The Computational Complexity of Disconnected Cut and $2K_2$-Partition [1]

Barnaby Martin and Daniël Paulusma [2,3]

School of Engineering and Computing Sciences, Durham University, Science Labs, South Road, Durham DH1 3LE, U.K.

Abstract

For a connected graph $G = (V, E)$, a subset $U \subseteq V$ is called a disconnected cut if U disconnects the graph and the subgraph induced by U is disconnected as well. We show that the problem to test whether a graph has a disconnected cut is NP-complete. This problem is polynomially equivalent to the following problems: testing if a graph has a $2K_2$-partition, testing if a graph allows a vertex-surjective homomorphism to the reflexive 4-cycle and testing if a graph has a spanning subgraph that consists of at most two bicliques. Hence, as an immediate consequence, these three decision problems are NP-complete as well. This settles an open problem frequently posed in each of the four settings.

Key words: Graph Theory, Disconnected Cut, 2K2-Partition, Biclique Cover

1 Introduction

We solve an open problem that showed up as a missing case (often *the* missing case) in a number of different research areas arising from connectivity theory, graph covers, graph homomorphisms and graph modification. It is the only open question in the papers by Dantas et al. [8] and Fleischner et al. [15], a principal open question of Ito et al. [24,25], and the central question discussed by Cook et al. [6] and Dantas et al. [9]. Indeed, the problem is considered important enough to generate its own complexity class [13,29], and it is known to be tractable for many graph classes [6,9,15,24].

[1] An extended abstract of this paper appeared in the proceedings of CP2011 [28].
[2] The first author was supported by EPSRC grant EP/G020604/1.
[3] The second author was supported by EPSRC grant EP/G043434/1.

Before we explain how these areas are related, we briefly describe them first. Throughout the paper, we consider undirected finite graphs that have no multiple edges. Unless explicitly stated otherwise they do not have self-loops either. We denote the vertex set and edge set of a graph G by V_G and E_G, respectively. If no confusion is possible, we may omit the subscripts. We let $n = |V(G)|$ denote the number of vertices of G. The *complement* of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \{uv \notin E \mid u \neq v\})$. For a subset $U \subseteq V_G$, we let $G[U]$ denote the subgraph of G *induced by* U, which is the graph $(U, \{uv \mid u, v \in U \text{ and } uv \in E_G\})$.

1.1 Vertex Cut Sets

A maximal connected subgraph of G is called a *component* of G. A *vertex cut (set)* or *separator* of a graph $G = (V, E)$ is a subset $U \subset V$ such that $G[V \backslash U]$ contains at least two components.

Vertex cuts play an important role in graph connectivity, and various kinds of vertex cuts have been studied in the literature. For instance, a cut U of a graph $G = (V, E)$ is called a *k-clique cut* if $G[U]$ has a spanning subgraph consisting of k complete graphs; a *strict k-clique cut* if $G[U]$ consists of k components that are complete graphs; a *stable cut* if U is an independent set; and a *matching cut* if $E_{G[U]}$ is a matching. The problem that asks whether a graph has a k-clique cut is solvable in polynomial time for $k = 1$ and $k = 2$, as shown by Whitesides [35] and Cameron et al. [5], respectively. The latter authors also showed that deciding if a graph has a strict 2-clique cut can be solved in polynomial time. On the other hand, the problems that ask whether a graph has a stable cut or a matching cut, respectively, are NP-complete, as shown by Chvátal [7] and Brandstädt et al. [2], respectively.

For a fixed constant $k \geq 1$, a cut U of a connected graph G is called a *k-cut* of G if $G[U]$ contains exactly k components. Testing if a graph has a k-cut is solvable in polynomial time for $k = 1$, whereas it is NP-complete for every fixed $k \geq 2$ [24]. For $k \geq 1$ and $\ell \geq 2$, a k-cut U is called a *(k, ℓ)-cut* of a graph G if $G[V \backslash U]$ consists of exactly ℓ components. Testing if a graph has a (k, ℓ)-cut is polynomial-time solvable when $k = 1$, $\ell \geq 2$, and NP-complete otherwise [24].

A cut U of a graph G is called *disconnected* if $G[U]$ contains at least two components. We observe that U is a disconnected cut if and only if $V \backslash U$ is a disconnected cut if and only if U is a (k, ℓ)-cut for some $k \geq 2$ and $\ell \geq 2$. The following question was posed in several papers [15,24,25] as an open problem.

Q1. How hard is it to test if a graph has a disconnected cut?

The problem of testing if a graph has a disconnected cut is called the Disconnected Cut problem. It is known that every graph of diameter 1 has no disconnected cut, and every graph of diameter at least 3 has a disconnected cut [15]. Hence, in order to determine the computational complexity of Disconnected Cut, we may restrict ourselves to graphs of diameter 2.

A disconnected cut U of a connected graph $G = (V, E)$ is *minimal* if $G[(V \setminus U) \cup \{u\}]$ is connected for every $u \in U$. Recently, the corresponding decision problem called Minimal Disconnected Cut was shown to be NP-complete [25].

1.2 H-partitions

A *model graph* H with $V_H = \{h_0, \ldots, h_{k-1}\}$ has two types of edges: solid and dotted edges, and an *H-partition* of a graph G is a partition of V_G into k (nonempty) sets $V_0, \ldots, V_{k-1}$ such that for all vertices $u \in V_i$, $v \in V_j$ and for all $0 \leq i < j \leq k-1$ the following two conditions hold. Firstly, if $h_i h_j$ is a solid edge of H, then $uv \in E_G$. Secondly, if $h_i h_j$ is a dotted edge of H, then $uv \notin E_G$. There are no such restrictions when h_i and h_j are not adjacent. Let $2K_2$ be the model graph with vertices $h_0, \ldots, h_3$, solid edges h_0h_2, h_1h_3 and no dotted edges, and $2S_2$ be the model graph with vertices $h_0, \ldots, h_3$, dotted edges h_0h_2, h_1h_3 and no solid edges. We observe that a graph G has a $2K_2$-partition if and only if its complement $\overline{G}$ has a $2S_2$-partition.

The following question was mentioned in several papers [6,8,9,13,29] as an open problem.

Q2. How hard is it to test if a graph has a $2K_2$-partition?

One of the reasons for posing this question is that the (equivalent) cases $H = 2K_2$ and $H = 2S_2$ are the only two cases of model graphs on at most four vertices for which the computational complexity of the corresponding decision problem, called H-Partition, is still open. In fact it is known that H-Partition is polynomial-time solvable for all other 4-vertex model graphs H. In particular, the model graph H with vertices $h_0, \ldots, h_3$, solid edge h_0h_2 and dotted edge h_1h_3 is well known. In that case the H-Partition problem is called the Skew Partition problem. Note that this problem is equivalent to asking whether the vertex set of a given graph can be partitioned into two sets V_1 and V_2 such that V_1 induces a disconnected graph in G and V_2 induces a disconnected graph in $\overline{G}$. Even the list version of this problem, where each vertex has been assigned a list of blocks in which it must be placed, is polynomial-time solvable, as shown by de Figueiredo, Klein and Reed [14] (later, Kennedy and Reed [27] presented a faster polynomial-time algorithm for the non-list version). All other cases of H-Partition for 4-vertex model

graphs $H \notin \{K_2, S_2\}$ have been settled by Dantas et al. [8].

In the literature, $2K_2$-partitions have been well studied, see e.g. three recent papers of Cook et al. [6], Dantas, Maffray and Silva [9] and Teixeira, Dantas and de Figueiredo [29]. The first two papers [6,9] study the $2K_2$-Partition problem for several graph classes, and the third paper [29] defines a new class of problems called $2K_2$-hard. In addition, the first paper also proves that $2K_2$-Partition can be solved in $O((2^d-1)n^2)$ time for n-vertex graphs of minimum vertex degree d. By a result on retractions of Hell and Feder [11], which we explain later, the list versions of $2S_2$-Partition and $2K_2$-Partition are NP-complete. A variant on H-partitions that allows empty blocks V_i in an H-partition is studied by Feder et al. [12], whereas Cameron et al. [5] consider the list version of this variant.

1.3 Graph Covers

Let G be a graph and $\mathcal{S}$ be a set of (not necessarily vertex-induced) subgraphs of G that has size $|\mathcal{S}|$. The set $\mathcal{S}$ is a *cover* of G if every edge of G is contained in at least one of the subgraphs in $\mathcal{S}$. The set $\mathcal{S}$ is a *vertex-cover* of G if every vertex of G is contained in at least one of the subgraphs in $\mathcal{S}$. If all subgraphs in $\mathcal{S}$ are *bicliques*, that is, complete connected bipartite graphs, then we speak of a *biclique cover* or a *biclique vertex-cover*, respectively. Testing whether a graph has a biclique cover of size at most k is polynomial-time solvable for any fixed k; it is even fixed-parameter tractable in k as shown by Fleischner et al. [15]. The same authors [15] show that testing whether a graph has a biclique vertex-cover of size at most k is polynomial-time solvable for $k = 1$ and NP-complete for $k \geq 3$. For $k = 2$, they show that this problem can be solved in polynomial time for bipartite input graphs, and they pose the following open problem.

Q3. How hard is it to test if a graph has a biclique vertex-cover of size 2*?*

The problem of testing if a graph has a biclique vertex-cover of size 2 is called the 2-Biclique Vertex-Cover problem. In order to answer question Q3 we may without loss of generality restrict to biclique vertex-covers in which every vertex is in exactly one of the subgraphs in $\mathcal{S}$ (cf. [15]).

1.4 Graph Homomorphisms

A *homomorphism* from a graph G to a graph H is a mapping $f : V_G \to V_H$ that maps adjacent vertices of G to adjacent vertices of H, i.e., $f(u)f(v) \in$

E_H whenever $uv \in E_G$. The problem H-Homomorphism tests whether a given graph G (also called the *guest graph*) allows a homomorphism to a graph H called the *target* which is fixed, i.e., not part of the input. This problem is also known as H-Coloring. Hell and Nešetřil [22] showed that H-Homomorphism is solvable in polynomial time if H is bipartite, and NP-complete otherwise. Here, H is assumed not to have a self-loop xx, as otherwise we can map every vertex of G to x.

A homomorphism f from a graph G to a graph H is *surjective* if for each $x \in V_H$ there exists at least one vertex $u \in V_G$ with $f(u) = x$. This leads to the problem of deciding if a given graph allows a surjective homomorphism to a fixed target graph H, which is called the Surjective H-Homomorphism or Surjective H-Coloring problem. For this variant, the presence of a vertex with a self-loop in the target graph H does not make the problem trivial. Such vertices are called *reflexive*, whereas vertices with no self-loop are said to be *irreflexive*. A graph that contains one or more reflexive vertices is called *partially reflexive*. In particular, a graph is *reflexive* if all its vertices are reflexive, and a graph is *irreflexive* if all its vertices are irreflexive. Golovach, Paulusma and Song [18] showed that for any fixed tree H, the Surjective H-Homomorphism problem is polynomial-time solvable if the (possibly empty) set of reflexive vertices in H induces a connected subgraph of H, and NP-complete otherwise. They mention that the smallest open case is the case, in which H is the reflexive 4-cycle denoted $\mathcal{C}_4$.

Q4. How hard is it to test if a graph has a surjective homomorphism to $\mathcal{C}_4$?

The following two notions are closely related to surjective homomorphisms. A homomorphism f from a graph G to an induced subgraph H of G is a *retraction* from G to H if $f(h) = h$ for all $h \in V_H$. In that case we say that G *retracts to* H. Note that this implies that G allows a vertex-surjective homomorphism to H, whereas the reverse implication does not necessarily hold. For a fixed graph H, the H-Retraction problem has as input a graph G that contains H as an induced subgraph and is to test if G retracts to H. Hell and Feder [11] showed that $\mathcal{C}_4$-Retraction is NP-complete.

We emphasize that a surjective homomorphism is vertex-surjective. A stronger notion is to require a homomorphism from a graph G to a graph H to be *edge-surjective*, which means that for any edge $xy \in E_H$ with $x \neq y$ there exists an edge $uv \in E_G$ with $f(u) = x$ and $f(v) = y$. Note that the edge-surjectivity condition only holds for edges $xy \in E_H$; there is no such condition on the self-loops $xx \in E_H$. An edge-surjective homomorphism is also called a *compaction*. If f is a compaction from G to H, we say that G *compacts* to H. The H-Compaction problem asks if a graph G compacts to a fixed graph H. Vikas [30–32] determined the computational complexity of this prob-

lem for several classes of fixed target graphs. In particular, he showed that $\mathcal{C}_4$-Compaction is NP-complete [30]. More recently, Vikas [33,34] considered H-Compaction for guest graphs belonging to some restricted graph class.

1.5 Graph Contractibility

A graph modification problem has as input a graph G and an integer k. The question is whether G can be modified to belong to some specified graph class that satisfies further properties by using at most k operations of a certain specified type such as deleting a vertex or deleting an edge. Another natural operation is the *contraction* of an edge, which removes both end-vertices of the edge and replaces them by a new vertex adjacent to precisely those vertices that were adjacent to at least one of the two end-vertices. If a graph H can be obtained from G by a sequence of edge contractions, then G is said to be *contractible to* H. The problem Π-Contractibility has as input a graph G together with an integer k and is to test whether G is contractible to a graph in Π by using at most k edge contractions.

Asano and Hirata [1] show that Π-Contractibility is NP-complete if Π satisfies certain conditions. As a consequence, this problem is NP-complete for many graph classes Π such as the classes of planar graphs, outerplanar graphs, series-parallel graphs, forests and chordal graphs. By a result of Heggernes et al. [20], Π-Contractibility is NP-complete for trees even if the input graph is bipartite. If Π is the class of paths or cycles, then Π-Contractibility is polynomially equivalent to the problems of determining the length of a longest path and a longest cycle, respectively, to which a given graph can be contracted. The first problem has been shown to be NP-complete by van 't Hof, Paulusma and Woeginger [23] even for graphs with no induced path on 6 vertices. The second problem has been shown to be NP-complete by Hammack [19]. Heggernes et al. [21] observed that Π-Contractibility is NP-complete when Π is the class of bipartite graphs, whereas Golovach et al. [17] showed that Π-Contractibility is NP-complete when Π is the class of graphs of a certain minimum degree d; they show that $d = 14$ suffices.

A graph G contains a graph H as a *minor* if G can be modified to H by a sequence of vertex deletions, edge deletions, and edge contractions. Eppstein [10] showed that it is NP-complete to decide if a given graph G has a complete graph K_h as a minor for some given integer h. This problem is equivalent to deciding if a graph G is contractible to K_h. Hence, Π-Contractibility is NP-complete if Π is the class of complete graphs.

The biclique with partition classes of size k and ℓ, respectively, is denoted $K_{k,\ell}$. A *star* is a biclique $K_{1,\ell}$ for some integer $\ell \geq 2$. If Π is the class of stars,

then Π-Contractibility is NP-complete. This can be seen as follows. Let $K_1 \bowtie G$ denote the graph obtained from a graph G after adding a new vertex and making it adjacent to all vertices of G. Then G has an independent set of size h for some given integer $h \geq 2$ if and only if $K_1 \bowtie G$ is contractible to a star $K_{1,h}$. Since the first problem is NP-complete [16,26], the result follows.

A remaining elementary graph class is the class of bicliques $K_{k,\ell}$ with $k \geq 2$ and $\ell \geq 2$; we call such bicliques *proper*. In order to determine the complexity for this graph class, we first consider the following question.

Q5. How hard is it to test if a graph is contractible to a proper biclique?

The problem of testing whether a graph can be contracted to a proper biclique is called the Biclique Contractibility problem. By setting $k = n$, we see that this problem is a special instance of the corresponding Π-Contractibility problem. If one of the two integers $k \geq 2$ or $\ell \geq 2$ is fixed, then testing if G is contractible to $K_{k,\ell}$ is known to be NP-complete [24].

1.6 The Relationships Between Questions Q1–Q5

Before we explain how questions Q1–Q5 are related, we introduce some new terminology. The *distance* $d_G(u,v)$ between two vertices u and v in a graph G is the number of edges in a shortest path between them; if there is no path between u and v then $d_G(u,v) = \infty$. The *diameter* $\mathrm{diam}(G)$ is defined as $\max\{d_G(u,v) \mid u,v \in V\}$. A biclique is called *nontrivial* if $k \geq 1$ and $\ell \geq 1$.

Proposition 1 ([24]) *Let G be a connected graph. Then statements* (1)–(5) *are equivalent:*

(1) *G has a disconnected cut.*
(2) *G has a $2S_2$-partition.*
(3) *G allows a vertex-surjective homomorphism to $\mathcal{C}_4$.*
(4) *$\overline{G}$ has a spanning subgraph that consists of exactly two nontrivial bicliques.*
(5) *$\overline{G}$ has a $2K_2$-partition.*

If $\mathrm{diam}(G) = 2$*, then* (1)–(5) *are also equivalent to the following statements:*

(6) *G allows a compaction to $\mathcal{C}_4$.*
(7) *G is contractible to some biclique $K_{k,\ell}$ for some $k, \ell \geq 2$.*

Due to Proposition 1, questions Q1–Q4 are equivalent. Recall that we may restrict ourselves to graphs of diameter 2, as every graph of diameter 1 has no disconnected cut, and every graph of diameter at least 3 has a disconnected cut [15]. Under this restriction, Proposition 1 tells us that Q1–Q4 are

also equivalent to Q5 and to the question of determining the computational complexity of $\mathcal{C}_4$-Compaction. Recall that Vikas [30] showed that the latter problem is NP-complete. However, the gadget in his NP-completeness reduction has diameter 3 as observed by Ito et al. [25].

Our Result. A pair of vertices in a graph is a *dominating (non-)edge* if the two vertices of the pair are (non-)adjacent, and any other vertex in the graph is adjacent to at least one of them. We solve question Q4 by showing that the problem Surjective $\mathcal{C}_4$-Homomorphism is indeed NP-complete for graphs of diameter 2 even if they have a dominating non-edge. In contrast, Fleischner et al. [15] showed that this problem is polynomial-time solvable on input graphs with a dominating edge. As a consequence of our result and Proposition 1, we find that the problems Disconnected Cut, $2K_2$-Partition, $2S_2$-Partition, and 2-Biclique Vertex-Cover and also that the problems $\mathcal{C}_4$-Compaction and Biclique Contraction are NP-complete for graphs of diameter 2 even if they have a dominating non-edge. Hence, we have not only solved question Q4 but also questions Q1, Q2, Q3 and Q5.

Our approach to prove NP-completeness is as follows. As mentioned before, we can restrict ourselves to graphs of diameter 2. We therefore try to reduce the diameter in the gadget of the NP-completeness proof of Vikas [30] for $\mathcal{C}_4$-Compaction from 3 to 2. This leads to NP-completeness of Surjective $\mathcal{C}_4$-Homomorphism, because these two problems coincide for graphs of diameter 2 due to Proposition 1. The proof that $\mathcal{C}_4$-Compaction is NP-complete [30] has its roots in the proof that $\mathcal{C}_4$-Retraction is NP-complete [11]. So far, it was only known that $\mathcal{C}_4$-Retraction stays NP-complete for graphs of diameter 3 [25]. We start our proof by showing that $\mathcal{C}_4$-Retraction is NP-complete even for graphs of diameter 2. The key idea is to base the reduction from an NP-complete homomorphism (constraint satisfaction) problem that we obtain only after a fine analysis under the algebraic conditions of Bulatov [3] and Bulatov, Krokhin and Jeavons [4], which we perform in Section 2. This approach is novel in the sense that usually graph theory provides a test-bed for constraint satisfaction problems whereas here we see a case where the flow of techniques is the other way around. We present our NP-completeness proof for $\mathcal{C}_4$-Retraction on graphs of diameter 2 in Section 3. This leads to a special input graph of the $\mathcal{C}_4$-Retraction problem, which enables us to modify the gadget of the proof of Vikas [30] for $\mathcal{C}_4$-Compaction in order to get its diameter down to 2, as desired. We explain this part in Section 4.

We also point out that Vikas [33,34] has announced to have an NP-completeness proof of $\mathcal{C}_4$-Homomorphism as well but so far has not made his proof publicly available.

2 Constraint Satisfaction

The notion of a graph homomorphism can be generalized as follows. A *structure* is a tuple $\mathcal{A} = (A; R_1, \ldots, R_k)$, where A is a set called the *domain* of $\mathcal{A}$ and R_i is an n_i-ary *relation* on A for $i = 1, \ldots, k$, i.e., a set of n_i-tuples of elements from A. Note that a graph $G = (V, E)$ can be seen as a structure $G = (V; \{(u, v), (v, u) \mid uv \in E\})$. Throughout the paper we only consider *finite* structures, i.e., with a finite domain.

Let $\mathcal{A} = (A; R_1, \ldots, R_k)$ and $\mathcal{B} = (B; S_1, \ldots, S_k)$ be two structures, where each R_i and S_i are relations of the same arity n_i. Then a *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $f : A \to B$ such that $(a_1, \ldots, a_{n_i}) \in R_i$ implies $(f(a_1), \ldots, f(a_{n_i})) \in S_i$ for every i and every n_i-tuple $(a_1, \ldots, a_{n_i}) \in A^{n_i}$. The decision problem that is to test if a given structure $\mathcal{A}$ allows a homomorphism to a fixed structure $\mathcal{B}$ is called the $\mathcal{B}$-Homomorphism problem, also known as the $\mathcal{B}$-Constraint Satisfaction problem.

Let $\mathcal{A} = (A; R_1, \ldots, R_k)$ be a structure and ℓ be an integer. The *power structure* $\mathcal{A}^\ell$ has domain A^ℓ and for $1 \leq i \leq k$, has relations

$$R_i^\ell := \{((a_1^1, \ldots, a_\ell^1), \ldots, (a_1^{n_i}, \ldots, a_\ell^{n_i})) \mid (a_1^1, \ldots, a_1^{n_i}), \ldots, (a_\ell^1, \ldots, a_\ell^{n_i}) \in R_i\}.$$

We note that $R_i^1 = R_i$ for $1 \leq i \leq k$. An (ℓ-ary) *polymorphism* of $\mathcal{A}$ is a homomorphism from $\mathcal{A}^\ell$ to $\mathcal{A}$ for some integer ℓ. A 1-ary polymorphism is also called an *endomorphism*. The set of polymorphisms of $\mathcal{A}$ is denoted $\mathrm{Pol}(\mathcal{A})$.

A binary function f on a domain A is a *semilattice* function if $f(h, f(i, j)) = f(f(h, i), j)$, $f(i, j) = f(j, i)$, and $f(i, i) = i$ for all $i, j \in A$. A ternary function f is a *Mal'tsev* function if $f(i, j, j) = f(j, j, i) = i$ for all $i, j \in A$. A ternary function f is a *majority* function if $f(h, h, i) = f(h, i, h) = f(i, h, h) = h$ for all $h, i \in A$. On the Boolean domain $\{0, 1\}$, we may consider propositional functions. The only two semilattice functions on the Boolean domain are the binary function $\wedge$, which maps (h, i) to $(h \wedge i)$, which is 1 if $h = i = 1$ and 0 otherwise, and the binary function $\vee$ which maps (h, i) to $(h \vee i)$, which is 0 if $h = i = 0$ and 1 otherwise. We may consider each of these two functions on any two-element domain (where we view one element as 0 and the other as 1). For a function f on B, and a subset $A \subseteq B$, we let $f_{|A}$ be the restriction of f to A.

A structure is a *core* if all of its endomorphisms are *automorphisms*, i.e., are invertible. We will make use of the following theorem from Bulatov, Krokhin and Jeavons [4] (it appears in this form in Bulatov [3]).

Theorem 2 ([3,4]) *Let $\mathcal{B} = (B; S_1, \ldots, S_k)$ be a core and $A \subseteq B$ be a subset*

of size $|A| = 2$ *that as a unary relation is in* $\mathcal{B}$*. If for each* $f \in \mathsf{Pol}(\mathcal{B})$*,* $f_{|A}$ *is not majority, semilattice or Mal'tsev, then* $\mathcal{B}$-HOMOMORPHISM *is* NP*-complete.*

Let $\mathcal{D}$ be the structure on domain $D = \{0, 1, 3\}$ with four binary relations

$$S_1 := \{(0,3),(1,1),(3,1),(3,3)\}$$

$$S_2 := \{(1,0),(1,1),(3,1),(3,3)\}$$

$$S_3 := \{(1,3),(3,1),(3,3)\}$$

$$S_4 := \{(1,1),(1,3),(3,1)\}.$$

We use $\{0, 1, 3\}$ (instead of, say, $\{0, 1, 2\}$) to tie in exactly with the Vikas [30] labelling of $\mathcal{C}_4$.

Proposition 3 *The* $\mathcal{D}$-HOMOMORPHISM *problem is* NP*-complete.*

PROOF. We use Theorem 2. We first show that $\mathcal{D}$ is a core. Let g be an endomorphism of $\mathcal{D}$. We must show that g is an automorphism. If $g(0) = 3$ then $g(1) = 3$ by preservation of S_2, i.e., as otherwise $(1, 0) \in S_2$ does not imply $(g(1), g(0)) \in S_2$. However, $(1, 1) \in S_4$ but $(g(1), g(1)) = (3, 3) \notin S_4$. Hence $g(0) \neq 3$. If $g(0) = 1$ then $g(3) = 1$ by preservation of S_1. However, $(3, 3) \in S_3$ but $(g(3), g(3)) = (1, 1) \notin S_3$. Hence $g(0) \neq 1$. This means that $g(0) = 0$. Consequently, $g(1) = 1$ by preservation of S_2, and $g(3) = 3$ by preservation of S_1. Hence, g is the identity mapping, which is an automorphism, as desired.

Let $A = \{1, 3\}$, which is in $\mathcal{D}$ in the form of $S_1(p, p)$ (or $S_2(p, p)$). Suppose that $f \in \mathsf{Pol}(\mathcal{D})$. In order to prove Proposition 3, we must show that $f_{|A}$ is neither majority nor semilattice nor Mal'tsev.

Suppose that $f_{|A}$ is semilattice. Then $f_{|A} = \wedge$ or $f_{|A} = \vee$. If $f_{|A} = \wedge$, then either $f(1,1) = 1$, $f(1,3) = 3$, $f(3,1) = 3$, $f(3,3) = 3$, or $f(1,1) = 1$, $f(1,3) = 1$, $f(3,1) = 1$, $f(3,3) = 3$ depending on how the elements $1, 3$ correspond to the two elements of the Boolean domain. The same holds if $f_{|A} = \vee$. Suppose that $f(1,1) = 1$, $f(1,3) = 3$, $f(3,1) = 3$, $f(3,3) = 3$. By preservation of S_4 we find that $f(1,3) = 1$ due to $f(3,1) = 3$. This is not possible, as $f(1,3) = 3$. Suppose that $f(1,1) = 1$, $f(1,3) = 1$, $f(3,1) = 1$, $f(3,3) = 3$. By preservation of S_3 we find that $f(1,3) = 3$ due to $f(3,1) = 1$. This is not possible either.

Suppose that $f_{|A}$ is Mal'tsev. By preservation of S_4, we find that $f(1,1,3) = 1$ due to $f(3,1,1) = 3$. However, because $f(1,1,3) = 3$, this is not possible.

Suppose that $f_{|A}$ is majority. By preservation of S_1, we deduce that $f(0,3,1) \in \{0,3\}$ due to $f(3,3,1) = 3$, and that $f(0,3,1) \in \{1,3\}$ due to $f(3,1,1) =$

1. Thus, $f(0,3,1) = 3$. By preservation of S_2, however, we deduce that $f(0,3,1) \in \{0,1\}$ due to $f(1,3,1) = 1$. This is a contradiction. Hence, we have completed the proof of Proposition 3. $\square$

3 Retractions

In the remainder of this paper, the graph H denotes the reflexive 4-vertex cycle $\mathcal{C}_4$. We let $h_0, \ldots, h_3$ be the vertices and h_0h_1, h_1h_2, h_2h_3, and h_3h_0 be the edges of H. We prove that H-Retraction is NP-complete for graphs of diameter 2 by a reduction from $\mathcal{D}$-Homomorphism.

Let $\mathcal{A} = (A; R_1, \ldots, R_4)$ be an instance of $\mathcal{D}$-Homomorphism, where we may assume that each R_i is a binary relation. From $\mathcal{A}$ we construct a graph G as follows. We let the elements in $\mathcal{A}$ correspond to vertices of G. If $(p,q) \in R_i$ for some $1 \leq i \leq 4$, then we say that vertex p in G is of *type* ℓ and vertex q in G is of *type* r. Note that a vertex can be of type ℓ and r simultaneously, because it can be the first element in a pair in $R_1 \cup \cdots \cup R_4$ and the second element of another such pair. For each $(p,q) \in R_i$ and $1 \leq i \leq 4$ we introduce four new vertices a_p, b_p, c_q, d_q with edges a_pp, a_pb_p, b_pp, c_qq, c_qd_q and d_qq. We say that a vertex a_p, b_p, c_q, d_q is of *type* a, b, c, d, respectively; note that these vertices all have a unique type.

We now let the graph H be an induced subgraph of G (with distinct vertices $h_0, \ldots, h_3$). Then formally G must have self-loops $h_0h_0, \ldots, h_3h_3$, because H has such self-loops. Outside of H in G, it does not matter whether we consider G to have self-loops or not. In any case we do not draw any loops in our figures in order to keep these uncluttered.

In G we join every a-type vertex to h_0 and h_3, every b-type vertex to h_1 and h_2, every c-type vertex to h_2 and h_3, and every d-type vertex to h_0 and h_1. We also add an edge between h_0 and every vertex of A.

We continue the construction of G by describing how we distinguish between two pairs belonging to different relations. If $(p,q) \in R_1$, then we add the edges c_qp and qh_2; see Figure 1. If $(p,q) \in R_2$, then we add the edges h_2p and b_pq; see Figure 2. If $(p,q) \in R_3$, then we add the edges h_2p, h_2q and a_pc_q; see Figure 3. If $(p,q) \in R_4$, then we add the edges h_2p, h_2q and b_pd_q; see Figure 4.

We finish the construction of G by adding an edge between any two vertices of type a, between any two vertices of type b, between any two vertices of type c, and between any two vertices of type d. Note that this leads to four mutually vertex-disjoint cliques in G; here a *clique* means a vertex set of a complete graph. We call G a $\mathcal{D}$*-graph* and prove the following result.

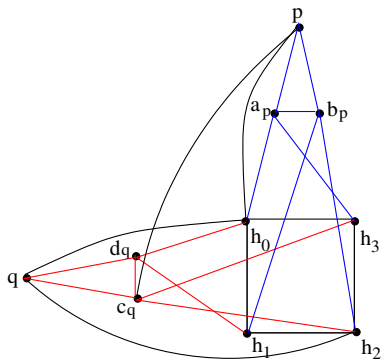

Fig. 1. The part of a $\mathcal{D}$-graph G for a pair $(p, q) \in R_1$.

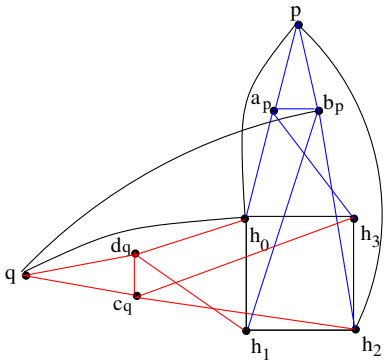

Fig. 2. The part of a $\mathcal{D}$-graph G for a pair $(p, q) \in R_2$.

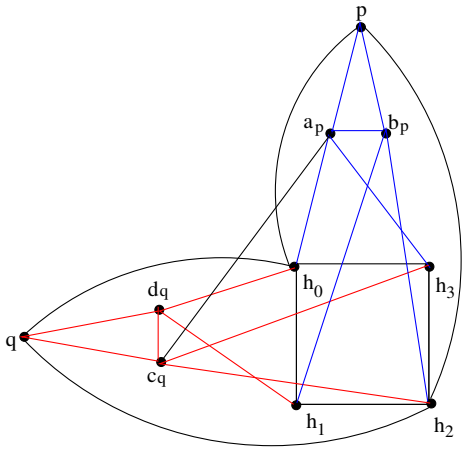

Fig. 3. The part of a $\mathcal{D}$-graph G for a pair $(p, q) \in R_3$.

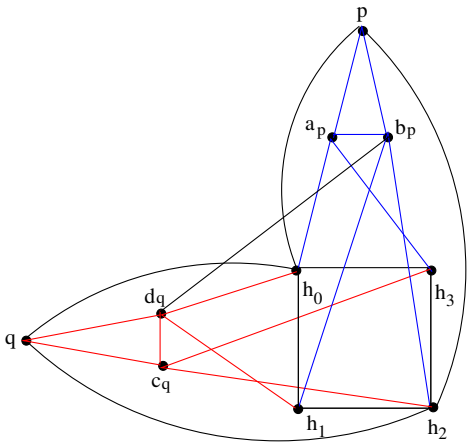

Fig. 4. The part of a $\mathcal{D}$-graph G for a pair $(p, q) \in R_4$.

Lemma 4 *Every $\mathcal{D}$-graph has diameter 2 and a dominating non-edge.*

PROOF. Let G be a $\mathcal{D}$-graph. We first show that G has a dominating non-edge. Note that h_0 is adjacent to all vertices except to h_2 and the vertices of type b and c. However, all vertices of type b and c are adjacent to h_2. Because h_0 and h_2 are not adjacent, this means that h_0 and h_2 form a dominating non-edge in G.

We show that G has diameter 2 in Table 1. In this table, ℓ, r, a, b, c, d denote vertices of corresponding type, and superscripts denote the vertex or its type that connects the two associated vertices in the case they are not adjacent already. The distances in the table must be interpreted as upper bounds. For example, the distance between a vertex p of type ℓ and a vertex $a_{p'}$ of type a is either 1 if $p = p'$ or 2 if $p \neq p'$. In the latter case, they are connected by h_0 (and perhaps by some other vertices as well). The table denotes this as 2^{h_0}. In two cases the connecting vertex depends on the relation R_i. Then

	h_0	h_1	h_2	h_3	ℓ	r	a	b	c	d
h_0	0	1	2^{h_1}	1	1	1	1	2^{h_1}	2^{h_3}	1
h_1	.	0	1	2^{h_0}	2^{h_0}	2^{h_0}	2^{h_0}	1	2^{h_2}	1
h_2	.	.	0	1	2^b	2^c	2^b	1	1	2^c
h_3	.	.	.	0	2^{h_0}	2^{h_0}	1	2^a	1	2^c
ℓ	.	.	.	.	2^{h_0}	2^{h_0}	2^{h_0}	2^b	$2_c^{h_2}$	2^{h_0}
r	.	.	.	.	.	2^{h_0}	2^{h_0}	$2_b^{h_2}$	2^c	2^{h_0}
a	.	.	.	.	.	.	1	2^a	2^{h_3}	2^{h_0}
b	.	.	.	.	.	.	.	1	2^{h_2}	2^{h_1}
c	.	.	.	.	.	.	.	.	1	2^c
d	.	.	.	.	.	.	.	.	.	1

Table 1
Determining the diameter of a $\mathcal{D}$-graph G.

a subscript denotes the necessary second possibility that occurs when the superscript vertex is not valid. For instance, when a vertex p of type ℓ is not adjacent to h_2, then p must be the first element in a pair $(p, q) \in R_1$, and then p is adjacent to c_q, and hence there is always an intermediate vertex (either h_2 or c_q) to connect p to an arbitrary vertex of type c not necessarily equal to c_q. In the table this is expressed as $2_c^{h_2}$. □

Recall that Feder and Hell [11] showed that H-Retraction is NP-complete. Ito et al. [25] observed that H-Retraction stays NP-complete on graphs of diameter 3. For our purposes, we need the following theorem. Note that Lemma 4 and Theorem 5 together imply that H-Retraction is NP-complete for graphs of diameter 2 that have a dominating non-edge.

Theorem 5 *The H-Retraction problem is NP-complete even for $\mathcal{D}$-graphs.*

PROOF. We recall that H-Retraction is in NP, because we can guess a partition of the vertex set of the input graph G into four (non-empty) sets and verify in polynomial time if this partition corresponds to a retraction from G to H.

To show NP-hardness, we reduce from the $\mathcal{D}$-Homomorphism problem. From an instance $\mathcal{A} = (A; R_1, \ldots, R_4)$ of $\mathcal{D}$-Homomorphism we construct a $\mathcal{D}$-graph G. We claim that $\mathcal{A}$ allows a homomorphism to $\mathcal{D}$ if and only if G retracts to H.

p	q	a_p	b_p	c_q	d_q
h_0	h_3	h_0	h_1	h_3	h_0
h_1	h_1	h_0	h_1	h_2	h_1
h_3	h_1	h_3	h_2	h_2	h_1
h_3	h_3	h_3	h_2	h_3	h_0

Table 2
g-values when $(p, q) \in R_1$.

p	q	a_p	b_p	c_q	d_q
h_1	h_0	h_0	h_1	h_3	h_0
h_1	h_1	h_0	h_1	h_2	h_1
h_3	h_1	h_3	h_2	h_2	h_1
h_3	h_3	h_3	h_2	h_3	h_0

Table 3
g-values when $(p, q) \in R_2$.

p	q	a_p	b_p	c_q	d_q
h_1	h_3	h_0	h_1	h_3	h_0
h_3	h_1	h_3	h_2	h_2	h_1
h_3	h_3	h_3	h_2	h_3	h_0

Table 4
g-values when $(p, q) \in R_3$.

p	q	a_p	b_p	c_q	d_q
h_1	h_1	h_0	h_1	h_2	h_1
h_1	h_3	h_0	h_1	h_3	h_0
h_3	h_1	h_3	h_2	h_2	h_1

Table 5
g-values when $(p, q) \in R_4$.

First suppose that $\mathcal{A}$ allows a homomorphism f to $\mathcal{D}$. We construct a mapping g from V_G to V_H as follows. For each $a \in A$ we let $g(a) = h_i$ if $f(a) = i$, and for $i = 0, \ldots, 3$ we let $g(h_i) = h_i$. Because f is a homomorphism from $\mathcal{A}$ to $\mathcal{D}$, this leads to Tables 2–5, which explain where a_p, b_p, c_q and d_q map under g, according to where p and q map. From these, we conclude that g is a retraction from G to H. In particular, we note that the edges $c_q p, b_p q, a_p c_q$, and $b_p d_q$ each map to an edge or self-loop in H when (p, q) belongs to $R_1, \ldots, R_4$, respectively.

To prove the reverse implication, suppose that G allows a retraction g to H. We construct a mapping $f : A \to \{0, 1, 2, 3\}$ by defining, for each $a \in A$, $f(a) = i$ if $g(a) = h_i$. We claim that f is a homomorphism from $\mathcal{A}$ to $\mathcal{D}$. In order to see this, we first note that g maps all a-type vertices to $\{h_0, h_3\}$, all b-type vertices to $\{h_1, h_2\}$, all c-type vertices to $\{h_2, h_3\}$ and all d-type vertices to $\{h_0, h_1\}$. We now show that $(p, q) \in R_i$ implies that $(f(p), f(q)) \in S_i$ for $i = 1, \ldots, 4$.

Suppose that $(p, q) \in R_1$. Because p is adjacent to h_0, we find that $g(p) \in \{h_0, h_1, h_3\}$. Because q is adjacent to h_0 and h_2, we find that $g(q) \in \{h_1, h_3\}$. If $g(p) = h_0$, then g maps c_q to h_3, and consequently $g(q) = h_3$. If $g(p) = h_1$, then g maps c_q to h_2, and consequently, d_q to h_1, implying that $g(q) = h_1$. Hence, we find that $(f(p), f(q)) \in \{(0, 3), (1, 1), (3, 1), (3, 3)\} = S_1$, as desired.

Suppose that $(p, q) \in R_2$. Because p is adjacent to h_0 and h_2, we find that $g(p) \in \{h_1, h_3\}$. Because q is adjacent to h_0, we find that $g(q) \in \{h_0, h_1, h_3\}$. If $g(q) = h_0$, then g maps b_p to h_1, and consequently, $g(p) = h_1$. If $g(q) = h_3$, then g maps b_p to h_2, and consequently, a_p to h_3, implying that $g(p) = h_3$.

Hence, we find that $(f(p), f(q)) \in \{(1,0),(1,1),(3,1),(3,3)\} = S_2$, as desired.

Suppose that $(p,q) \in R_3$. Because both p and q are adjacent to both h_0 and h_2, we find that $g(p) \in \{h_1, h_3\}$ and $g(q) \in \{h_1, h_3\}$. If $g(p) = h_1$, then g maps a_p to h_0, and consequently, c_q to h_3, implying that $g(q) = h_3$. Hence, we find that $(f(p), f(q)) \in \{(1,3),(3,1),(3,3)\} = S_3$, as desired.

Suppose that $(p,q) \in R_4$. Because both p and q are adjacent to both h_0 and h_2, we find that $g(p) \in \{h_1, h_3\}$ and $g(q) \in \{h_1, h_3\}$. If $g(q) = h_3$, then g maps d_q to h_0, and consequently, b_p to h_1, implying that $g(p) = h_1$. Hence, we find that $(f(p), f(q)) \in \{(1,1),(1,3),(3,1)\} = S_4$, as desired. This completes the proof of Lemma 5. □

4 Surjective Homomorphisms

Vikas [30] constructed the following graph from a graph $G = (V, E)$ that contains H as an induced subgraph. For each vertex $v \in V_G \backslash V_H$ we add three new vertices u_v, w_v, y_v with edges $h_0u_v, h_0y_v, h_1u_v, h_2w_v, h_2y_v, h_3w_v, u_vv, u_vw_v, u_vy_v, vw_v, w_vy_v$. We say that a vertex u_v, w_v and y_v has *type* u, w, or y, respectively. We also add all edges between any two vertices $u_v, u_{v'}$ and between any two vertices $w_v, w_{v'}$ with $v \neq v'$. For each edge vv' in $E_G \backslash E_H$ we choose an arbitrary orientation, say from v to v', and then add a new vertex $x_{vv'}$ with edges $vx_{vv'}, v'x_{vv'}, u_vx_{vv'}, w_{v'}x_{vv'}$. We say that this new vertex has *type* x. The new graph G' obtained from G is called an *H-compactor* of G. See Figure 5 for an example. This figure does not depict any self-loops, although formally G' must have at least four self-loops, because G' contains G, and consequently, H as an induced subgraph. However, for the same reason as for the H-Retraction problem, this is irrelevant for the Surjective H-Homomorphism problem, and we may assume that G and G' are irreflexive.

Vikas [30] showed that a graph G retracts to H if and only if an (arbitrary) H-compactor G' of G retracts to H if and only if G' compacts to H. Recall that an H-compactor is of diameter 3 as observed by Ito et al. [25]. Our aim is to reduce the diameter in such a graph to 2. This forces us to make a number of modifications. Firstly, we must remove a number of vertices of type x. Secondly, we can no longer choose the orientations regarding the remaining vertices of type x arbitrarily. Thirdly, we must connect the remaining x-type vertices to H via edges. We explain these modifications in detail below.

Let G be a $\mathcal{D}$-graph. For all vertices in G we create vertices of type u, v, w, y with incident edges as in the definition of a compactor. We then perform the following three steps.

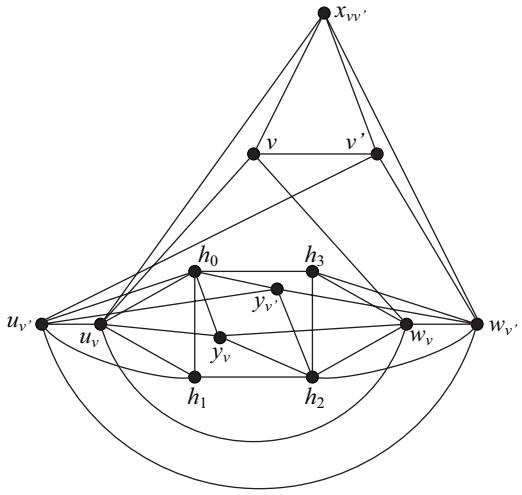

Fig. 5. The part of G' that corresponds to edge $vv' \in E_G \setminus E_H$ as displayed in [30].

1. Not creating all the vertices of type x.
We do not create x-type vertices for the following edges in G: edges between two a-type vertices, edges between two b-type vertices, edges between two c-type vertices, and edges between two d-type vertices. We create x-type vertices for all the other edges in $E_G \setminus E_H$ as explained in Step 2.

2. Choosing the "right" orientation of the other edges of G – H.
For $(p, q) \in R_i$ and $1 \leq i \leq 4$, we choose x-type vertices $x_{a_p p}$, $x_{p b_p}$, $x_{a_p b_p}$, $x_{q c_q}$, $x_{q d_q}$, and $x_{d_q c_q}$. In addition we create the following x-type vertices. For $(p, q) \in R_1$ we choose $x_{p c_q}$. For $(p, q) \in R_2$ we choose $x_{q b_p}$. For $(p, q) \in R_3$ we choose $x_{a_p c_q}$. For $(p, q) \in R_4$ we choose $x_{d_q b_p}$. Note that in this way we have indeed created x-type vertices for all the other edges of $E_G \setminus E_H$.

3. Connecting the created x-type vertices to H.
We add an edge between h_0 and every vertex of type x that we created in Step 2. We also add an edge between h_2 and every such vertex.

We call the resulting graph a *semi-compactor* of G and prove two essential lemmas.

Lemma 6 *Let G be a $\mathcal{D}$-graph. Every semi-compactor of G has diameter 2 and a dominating non-edge.*

PROOF. Let G'' be a semi-compactor of a $\mathcal{D}$-graph G. We first show that G'' has a dominating non-edge. We note that h_0 is adjacent to all vertices except to h_2 and the vertices of type b, c, and w. However, all vertices of type b, c, and w are adjacent to h_2. Because h_0 and h_2 are not adjacent, this means that h_0 and h_2 form a dominating non-edge in G''.

	v	u	w	y	x
v	2	2	2	2	2
u	.	1	2^w	2^{h_0}	2^{h_0}
w	.	.	1	2^{h_2}	2^{h_2}
y	.	.	.	2^{h_0}	2^{h_0}
x	.	.	.	.	2^{h_0}

Table 6
Determining the diameter of a semi-compactor G''.

We show that G'' has diameter 2 in Table 6. In this table, v denotes a vertex of V_G, and u, w, y, x denote vertices of the corresponding type. For reasons of clarity we explain the first row of Table 6 below; superscripts for the other rows are used in the same way as in Table 1.

For the first position of row 1 we use Table 1 to determine an upper bound for the distance between two vertices in G''; hence, there is no superscript for this position. For the second position of row 1 we use the fact that every vertex $v \in V_G \setminus V_H$ is adjacent to u_v and that u_v is adjacent to every other vertex of type u. Furthermore, if $v = h_0$ or $v = h_1$ then v is adjacent to every vertex of type u, and if $v = h_2$ or $v = h_3$ then v is of distance two from every vertex of type u by using h_0 or h_1 as an intermediate vertex, respectively. The third position of row 1 can be explained by similar arguments. The fourth and fifth positions follow from the already deduced property of G'' that h_0 and h_2 form a dominating non-edge combined with the property that every vertex of type x and y is adjacent to both h_0 and h_2. This completes the proof of Lemma 6. □

Lemma 7 *Let G'' be a semi-compactor of a $\mathcal{D}$-graph G. Then the following statements are equivalent:*

(i) G retracts to H;
(ii) G'' retracts to H;
(iii) G'' compacts to H;
(iv) G'' has a vertex-surjective homomorphism to H.

PROOF. We show the following implications: $(i) \Rightarrow (ii)$, $(ii) \Rightarrow (i)$, $(ii) \Rightarrow (iii)$, $(iii) \Rightarrow (ii)$, $(iii) \Rightarrow (iv)$, and $(iv) \Rightarrow (iii)$.

"$(i) \Rightarrow (ii)$" Let f be a retraction from G to H. We show how to extend f to a retraction from G'' to H. We observe that every vertex of type u can only be mapped to h_0 or h_1, because such a vertex is adjacent to h_0 and h_1.

We also observe that every vertex of type w can only be mapped to h_2 or h_3, because such a vertex is adjacent to h_2 and h_3. This implies the following. Let $v \in V_G \setminus V_H$. If $f(v) = h_0$ or $f(v) = h_1$, then w_v must be mapped to h_3 or h_2, respectively. Consequently, u_v must be mapped to h_0 or h_1, respectively, due to the edge $u_v w_v$. If $f(v) = h_2$ or $f(v) = h_3$, then u_v must be mapped to h_1 or h_0, respectively. Consequently, w_v must be mapped to h_2 or h_3, respectively, due to the edge $u_v w_v$. Hence, $f(v)$ fixes the mapping of the vertices u_v and w_v. Moreover, we showed that either u_v is mapped to h_1 or w_v is mapped to h_3. Note that both vertices are adjacent to y_v. Then, because y_v can only be mapped to h_1 or h_3 due to the edges $h_0 y_v$ and $h_2 y_v$, the mapping of y_v is fixed as well; if u_v is mapped to h_1 then y_v is mapped to h_1, and if w_v is mapped to h_3 then y_v is mapped to h_3.

What is left to do is to verify whether we can map the vertices of type x. For this purpose we refer to Table 7, where v, v' denote two adjacent vertices of $V_G \setminus V_H$. Every possible combination of $f(v)$ and $f(v')$ corresponds to a row in this table. As we have just shown, this fixes the image of the vertices u_v, $u_{v'}$, w_v, $w_{v'}$, $y_{v'}$ and y_v. For $x_{vv'}$ we use its adjacencies to v, v', u_v and $w_{v'}$ to determine potential images. For some cases, this number of potential images is not one but two. This is shown in the last column of Table 7; here we did not take into account that every $x_{vv'}$ is adjacent to h_0 and h_2 in our construction. Because of these adjacencies, every $x_{vv'}$ can only be mapped to h_1 or h_3. In the majority of the 12 rows in Table 7 we have this choice; the exceptions are row 4 and row 9. In rows 4 and 9, we find that $x_{vv'}$ can only be mapped to one image, which is h_0 or h_2, respectively. We will show that neither row can occur.

By Steps 1-2 of the definition of a semi-compactor, we have that (v, v') belongs to

$$\{(a_p, p), (p, b_p), (a_p, b_p), (q, c_q), (q, d_q), (d_q, c_q), (p, c_q), (q, b_p), (a_p, c_q), (d_q, b_p)\}.$$

We first show that row 4 cannot occur. In order to obtain a contradiction, suppose that row 4 does occur, i.e., that $f(v) = h_1$ and $f(v') = h_0$ for some $v, v' \in V_G \setminus V_H$. Due to their adjacencies with vertices of H, every vertex of type a is mapped to h_0 or h_3, every vertex of type b to h_1 or h_2, every vertex of type c to h_2 or h_3 and every vertex of type d to h_0 or h_1. This means that v can only be p, q, b_p, or d_q, whereas v' can only be p, q, a_p or d_q. If $v = p$ then $v' \in \{b_p, c_q\}$. If $v = q$ then $v' \in \{c_q, d_q, b_p\}$. If $v = b_p$ then v' cannot be chosen. If $v = d_q$ then $v' \in \{c_q, b_p\}$. Hence, we find that $v = q$ and $v' = d_q$. However, then f is not a retraction from G to H, because c_q is adjacent to d_q, q, h_2, h_3, and f maps these vertices to h_0, h_1, h_2, h_3, respectively. Hence, row 4 does not occur.

We now show that row 9 cannot occur. In order to obtain a contradiction, suppose that row 9 does occur, i.e., that $f(v) = h_2$ and $f(v') = h_3$. As in the

v	v'	u_v	$u_{v'}$	w_v	$w_{v'}$	y_v	$y_{v'}$	$x_{vv'}$
h_0	h_0	h_0	h_0	h_3	h_3	h_3	h_3	h_0/h_3
h_0	h_1	h_0	h_1	h_3	h_2	h_3	h_1	h_1
h_0	h_3	h_0	h_0	h_3	h_3	h_3	h_3	h_0/h_3
h_1	h_0	h_1	h_0	h_2	h_3	h_1	h_3	h_0
h_1	h_1	h_1	h_1	h_2	h_2	h_1	h_1	h_1/h_2
h_1	h_2	h_1	h_1	h_2	h_2	h_1	h_1	h_1/h_2
h_2	h_1	h_1	h_1	h_2	h_2	h_1	h_1	h_1/h_2
h_2	h_2	h_1	h_1	h_2	h_2	h_1	h_1	h_1/h_2
h_2	h_3	h_1	h_0	h_2	h_3	h_1	h_3	h_2
h_3	h_0	h_0	h_0	h_3	h_3	h_3	h_3	h_0/h_3
h_3	h_2	h_0	h_1	h_3	h_2	h_3	h_1	h_3
h_3	h_3	h_0	h_0	h_3	h_3	h_3	h_3	h_0/h_3

Table 7
Determining a retraction from G'' to H.

previous case, we deduce that every vertex of type a is mapped to h_0 or h_3, every vertex of type b to h_1 or h_2, every vertex of type c to h_2 or h_3 and every vertex of type d to h_0 or h_1. Moreover, every vertex of type ℓ or r cannot be mapped to h_2, because it is adjacent to h_0. Then v can only be b_p or c_q, and v' can only be p, q, a_p or c_q. However, if $v = b_p$ or $v = c_q$ then v' cannot be chosen. Hence, row 9 cannot occur, and we conclude that f can be extended to a retraction from G'' to H, as desired.

"$(ii) \Rightarrow (i)$" Let f be a retraction from G'' to H. Then the restriction of f to V_G is a retraction from G to H. Hence, this implication is valid.

"$(ii) \Rightarrow (iii)$" This implication is valid, because every retraction from G'' to H is an edge-surjective homomorphism, so *a fortiori* a compaction from G'' to H.

"$(iii) \Rightarrow (ii)$" Let f be a compaction from G'' to H. We will show that f is without loss of generality a retraction from G'' to H. Our proof goes along the same lines as the proof of Lemma 2.1.2 in Vikas [30], i.e., we use the same arguments but in addition we must examine a few more cases due to our modifications in steps 1–3; we therefore include all the proof details below.

We let U consist of h_0, h_1 and all vertices of type u. Similarly, we let W consist

of h_2, h_3 and all vertices of type w. Because U forms a clique in G, we find that $f(U)$ is a clique in H. This means that $1 \leq |f(U)| \leq 2$. By the same arguments, we find that $1 \leq f(W) \leq 2$.

We first prove that $|f(U)| = |f(W)| = 2$. In order to derive a contradiction, suppose that $|f(U)| \neq 2$. Then $f(U)$ has only one vertex. By symmetry, we may assume that f maps every vertex of U to h_0; otherwise we can redefine f. Because every vertex of G'' is adjacent to a vertex in U, we find that G'' contains no vertex that is mapped to h_2 by f. This is not possible, because f is a compaction from G'' to H. Hence $|f(U)| = 2$, and by the same arguments, $|f(W)| = 2$. Because U is a clique, we find that $f(U) \neq \{h_0, h_2\}$ and $f(U) \neq \{h_1, h_3\}$. Hence, by symmetry, we assume that $f(U) = \{h_0, h_1\}$.

We now prove that $f(W) = \{h_2, h_3\}$. In order to obtain a contradiction, suppose that $f(W) \neq \{h_2, h_3\}$. Because f is a compaction from G'' to H, there exists an edge st in G'' with $f(s) = h_2$ and $f(t) = h_3$. Because $f(U)$ only contains vertices mapped to h_0 or h_1, we find that $s \notin U$ and $t \notin U$. Because we assume that $f(W) \neq \{h_2, h_3\}$, we find that st is not one of $w_v w_{v'}, w_v h_2, w_v h_3, h_2 h_3$. Hence, st is one of the following edges

$$vw_v, w_v y_v, v x_{vv'}, y_v h_2, v h_2, v h_3, vv', v' x_{vv'}, w_{v'} x_{vv'}, x_{vv'} h_2,$$

where $v, v' \in V_G \setminus V_H$. We must consider each of these possibilities.

If $st \in \{vw_v, w_v y_v, v x_{vv'}\}$ then $f(u_v) \in \{h_2, h_3\}$, because u_v is adjacent to $v, w_v, y_v, x_{vv'}$. However, this is not possible because $f(u_v) \in \{h_0, h_1\}$.

If $st = y_v h_2$, then $f(w_v) = h_2$ or $f(w_v) = h_3$, because w_v is adjacent to both y_v and h_2, and $\{f(y_v), f(h_2)\} = \{h_2, h_3\}$. This means that either $f(w_v) = f(y_v)$ or $f(w_v) = f(h_2)$. If $f(w_v) = f(y_v)$, then $\{f(w_v), f(h_2)\} = \{h_2, h_3\}$. Consequently, $f(W) = \{h_2, h_3\}$, which we assumed is not the case. Hence, $f(w_v) \neq f(y_v)$. Then f maps the edge $w_v y_v$ to $h_2 h_3$, and we return to the previous case. We can repeat the same arguments if $st = v h_2$ or $st = v h_3$. Hence, we find that st cannot be equal to those edges either.

If $st = vv'$, then by symmetry we may assume without loss of generality that $f(v) = h_2$ and $f(v') = h_3$. Consequently, $f(u_v) = h_1$, because $u_v \in U$ is adjacent to v, and can only be mapped to h_0 or h_1. By the same reasoning, $f(u_{v'}) = h_0$. Because w_v is adjacent to v with $f(v) = h_2$ and to u_v with $f(u_v) = h_1$, we find that $f(w_v) \in \{h_1, h_2\}$. Because $w_{v'}$ is adjacent to v' with $f(v') = h_3$ and to $u_{v'}$ with $f(u_{v'}) = h_0$, we find that $f(w_{v'}) \in \{h_0, h_3\}$. Recall that $f(W) \neq \{h_2, h_3\}$. Then, because w_v and $w_{v'}$ are adjacent, we find that $f(w_v) = h_1$ and $f(w_{v'}) = h_0$. Suppose that $x_{vv'}$ exists. Then $x_{vv'}$ is adjacent to vertices v with $f(v) = h_2$, to v' with $f(v') = h_3$, to u_v with $f(u_v) = h_1$ and to $w_{v'}$ with $f(w_{v'}) = h_0$. This is not possible. Hence $x_{vv'}$ cannot exist. This means that v, v' are both of type a, both of type b, both

of type c or both of type d. If v, v' are both of type a or both of type d, then $f(h_0) \in \{h_2, h_3\}$, which is not possible because $h_0 \in U$ and $f(U) = \{h_0, h_1\}$. If v, v' are both of type b, we apply the same reasoning with respect to h_1. Suppose that v, v' are both of type c. Then both v and v' are adjacent to h_2. This means that $f(h_2) \in \{h_2, h_3\}$. Then either $\{f(v), f(h_2)\} = \{h_2, h_3\}$ or $\{f(v'), f(h_2)\} = \{h_2, h_3\}$. Hence, by considering either the edge vh_2 or $v'h_2$ we return to a previous case. We conclude that $st \neq vv'$.

If $st = v'x_{vv'}$ then $f(v) \in \{h_2, h_3\}$, because v is adjacent to v' and $x_{vv'}$. Then one of vv' or $vx_{vv'}$ maps to h_2h_3, and we can return to a previous case. Hence, we find that $st \neq v'x_{vv'}$.

If $st = w_{v'}x_{vv'}$ then $f(v') \in \{h_2, h_3\}$, because v' is adjacent to $w_{v'}$ and $x_{vv'}$. Then one of $v'w_{v'}$ or $v'x_{vv'}$ maps to h_2h_3, and we can return to a previous case. Hence, we find that $st \neq w_{v'}x_{vv'}$.

If $st = x_{vv'}h_2$ then $f(w_{v'}) \in \{h_2, h_3\}$, because $w_{v'}$ is adjacent to $x_{vv'}$ and h_2. Because $f(W) \neq \{h_2, h_3\}$, we find that $f(w_{v'}) = f(h_2)$. Then $w_{v'}x_{vv'}$ is mapped to h_2h_3, and we return to a previous case. Hence, we find that $st \neq x_{vv'}h_2$.

We conclude that G'' has no edge st with $f(s) = h_2$ and $f(t) = h_3$. This is a contradiction; recall that f is a compaction. Hence, $f(W) = \{h_2, h_3\}$.

The next step is to prove that $f(h_0) \neq f(h_1)$. In order to obtain a contradiction, suppose that $f(h_0) = f(h_1)$. By symmetry we may assume without loss of generality that $f(h_0) = f(h_1) = h_0$. Because $f(U) = \{h_0, h_1\}$, there exists a vertex u_v with $f(u_v) = h_1$. Because w_v with $f(w_v) \in \{h_2, h_3\}$ is adjacent to u_v, we find that $f(w_v) = h_2$. Because h_2 with $f(h_2) \in \{h_2, h_3\}$ is adjacent to h_1 with $f(h_1) = h_0$, we find that $f(h_2) = h_3$. However, then y_v is adjacent to h_0 with $f(h_0) = h_0$, to u_v with $f(u_v) = h_1$, to w_v with $f(w_v) = h_2$, and to h_2 with $f(h_2) = h_3$. This is not possible. Hence, we find that $f(h_0) \neq f(h_1)$. By symmetry, we may assume without loss of generality that $f(h_0) = h_0$ and $f(h_1) = h_1$.

We are left to show that $f(h_2) = h_2$ and $f(h_3) = h_3$. This can be seen as follows. Because h_2 is adjacent to h_1 with $f(h_1) = h_1$, and $f(h_2) \in \{h_2, h_3\}$ we find that $f(h_2) = h_2$. Because h_3 is adjacent to h_0 with $f(h_0) = h_0$, and $f(h_3) \in \{h_2, h_3\}$ we find that $f(h_3) = h_3$. Hence, we have found that f is a retraction from G'' to H, as desired.

"$(iii) \Rightarrow (iv)$" and "$(iv) \Rightarrow (iii)$" immediately follow from the equivalence between statements 3 and 6 in Proposition 1, after recalling that G'' has diameter 2 due to Lemma 6. $\square$

We are now ready to state the main result of our paper. Its proof follows from Lemmas 6 and 7, in light of Theorem 5; note that all constructions may be carried out in polynomial time.

Theorem 8 *The* Surjective $\mathcal{C}_4$-Homomorphism *problem is* NP*-complete for graphs of diameter 2 even if they have a dominating non-edge.*

Acknowledgments. The authors thank Andrei Krokhin for useful comments on Section 2 and an anonymous reviewer for helpful comments on the presentation of our paper.

References

[1] T. Asano and T. Hirata, Edge-contraction problems, *Journal of Computer and System Sciences* **26** (1983) 197–208.

[2] A. Brandstädt, F.F. Dragan, V.B. Le and T. Szymczak, On stable cutsets in graphs, *Discrete Applied Mathematics* **105** (2000) 39–50.

[3] A. Bulatov, Tractable conservative constraint satisfaction problems, In: Proceedings of LICS 2003 (2003) 321–330.

[4] A. Bulatov, A. Krokhin and P. G. Jeavons, Classifying the complexity of constraints using finite algebras, *SIAM Journal on Computing* **34** (2005) 720–742.

[5] K. Cameron, E. M. Eschen, C. T. Hoáng, and R. Sritharan, The complexity of the list partition problem for graphs, *SIAM Journal on Discrete Mathematics* **21** (2007) 900–929.

[6] K. Cook, S. Dantas, E.M. Eschen, L. Faria, C.M.H. de Figueiredo and S. Klein, $2K_2$ vertex-set partition into nonempty parts, *Discrete Mathematics* **310** (2010) 1259–1264.

[7] V. Chvátal, Recognizing decomposable graphs, *Journal of Graph Theory* **8** (1984) 51–53.

[8] S. Dantas, C.M.H. de Figueiredo, S. Gravier and S. Klein, Finding H-partitions efficiently, *RAIRO - Theoretical Informatics and Applications* **39** (2005) 133–144.

[9] S. Dantas, F. Maffray and A. Silva, $2K_2$-partition of some classes of graphs, *Discrete Applied Mathematics* **160** (2012) 2662–2668.

[10] D. Eppstein, Finding Large Clique Minors is Hard, *Journal of Graph Algorithms and Applications* **13** (2009) 197–204.

[11] T. Feder and P. Hell, List homomorphisms to reflexive graphs, *Journal of Combinatorial Theory, Series B* **72** (1998) 236–250.

[12] T. Feder, P. Hell, S. Klein and R. Motwani, List partitions, *SIAM Journal on Discrete Mathematics* **16** (2003) 449–478.

[13] C.M.H. de Figueiredo, The P versus NP-complete dichotomy of some challenging problems in graph theory, *Discrete Applied Mathematics* **160** (2012) 2681–2693.

[14] C.M.H. de Figueiredo, S. Klein and B. Reed, Finding skew partitions efficiently, *Journal of Algorithms* **37** (2000) 505–521.

[15] H. Fleischner, E. Mujuni, D. Paulusma and S. Szeider, Covering graphs with few complete bipartite subgraphs, *Theoretical Computer Science* **410** (2009) 2045–2053.

[16] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

[17] P.A. Golovach, M. Kaminski, D. Paulusma and D.M. Thilikos, Increasing the minimum degree of a graph by contractions, *Theoretical Computer Science* **481** (2013) 74–84.

[18] P.A. Golovach, D. Paulusma and J. Song, Computing vertex-surjective homomorphisms to partially reflexive trees, *Theoretical Computer Science* **457** (2012) 86–100.

[19] R. Hammack, A note on the complexity of computing cyclicity, *Ars Combinatoria* **63** (2002) 89–95.

[20] P. Heggernes, P. van 't Hof, B. Lévêque, D. Lokshtanov and C. Paul. Contracting graphs to paths and trees, *Algorithmica* **68** (2014) 109–132.

[21] P. Heggernes, P. van 't Hof, D. Lokshtanov and C. Paul. Obtaining a bipartite graph by contracting few edges, *SIAM Journal on Discrete Mathematics* **27** (2013) 2143–2156.

[22] P. Hell and J. Nešetřil, On the complexity of H-colouring, *Journal of Combinatorial Theory, Series B* **48** (1990) 92–110.

[23] P. van 't Hof, D. Paulusma and G.J. Woeginger, Partitioning graphs into connected parts, *Theoretical Computer Science* **410** (2009) 4834–4843.

[24] T. Ito, M. Kamiński, D. Paulusma and D. M. Thilikos, Parameterizing cut sets in a graph by the number of their components, *Theoretical Computer Science* **412** (2011) 6340–6350.

[25] T. Ito, M. Kaminski, D. Paulusma and D.M. Thilikos, On disconnected cuts and separators, *Discrete Applied Mathematics* **159** (2011) 1345–1351.

[26] R. M. Karp, Reducibility among combinatorial problems, In: Complexity of Computer Computations (1972) 85–103.

[27] W.S. Kennedy and B. Reed, Fast skew partition recognition, In: Computational Geometry and Graph Theory, Lecture Notes in Computer Science **4535** (2008) 101–107.

[28] B. Martin and D. Paulusma, The computational complexity of Disconnected Cut and $2K_2$-Partition, In: Proceedings of CP 2011, Lecture Notes in Computer Science **6876** (2011) 561–575.

[29] R. B. Teixeira, S. Dantas and C.M.H. de Figueiredo, The external constraint 4 nonempty part sandwich problem, *Discrete Applied Mathematics* **159** (2011) 661–673.

[30] N. Vikas, Computational complexity of compaction to reflexive cycles, *SIAM Journal on Computing* **32** (2002) 253–280.

[31] N. Vikas, Compaction, retraction, and constraint satisfaction, *SIAM Journal on Computing* **33** (2004) 761–782.

[32] N. Vikas, A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results, *Journal of Computer and System Sciences* **71** (2005) 406–439.

[33] N. Vikas, Algorithms for partition of some class of graphs under compaction, In: Proceedings of COCOON 2011, Lecture Notes in Computer Science **6842** (2011) 319–330.

[34] N. Vikas, Algorithms for partition of some class of graphs under compaction and vertex-compaction, Algorithmica **67** (2013) 180–206.

[35] S.H. Whitesides, An algorithm for finding clique cut-sets, *Information Processing Letters* **12** (1981) 31–32.