problem of deciding whether a given graph contains a 3-regular subgraph, is NP-complete [11]. This result was later expanded in [22], where it was shown that Cubic Subgraph is NP-complete on planar graphs with maximum degree 7. Moreover, it was shown by Stewart that Cubic Subgraph is also NP-complete on bipartite graphs with maximum degree 4 [24]. The same author showed that the more general problem of deciding whether a given graph contains an r-regular subgraph for any fixed degree $r > 3$ is NP-complete on general graphs as well as on planar graphs [23] (where in the latter case only $r = 4$ and $r = 5$ were considered, since any planar graph contains a vertex of degree at most 5). Note that this problem is polynomial-time solvable for $r \leq 2$ [5].

We consider a variant of this problem, where we ask whether a given graph G contains an *induced* subgraph of at least k vertices that is r-regular, and we call it r-regular induced subgraph for any $r \geq 0$. The *exact* version of this problem is obtained if we ask for an induced subgraph of size *exactly* k (for the difference between the original and the exact version of the problem, see Figure 1). In this paper, we examine the following dual parameterizations of the exact version of the problem.

(1) k-Almost r-Regular Graph:
Input: A graph $G = (V, E)$ and an integer k.
Parameter: k.
Question: Is there a vertex subset $S \subseteq V$ of size exactly k such that $G[V \setminus S]$, that is, the graph induced by $V \setminus S$, is r-regular?

(2) k-Size r-Regular Induced Subgraph
Input: A graph $G = (V, E)$ and an integer k.
Parameter: k.
Question: Is there a vertex subset $S \subseteq V$ of size exactly k such that $G[S]$, that is, the graph induced by S, is r-regular?

The non-exact version of the k-Almost r-Regular Graph problem asks for a set S of size at most k and is denoted as $\leqslant k$-Almost r-Regular Graph. Similarly, the non-exact version of the k-Size r-Regular Induced Subgraph problem asks for a set S of size at least k and is denoted as $\geqslant k$-Size r-Regular Induced Subgraph.

The r-Regular Induced Subgraph problem belongs to the general category of subgraph problems, i.e., problems that ask for the existence of an (induced) subgraph with a certain property (e.g., being r-regular). Most problems of this type concern hereditary properties (a property is *hereditary* if it holds for any induced subgraph of G whenever it holds for G) and can be classified as NP-hard [16,25]. Basically all such problems, when parameterized by the number of vertices that need to be removed in order to obtain the desired property, admit fixed-parameter algorithms provided that the corresponding property is

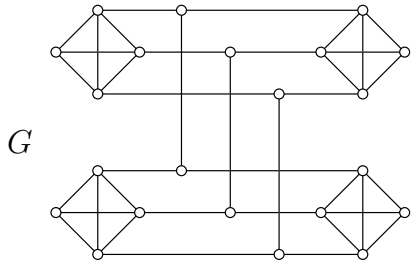

Fig. 1. For 3-Regular Induced Subgraph, the instance (G, k) is a yes-instance iff $k \in \{0, 1, \ldots, 17, 18\}$. However, for its "exact version", the same instance is a yes-instance iff $k \in \{4, 8, 12, 16, 18\}$.

hereditary [2,15]. For related results, we refer to [17], [10,21], and [4,6,13] where the imposed property is chordality, 2-colorability, and acyclicity, respectively.

Notice that r-regularity is not a hereditary property as subgraphs of r-regular graphs are generally not r-regular, for $r \geq 1$. This suggests that the existing results do not help to prove $W[1]$-hardness or fixed-parameter intractability for the problems we consider in this paper. Moreover, the lack of heredity makes it harder to design fixed-parameter algorithms. Another vertex removal problem with a non-hereditary property was examined in [7] where it is shown that the problem of converting a given graph into a grid by vertex/edge removals and additions is fixed-parameter tractable.

Since r-regularity is not a hereditary property it makes sense to consider the different exact and non-exact versions of the problems above (see Figure 1). As the non-exact versions of the problems are weaker than (can be reduced to) their exact counterparts, we will prove all of our hardness results for the above weaker setting. However, in order to obtain a stronger statement, all algorithms in this paper are designed for the "exact" versions of our problems (modifications for solving the original versions are easy and have the same running times).

Note that for $r = 0$, r-Regular Induced Subgraph is equivalent to Independent Set. In this paper, we show that r-Regular Induced Subgraph is NP-complete for $r \geq 1$ even if we restrict the input of the problem to planar graphs or triangle-free planar graphs (for planar graphs, we consider $r \leq 5$ and for triangle-free planar graphs we consider $r \leq 3$)[4]. Our reduction also implies that $\geqslant k$-Size r-Regular Induced Subgraph is $W[1]$-hard for any $r \geq 0$ on general graphs, suggesting that no fixed-parameter algorithm exists for this parameterized problem (and therefore, neither for its exact counterpart). However, the exact version of the dual problem, k-Almost r-Regular Graph, is easier. We design a fixed-parameter algorithm running in $O(n(k + r) + kr^2(k + r)^2 \cdot (r + 2)^k)$ steps, where n is the number of vertices of the input graph. Our algorithm is based on a search tree algorithm

[4] This restriction follows from Euler's formula.

and the existence of a reduction of k-Almost r-Regular Graph to a problem kernel of size $O(kr(k+r)^2)$ (in the special case where $r = 1$, this size can be improved to $O(k^2)$). We stress that this positive result only holds for constant r, it has been shown very recently by Mathieson and Szeider that $\leqslant k$-Almost r-Regular Graph becomes $W[1]$-hard if r is part of the input [18]. The authors also consider a weighed variant of the problem, where the desired degree can be specified for each vertex separately, and they also consider the operations edge addition and/or edge deletion and show several hardness and fixed-parameter tractability results for these variants [18,19].

The remaining document is organized as follows: First, we shortly introduce necessary definitions from parameterized complexity theory and graph theory in Section 2. Section 3 is dedicated to hardness results. In Sections 4 and 5 we present the problem kernel and the exact algorithm for k-Almost r-Regular Graph, respectively.

2 Preliminaries

In this paper we deal with fixed-parameter algorithms that emerge from the field of parameterized complexity analysis [8,20], where the computation complexity of a problem is analyzed in a two-dimensional framework. One dimension of an instance of a parameterized problem is the input size n, and the other the *parameter* k. A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function depending only on the parameter k, not on the input size $|I|$. One of the common methods to prove that a problem is fixed-parameter tractable is to provide *data reduction rules* that lead to a *problem kernel*: Given a problem instance (I, k), a data reduction rule replaces that instance by a another instance (I', k') in polynomial (with respect of the size of the input) time, such that (I, k) is a yes-instance iff (I', k') is a yes-instance. An instance to which none of a given set of reduction rules applies is called *reduced* with respect to the rules. A parameterized problem is said to have a problem kernel if, after the application of the reduction rules, the resulting reduced instance has size $f(k)$ for a function f depending only on k. For more about problem kernelization, we refer to a recent survey by Guo and Niedermeier [14]. Analogously to classical complexity theory, Downey and Fellows [8] developed a framework providing a reducibility and completeness program. A *parameterized reduction* from a parameterized language L to another parameterized language L' is a function that, given an instance (I, k), returns in time $f(k) \cdot n^{O(1)}$ an instance (I', k') (with k' depending only on k) such that $(I, k) \in L \Leftrightarrow (I', k') \in L'$. The basic complexity class for fixed-parameter intractability is $W[1]$ as there is good reason to believe that $W[1]$-hard problems are not fixed-parameter tractable [8].

In this paper we assume that all graphs are simple and undirected. For a graph $G = (V, E)$ we write $V(G)$ to denote its vertex set and $E(G)$ to denote its edge set. By default, we use n to denote the number of vertices of a given graph. For a subset $V' \subseteq V$, by $G[V']$ we mean the subgraph of G induced by V'. We write $G \setminus V'$ to denote the graph $G[V \setminus V']$. If $v \in V$ we also write $G - v$ instead of $G \setminus \{v\}$. The *(open) neighborhood* $N(V')$ of a given set $V' \subseteq V$ is the set of all vertices in $V \setminus V'$ adjacent to some vertex in V'. We sometimes write $N_G(V')$ to emphasize that we refer to the open neighborhood of V' within the graph G. We write K_r to denote the complete graph with r vertices.

Given a graph $G = (V, E)$ and an edge subset $E' \subseteq E$, to *subdivide* the edges E' in G means to remove in G all edges in E', and then to add for each edge $\{u, v\} \in E'$ a vertex $x_{u,v}$, making it adjacent to u and v. The vertices in $\{x_{u,v} \mid \{u, v\} \in E'\}$ are called *subdivision vertices*.

In the next section, we show that $\leqslant k$-Almost r-Regular Graph as well as $\geqslant k$-Size r-Regular Induced Subgraph are NP-complete. Moreover, we show that the latter problem is also $W[1]$-hard.

3 Hardness and Completeness Results

In this section we first prove that r-Regular Induced Subgraph is NP-complete by giving a polynomial-time reduction from Vertex Cover. A similar independent result has recently appeared in [3] proving the NP-hardness of finding an induced r-regular bipartite graph.

Theorem 1 *The r-Regular Induced Subgraph problem is NP-complete for any $r \geq 0$. It also remains NP-complete when restricted to planar graphs (for $r \leq 5$) or to triangle-free planar graphs (for $r \leq 3$).*

Proof. We first prove the theorem in its general statement and then we explain how to modify the proof for its planar versions. For the proof we use $\leqslant k$-Almost r-Regular Graph, that is, we search for a vertex subset S of size at most k such that $G \setminus S$ is r-regular. This problem is polynomially equivalent to r-Regular Induced Subgraph. For $r = 0$, $\leqslant k$-Almost r-Regular Graph is identical to Vertex Cover, which is known to be NP-complete. For all remaining $r > 0$ we give a reduction from Vertex Cover.

Let (G, k) be an instance of Vertex Cover. We construct an instance (G', k') of $\leqslant k$-Almost r-Regular Graph with $r > 0$ as follows: First, we set $G' := G$ and $k' := k \cdot (r + 1)$. For each vertex $v \in V(G)$ we add a copy of K_{r+1} to G'. Let R_v be the copy corresponding to vertex v. For all vertices $v \in V(G)$ we

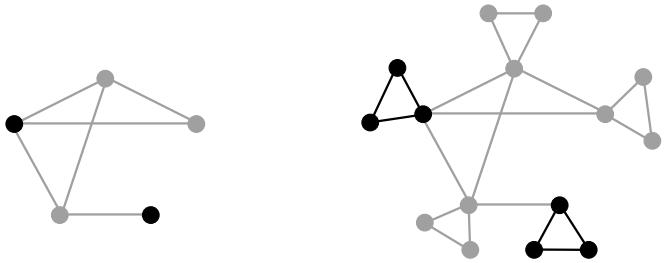

Fig. 2. Example of a graph G (left) and the corresponding graph G' (right) with $r = 2$. Vertices in the solution are gray, the remaining vertices which are not removed are black.

identify v with an arbitrary vertex in R_v, that is, we set $v = w$ for some arbitrary $w \in V(R_v)$. Figure 2 gives an example of a graph G and the corresponding graph G'.

We have to show that (G, k) with $r = 0$ is a yes-instance iff (G', k') with $r > 0$ is a yes-instance.

($\Rightarrow$): Suppose that there is a size-k solution S for (G, k), that is, $G \setminus S$ consists of isolated vertices. We define a new solution set $S' := \bigcup_{v \in S} V(R_v)$ of size $k \cdot (r+1)$ for G'. Clearly, $G' \setminus S'$ is a graph in which every connected component is an R_v, i.e., $G' \setminus S'$ is an r-regular graph, thus S' is a solution for (G', k').

($\Leftarrow$): Suppose that there is a size-k' solution S' for (G', k'). We say that S' is *clustered* if

$$\forall v \in V(G) : R_v \cap S' \neq \emptyset \Rightarrow R_v \subseteq S',$$

and notice that if S' is clustered then $S := \{v \in V(G) \mid R_v \cap S' \neq \emptyset\}$ is a solution for the instance (G, k) of Vertex Cover. In the case that the solution S' is not clustered, we can turn it into a clustered one according to the following claim, which completes the proof of correctness of the reduction.

Claim: *Given a solution S' for (G', k') where $|S'| \leq k'$, we can always construct a clustered solution S'' for the same problem instance.*

Proof of claim: We first show that $G' \setminus S'$ is an r-regular graph in which each connected component is either a R_v or an r-regular subgraph containing vertices exclusively from G. Recall that $G' \setminus S'$ is r-regular. If $v \in S'$ for some $v \in G$, then we know that $R_v \subseteq S'$, as otherwise some vertices in R_v would have degree smaller than r in $G' \setminus S'$. The same argument shows that if $v \notin S'$, then either $R_v \cap S' = \emptyset$ or $(R_v - v) \subseteq S'$. The first case implies that every neighbor w of v in G is in the solution S' (and therefore R_w likewise), since otherwise v would have a degree greater than r in $G' \setminus S'$. The second case implies that r neighbors of v in G must not be in the solution S', since otherwise v would have a degree smaller than r in $G \setminus S'$.

With these observations we can prove the claim as follows. Assume that $G' \setminus S'$ contains some connected components which are subgraphs consisting only of vertices from G. Let A be the set of vertices of all such connected components of $G' \setminus S'$. As $G'[A]$ is an r-regular graph, it will contain an independent set I of size at least $\lceil \frac{|A|}{r+1} \rceil$ (I is constructed by greedily picking vertices and removing their neighbors). We set $S'' = (S' \cup (A \setminus I)) \setminus \{R_v \mid v \in I\}$ and we observe that $G' \setminus S''$ is also an r-regular graph where each connected component is a R_v. To show that S'' is a solution for (G', k') it remains to prove that $|S''| \leq k$. Observe that the above modification added at most $|A| - \lceil \frac{|A|}{r+1} \rceil$ vertices in the solution and removed at least $r \cdot \lceil \frac{|A|}{r+1} \rceil$ vertices in it. Using the tautological relation $\frac{|A|}{1+r} \leq \lceil \frac{|A|}{1+r} \rceil$, which can be rewritten as $|A| - \lceil \frac{|A|}{r+1} \rceil - r \cdot \lceil \frac{|A|}{r+1} \rceil \leq 0$, we get $|S''| - |S'| \leq |A| - \lceil \frac{|A|}{r+1} \rceil - r \cdot \lceil \frac{|A|}{r+1} \rceil \leq 0$. Hence $|S''| \leq |S'| \leq k$, and this completes the proof of the claim.

Vertex Cover remains NP-complete when restricted to triangle-free planar graphs [12]. Therefore, the above proof also implies that r-Regular Induced Subgraph remains NP-complete even when we restrict it to planar graphs for $r \leq 5$. The only modification is that, for the cases where $r = 4$ or $r = 5$, we attach to G a graph corresponding to an octahedron or an icosahedron instead of K_5 or K_6, respectively. Moreover, the same reduction implies also the NP-completeness for triangle-free planar graphs for $r \leq 3$ when, in the case $r = 3$, we replace K_4 by the cube. □

It is known that the parameterized version of Independent Set (the dual problem of Vertex Cover), where the parameter is the size of the independent set, is $W[1]$-hard [9]. The reduction in the above proof can be used to show the $W[1]$-hardness of $\geqslant k$-Size r-regular induced subgraph as follows. Let (G, k) be an instance of Independent Set. It can be regarded as an instance $(G, n-k)$ of Vertex Cover. This instance is reduced to an instance $(G', (n-k)(r+1))$ of $\leqslant k$-Almost r-Regular Graph, which can be regarded as an instance $(G', n(r-1) - (n-k)(r+1)) = (G', k(r+1))$ of $\geqslant k$-size r-regular induced subgraph. Clearly, this is a parameterized reduction, as all these steps can be performed in time $f(k) \cdot n^{O(1)}$, and the parameter $k' = k(r+1)$ is only depending on k. We arrive at the following theorem.

Theorem 2 $\geq k$-Size r-regular induced subgraph *is* $W[1]$*-hard.*

4 A Problem Kernel

A central ingredient for the problem kernel is the notion of a clean region.

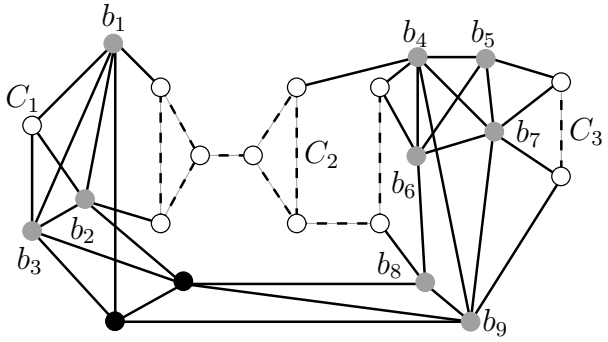

Fig. 3. Example of a graph with clean regions C_1, C_2, and C_3 (white vertices, $r = 3$). The dotted edges denote the connected subgraph each clean region induces. Dirty vertices are gray or black, boundary vertices are gray and all other dirty vertices are black. The boundary for C_1 is $B_1 = \{b_1, b_2, b_3\}$, the boundary for C_2 is $B_2 = \{b_1, b_2, b_4, b_6, b_8\}$, and the boundary for C_3 is $B_3 = \{b_5, b_7, b_9\}$. Note that boundaries can have vertices in common, for instance, $B_1 \cap B_2 \neq \emptyset$.

Definition 1 *We call a vertex of G* clean *if it has degree r, and* dirty *otherwise. We define a* clean region *in G as a maximal subset of clean vertices that induces a connected subgraph in G.*

Let $\{C_i : i \in I\}$ be the set of all clean regions. The open neighborhood of each clean region C_i is called its *boundary* B_i (notice that two different boundaries may share common vertices). A clean region C_i is called *isolated* if $B_i = \emptyset$. Observe that the neighborhood of a non-isolated clean region consists entirely of dirty vertices. See Figure 3 for examples of clean regions and their boundaries. The detection of all clean regions in G can be done in $O(nr)$ steps.

The main result of this section is the following.

Theorem 3 *The k-Almost r-Regular Graph problem, for $r \geq 1$, has a kernel with $O(kr(k+r)^2)$ vertices, which can be constructed in $O(n \cdot (k+r))$ time.*

Proof. The idea of our kernelization is to apply a series of reduction steps to the input instance that either give a negative answer or produce a new equivalent instance satisfying a bigger subset of the following properties.

(1) All vertices in G have degree at least r and at most $k + r$,
(2) each vertex of a boundary B_i has at most r clean neighbors in C_i,
(3) the isolated clean regions of G contain in total at most k^2 vertices,
(4) for every clean region C_i with boundary B_i,

$$|C_i| \leq (r+1) \cdot (1 + \max\{\lceil \frac{k+1}{r+1} \rceil, |B_i|\}).$$

Then we will prove that if all the above properties hold for an instance of k-Almost r-Regular Graph, then the size of the instance is the claimed one. For our presentation, we will use $(K_r, 1)$ as the no-instance of k-Almost r-Regular Graph for $r \geq 0$. We also suppose that the graph is stored using adjacency lists. We proceed with the first reduction step.

Step 1:

1. While $k \geq 0$ and $\exists_{v \in V(G)}\ (\deg_G(v) < r) \vee (\deg_G(v) > k + r)$:
   $(G, k) \leftarrow (G \setminus \{v\}, k - 1)$.
2. If $k \geq 0$, then return (G, k), otherwise, return $(K_r, 1)$.

Consider an instance (G, k) of k-Almost r-Regular Graph. Vertices v in G with $\deg(v) < r$ obviously must be contained in the solution S. Likewise, vertices v with degree $\deg(v) > k + r$ must be in S, as we would have to put more than k of its neighbors into S to achieve degree r for v. We conclude that **Step 1** produces an equivalent instance satisfying property (1).

For the next step, observe that taking a vertex of a clean region into the solution S causes its clean neighbors to have a degree less than r in $G \setminus S$, forcing them into the solution as well. By applying the same argument inductively to the clean neighbors, we can see that either no vertex of a clean region is a part of the solution S, or the entire clean region is contained in S.

We now briefly comment on the data structure supporting the implementation of the first step. First of all, we may assume from the beginning that $|E(G)| \leq n(k + r))$, otherwise G contains a subgraph of minimum degree $> k + r$ and in this case we know in advance that the input graph is a no-instance. We construct an auxiliary data structure as follows. We create an array A of length n with entries from 0 to $n - 1$, where each entry i points to a linked list L_i. The entries of L_i correspond to the vertices of G that have degree i in G and contain pointers to these vertices in the adjacency list structure. Also each vertex v in the adjacency list structure points back to the entry of $L_{\deg(v)}$ that points to it. This structure can be built on the top of the adjacency list structure in $O(|E(G)|) = O(n(k + r))$ time. It is now easy to verify that, using this enhanced data structure, **Step 1** can be implemented in $O(|E(G)|) = O(n(k + r))$ time.

Step 2:

1. While $k \geq 0$ and G contains a clean region C_i whose boundary B_i contains a vertex with more than r clean neighbors in C_i:
   $(G, k) \leftarrow (G \setminus C_i, k - |C_i|)$.
2. If $k \geq 0$ then return (G, k), otherwise, return $(K_r, 1)$.

To justify **Step 2**, assume that a solution S of size exactly k exists, that is, $G \setminus S$ is r-regular. Notice that all vertices in $N_G(S)$ are dirty. Therefore, a clean region will be a subset of either S or $V(G) \setminus (S \cup N_G(S))$.

We show that if there is a vertex $v \in B_i$ with $|N(v) \cap C_i| > r$, then C_i is a subset of S. To this end, suppose that there exists a vertex $v \in B_i$ with $|N(v) \cap C_i| > r$ and $C_i \cap S = \emptyset$. Thus, v must be in S, which yields a contradiction as then at least one vertex of C_i does not have degree r in $G \setminus S$. Therefore $C_i \subseteq S$ and thus **Step 2** produces an equivalent instance for k-Almost r-Regular Graph, satisfying properties (1) and (2). Using the same data structure as in the previous step, this step requires $O(r \cdot n)$.

We now comment on the implementation of **Step 2**. First, we find all clean regions by a modified breadth-first search in $O(r \cdot n)$ time. The clean regions are stored as linked lists. For each clean region, we go through all vertices and their adjacency lists and, by maintaining a counter for each vertex, we count for every dirty vertex in the boundary how many neighbors there are in this clean region. This takes $O(r \cdot n)$ time for all clean regions. If in this process we find a dirty vertex whose counter exceeds r, then we remove the corresponding clean region. This takes $O(r \cdot k)$ time in total (since we can remove at most k vertices, each having degree r). Therefore, the total running time of **Step 2** is dominated by $O(r \cdot n)$.

Our observation that either no vertex of a clean region or the entire clean region is contained in the solution implies that isolated clean regions that contain more than k vertices cannot be part of the solution. This leads to the next reduction step. Recall that we are solving the exact version of k-Almost r-Regular Graph, that is, we are demanding for a solution of size exactly k. For this reason, we cannot just delete every isolated clean region in the graph.

Step 3:

1. While G contains an isolated clean region C_i where $|C_i| \geq k + 1$: $(G, k) \leftarrow (G \setminus C_i, k)$.
2. For $i = r + 1, \ldots, k$ do:
   If G contains s isolated clean regions of i vertices, then modify G by removing $\max\{0, s - \lfloor k/i \rfloor\}$ of them.

Concerning the second part of this step, observe that if there are more than $\lfloor k/i \rfloor$ isolated clean regions of equal size i, then we can remove all but $\lfloor k/i \rfloor$ of them. Considering all possible sizes (at most k), we can conclude that there are at most

$$\sum_{i=1}^{k} \lfloor k/i \rfloor \cdot i \leq \sum_{i=1}^{k} k = k^2$$

vertices in isolated clean regions. Therefore, **Step 3** produces an equivalent instance for k-Almost r-Regular Graph, satisfying properties (1)–(3).

As in **Step 2**, we can find all isolated clean regions by a modified breadth-first search in $O(r \cdot n)$ time. Build an array of length $k - r$ with entries from $r + 1$ to k, where entry i points to a (linked) list of all clean regions containing i

vertices. With the help of this array, it is easy to find (in $O(k)$ time) and remove (in $O(r \cdot n)$ time) all clean regions that have to be removed due to the reduction rule of **Step 3**. Thus $O(r \cdot n)$ dominates the running time required for this step.

The idea for the next reduction step is to replace big non-isolated clean regions that contain more than k vertices by smaller ones, which have a size bounded by a function of k, but contain still more than k vertices in order to get an equivalent problem instance. In this process, the degree of the vertices in the boundary of the corresponding clean regions must not change. For $r = 1$ this step does not apply, as then each non-isolated clean region contains exactly one vertex. For $r = 2$, the task is essentially just to replace long paths by shorter ones, which can be easily dealt with (we do that later in the description of the forth reduction step). The replacement gets more involved for $r \geq 3$, as we generally must be able to give an appropriate regular gadget with the constraints mentioned above. We describe the technical details of the replacement in what follows, then we give the reduction step and, after that, we show its correctness.

We need to consider the set of edges between a clean region and its boundary. Let E_i be the set of edges connecting vertices in B_i with vertices in C_i. We search for all clean regions C_i of size greater than $(x+1) \cdot (r+1)$ where $x = \max\{|B_i|, \lceil \frac{k+1}{r+1} \rceil\}$ and replace each one by a new clean region of size $(x+1) \cdot (r+1)$ without affecting the neighborhood of any vertex in the corresponding boundary B_i (notice that $(x+1) \cdot (r+1) \geq x \cdot (r+1) \geq k+1$, which is important as this prevents such a new clean region from being part of the solution). We first describe some structure needed for the replacement, then we state the reduction step.

We replace a clean region C_i by an r-regular structure $R_{r,x}$ of size $(x+1) \cdot (r+1)$, and then reconnect the vertices in B_i with vertices in $R_{r,x}$ such that $R_{r,x}$ remains clean (for this, we apply some modifications in $R_{r,x}$) and such that the degree of the vertices in B_i is as before the replacement. The regular structure $R_{r,x}$ is constructed as follows. Take a cycle of $2 \cdot (x+1)$ edges, remove every second edge $\{u, v\}$ and replace it by a graph $G_{u,v}$ consisting of a $(r-1)$-clique whose vertices are all connected with v and u (notice that $G_{u,v}$ is K_{r+1} with an edge removed). See Figure 4 for an example of such a graph $G_{u,v}$. The resulting graph is r-regular and contains $(x+1) \cdot (r+1)$ vertices.

In the reduction step we reconnect vertices in B_i with vertices in $R_{r,x}$ by removing some edges in $R_{r,x}$ and then connecting their endpoints with vertices in B_i. In this process, we must assure that the new clean region does not decay into several clean regions. Therefore, we define a set of edges M in $R_{r,x}$ such that the graph which results by removing M from $R_{r,x}$ consists of exactly one connected component: Any single $G_{u,v}$ contains a matching $M_{u,v}$ of size $\lceil \frac{r}{2} \rceil$,

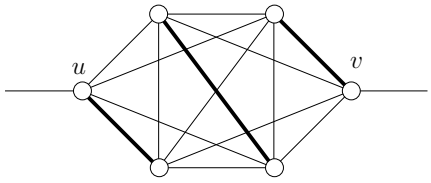

Fig. 4. The graph $G_{u,v}$ for $r = 5$ and a matching (bold edges) in it.

which can be constructed by using a Hamiltonian path from u to v, removing every second edge on it. See Figure 4 for an example. Let M be the union of all $M_{u,v}$. Observe that M is a matching of size $(x+1)(\lceil \frac{r}{2} \rceil)$ for $R_{r,x}$, since there are $x+1$ copies of $G_{u,v}$ in $R_{r,x}$. Furthermore, observe that $R_{r,x}$ remains connected if we remove from it all edges of M, since in each $G_{u,v}$ there is always a u-v-path that contains no edge from $M_{u,v}$. (this property is independent of how the set $M_{u,v}$ was chosen for each $G_{u,v}$). The next reduction step applies these observations.

Step 4:
For $r = 2$, apply the following replacement procedure for each non-isolated clean region C_i of G with $|C_i| > k+1$. The clean region C_i forms a path in G. Let a and b be its endpoints in $G[C_i]$. Remove all vertices in $C_i \setminus \{a, b\}$ from G and reconnect a and b by a path with $k-1$ new vertices.
For $r \geq 3$, apply the following replacement procedure for each non-isolated clean region C_i of G with $|C_i| > (x+1) \cdot (r+1)$:

1. Subdivide in G all edges in E_i. Let L be the set of subdivision vertices.
2. Remove all vertices in C_i from G.
3. Add $R_{r,x}$ to G, that is, set $G := (V(G) \cup V(R_{r,x}), E(G) \cup E(R_{r,x}))$.

4.1 If $|L|$ is even, then choose, arbitrarily, a subset $M' \subseteq M$ where $|M'| = |L|/2$, and remove the edges of M' from $R_{r,x}$. Identify, arbitrarily, their endpoints with the vertices in L.

4.2 If $|L|$ is odd, then choose, arbitrarily, a subset $M' \subseteq M$ where $|M'| = (|L| + r - 2)/2$, and remove the edges of M' from $R_{r,x}$ (as we will see, r is also odd in this case). Identify, arbitrarily, $|L| - 1$ of their endpoints with all vertices in L except one (say w), and then make w adjacent with the remaining $2|M'| - (|L| - 1) = |L| + r - 2 - (|L| - 1) = r - 1$ endpoints of the edges in M'.

For $r \geq 2$ the replacement is clearly correct, as clean regions with more than $k+1$ vertices are replaced by clean regions with exactly $k+1$ vertices. To see that this procedure works correctly for $r \geq 3$, consider the following remarks.
(1.) From property (2) we have $|E_i| \leq r \cdot |B_i|$ and hence $|L| \leq r \cdot |B_i|$.
(4.) The choice of a set M' of claimed size is always possible since the set M

is large enough. More formally, we verify that

$$\begin{aligned} |M| &= (x+1) \cdot \lceil r/2 \rceil \\ &\geq (x+1) \cdot r/2 \\ &\geq (|B_i|+1) \cdot r/2 \\ &\geq (|L|+r)/2 > |M'|. \end{aligned}$$

(4.1.) After removing the edges, its endpoints have degree $r-1$. However, after identifying each such endpoint with one vertex in L, all vertices in $R_{r,x}$ have degree r again, since M is a matching. Due to the above mentioned properties of M, $R_{r,x}$ is still one clean region.
(4.2.) First of all, note that $(|L|+r-2)/2$ is a positive integer, since $|L| \geq 1$ and since $|L| = |E_i|$ being odd implies r to be odd, which can be seen easily by stating the number of edges in $G[C_i]$ and between C_i and B_i as $2|E'| + |E_i| = r|C_i|$, where E' is the set of edges in $G[C_i]$. With the method in (4.1.) we can only identify an even number of vertices in L with vertices in $R_{r,x}$. Therefore, there remains one vertex in L which has to be made adjacent to the remaining endpoints of edges in M'. It is easy to see that afterwards all vertices in $R_{r,x}$ have degree r and that $R_{r,x}$ is still one clean region.

Thus, **Step 4** replaces a clean region C_i of size more than $(x+1)\cdot(r+1)$ by one of size $(x+1)\cdot(r+1) \geq k+1$. Moreover, the new clean region C_i' is connected, has the same boundary B_i as C_i, and all vertices in B_i have the same number of neighbors in C_i' as they had in C_i. We will now prove that **Step 4** produces an equivalent instance for k-Almost r-Regular Graph. For this, let S be a size-k vertex set such that $G \setminus S$ is r-regular. A solution S for G cannot contain a vertex of any C_i that changed, as C_i contains more than k vertices. We retained the vertex degree of all vertices in B_i, C_i' is also a clean region in G', and we did not alter the subgraph $G \setminus C_i = G' \setminus C_i'$, thus $G' \setminus S$ must also be r-regular. Therefore, S is also a solution for G'. The same argument holds for the other direction: A solution S' for G' cannot contain any vertex of any C_i', as C_i' contains more than k vertices, and since $G \setminus C_i = G' \setminus C_i'$ we know that S' is also a solution for G. Finally, it is easy to verify that the new instance satisfies properties (1) – (4).

The implementation of this step again follows the ideas of the previous ones. Again, clean regions can be detected by a modified breadth-first search in $O(r \cdot n)$ time. Notice that the number of vertices to be subdivided in the whole graph is at most $r \cdot n$. Subdividing the edges takes $O(r \cdot n)$ time, and removing clean regions takes $O(r \cdot n)$ time in total. The new clean regions can have at most n vertices and $O(r \cdot n)$ edges in total, thus the total running time for this step is again $O(r \cdot n)$.

The last reduction step is the following:

Step 5:
If G contains more than $rk(k+r)(k+3r+4)+k+k(k+r)+k^2$ vertices then return $(K_r, 1)$, otherwise return (G, k).

Assume that a solution S of size exactly k exists, i.e., $G \setminus S$ is r-regular. We define $D = N_G(S)$ and $F = V(G) \setminus (S \cup D)$ and observe that S, D, F is a 3-partition of $V(G)$. From property (1) every vertex in G has degree at most $r+k$. Therefore, the number of vertices in the neighborhood of S cannot exceed $k(r+k)$ and thus $|D| \leq k(r+k)$. We also observe that all vertices in F are clean, otherwise $G \setminus S$ would contain a vertex not having degree r, which contradicts S being a solution. It remains to bound the size of F. Recall that a clean region C_i is either completely contained in S or no vertex in C_i is member of S, thus $C_i \subseteq F$. Therefore, as all vertices in F are clean, F is a union of clean regions. Suppose that F consists of a set $\mathcal{C} = \{C_i \mid 0 \leq i \leq q\}$ of q non-isolated clean regions. As there is no edge in G between S and F (i.e., D separates S and F) and all vertices in F are clean, we obtain that all boundary vertices of the clean regions in $\mathcal{C}$ must be in D, i.e., $\bigcup_{i=1,\ldots,q} B_i \subseteq D$. Also, since $G \setminus S$ is r-regular, each vertex of D belongs to at most r sets in $\mathcal{B} = \{B_1, \ldots, B_q\}$, and therefore $\sum_{i=1,\ldots,q} |B_i| \leq r \cdot |D| \leq rk(k+r)$. From property (4),

$$\begin{aligned} |C_i| &\leq (\max\{\lceil \frac{k+1}{r+1} \rceil, |B_i|\} + 1) \cdot (r+1) \\ &\leq \max\{k+r+2, |B_i| \cdot (r+1)\} + r + 1. \end{aligned}$$

Recall that F contains at most $\sum_{1,\ldots,q} |C_i|$ vertices from non-isolated clean regions. From property (3), no more than k^2 vertices are contained in isolated regions. Therefore,

$$\begin{aligned} |F| &\leq k^2 + \sum_{1,\ldots,q} (\max\{|B_i| \cdot (r+1), k+r+2\} + r + 1) \\ &\leq k^2 + \sum_{1,\ldots,q} (|B_i| \cdot (r+1) + k + 2r + 3) \\ &\leq k^2 + \sum_{1,\ldots,q} (|B_i| \cdot (r+1)) + \sum_{1,\ldots,q} (k + 2r + 3) \\ &= k^2 + rk(k+r)(r+1) + rk(k+r)(k+2r+3) \\ &= k^2 + rk(k+r)(k+3r+4). \end{aligned}$$

Since $|S| = k$ and $|D| \leq k(k+r)$ we can conclude that **Step 5** returns an equivalent instance of size $O(kr(k+r)^2)$ and the claimed kernel size is correct. To complete the proof, recall that for all reduction steps described, the running time was no worse than the running time of constructing an enhanced data structure, which is $O(|E(G)|) = O(n(k+r))$. □

Notice that Theorem 3 holds also for the non-exact version (demanding a solution of size at most k) of k-Almost r-Regular Graph. The only mod-

ification is that we have to replace the second part of **Step 3** by a deletion of all isolated clean regions.

For $r = 1$, every non-isolated clean region contains a single vertex and **Step 3** and **Step 4** do not apply at all. This permits us to make a better counting of the vertices in F that, apart from those belonging to isolated clean regions, are at most as many as the vertices in D. As any isolated clean region contains exactly 2 vertices when $r = 1$, the second part of **Step 3** should be applied only for $i = 2$, leaving at most $k+1$ vertices in isolated clean regions. Therefore, in the case $r = 1$, the kernel has size at most $|S|+2|D| \leq k+2k(k+1)+k+1 = O(k^2)$.

5 A Search Tree Based Algorithm for k-Almost r-Regular Graph

In this section we present a simple exact algorithm for k-Almost r-Regular Graph running in $O(n \cdot (r+2)^k)$ time, where n is the number of vertices in the input graph. It is based on a bounded search tree technique. Let (G, k, r) be an instance of k-Almost r-Regular Graph. While the graph G is not r-regular, we choose an arbitrary vertex $v \in V(G)$ with $\deg(v) > r$ and branch into the following two cases, where in each case we also always exhaustively apply the simple rule that each vertex with degree less than r is removed from G, setting $k \leftarrow k-1$.

(1) v is a part of the solution, then remove v from G and set $k \leftarrow k-1$.
(2) v is not a part of the solution, thus it remains in G. Then choose an arbitrary subset of $r+1$ neighbors of v. At least one of these neighbors must be contained in the solution in order to achieve degree r for v; thus we branch into the $r+1$ subcases, and, in each of the subcases, we choose a neighbor, remove it from G, and set $k \leftarrow k-1$.

In each subcase of the branching we put at least one vertex from G in the solution, and we branch into $r+2$ subcases. This results in a search tree of size $O((r+2)^k)$. The test of r-regularity can be done in $O(nr)$ time, where $n := |V(G)|$. This means that a solution for k-Almost r-Regular Graph can be found in $O(nr(r+2)^k)$ time, if it exists. Combining this with Theorem 3 we arrive at the following result.

Theorem 4 *For any $r \geq 0$, there exists an algorithm for k-Almost r-Regular Graph with parameter k that runs in $O(n(k+r) + kr^2(k+r)^2 \cdot (r+2)^k)$ steps.*

6 Conclusion

In this paper, we showed that the parameterized problem asking whether we can make a graph r-regular by removing k vertices, with k as parameter, is fixed-parameter tractable by giving a (polynomial size) problem kernel and a search tree algorithm. In the construction of the kernel we used the fact that big "clean regions" can be safely replaced by smaller ones (but not too small). Because r-regularity is not a hereditary property, we had to take care that such a replacement locally maintains r-regularity. Similar ideas were employed in [7] for another non-hereditary property. It is an interesting problem to characterize the properties for which the vertex removal problem is fixed-parameter tractable. That way, one might extend the general result in [2] for non-hereditary properties as well.

Acknowledgements

We wish to thank an anonymous referee for detecting, in a previous version of this paper, a flaw in **Step 4** of our kernelization. Moreover, we thank two anonymous referees for helpful comments, especially for pointing out a simplified version of the gadget construction in the kernelization algorithm. We thank Jiong Guo and Sebastian Wernicke (University Jena, Germany) for giving helpful advice and inspiring ideas. We also thank Josep Díaz (UPC, Barcelona, Spain) for encouraging us to work on this problem.

References

[1] N. Biggs. *Algebraic Graph Theory, Second Edition*. Cambridge University Press, 1994.

[2] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[3] D. M. Cardoso, M. Kamiński, and V. Lozin. Maximum k-regular induced subgraphs. *Journal of Combinatorial Optimization*, 14(4):455–463, 2007.

[4] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of STOC'08*, pages 177–186, ACM Press, 2008.

[5] S. A. Cook and P. McKenzie. Problems complete for deterministic logspace. *Journal of Algorithms*, 8(3):385–394, 1987.

[6] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT-algorithm for the undirected feedback vertex set problem. *Theory of Computing Systems*, 41(3):479–492, 2007.

[7] J. Díaz and D. M. Thilikos. Fast FPT-algorithms for cleaning grids. In *Proceedings of STACS'06*, volume 3884 of *LNCS*, pages 361–371. Springer, 2006.

[8] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[9] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science*, 141(1-2): 109–131, 1995.

[10] S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Planar graph bipartization in linear time. In *Proceedings of GRACO'05*, volume 19 of *Electronic Notes in Discrete Mathematics*, pages 265–271 (electronic). Elsevier, 2005.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[12] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[13] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for Feedback Vertex Set and Edge Bipartization. *Journal of Computer and System Sciences*, 72(8): 1386–1396, 2006.

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

[15] S. Khot and V. Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997-1008, 2002.

[16] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[17] D. Marx. Chordal deletion is fixed-parameter tractable. In *Proceedings of WG'06*, volume 4271 of *LNCS*, pages 37–48. Springer, 2006.

[18] L. Mathieson and S. Szeider. The parameterized complexity of regular subgraphs problems and generalizations, In *Proceedings of CATS'08*, volume 77 of Conferences in Research and Practice in Information Technology, pages 79–86, 2008.

[19] L. Mathieson and S. Szeider. Parameterized graph editing with chosen vertex degrees. In *Proceedings of COCOA'08*, *LNCS*, Springer, 2008. To appear.

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

[21] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[22] I. A. Stewart. Deciding whether a planar graph has a cubic subgraph is NP-complete. *Discrete Mathematics*, 126(1-3):349–357, 1994.

[23] I. A. Stewart. Finding regular subgraphs in both arbitrary and planar graphs. *Discrete Applied Mathematics*, 68(3):223–235, 1996.

[24] I. A. Stewart. On locating cubic subgraphs in bounded-degree connected bipartite graphs. *Discrete Mathematics*, 163(1-3):319–324, 1997.

[25] M. Yannakakis. Node- and edge-deletion NP-complete problems. In *Proceedings of STOC'78*, pages 253–264. ACM, New York, 1978.