Knocking Out P_k-free Graphs*

Matthew Johnson, Daniël Paulusma**, and Anthony Stewart

School of Engineering and Computing Sciences, Durham University,
South Road, Durham, DH1 3LE, U.K.
{matthew.johnson2,daniel.paulusma,a.g.stewart}@durham.ac.uk

Abstract. A parallel knock-out scheme for a graph proceeds in rounds in each of which each surviving vertex eliminates one of its surviving neighbours. A graph is KO-reducible if there exists such a scheme that eliminates every vertex in the graph. The Parallel Knock-Out problem is to decide whether a graph G is KO-reducible. This problem is known to be NP-complete and has been studied for several graph classes. We show that the problem is NP-complete even for split graphs, a subclass of P_5-free graphs. In contrast, our main result is that it is linear-time solvable for P_4-free graphs (cographs).

1 Introduction

We consider *parallel knock-out schemes* for finite undirected graphs with no self-loops and no multiple edges. These schemes, which were introduced by Lampert and Slater [17], proceed in rounds. In the first round each vertex in the graph selects exactly one of its neighbours, and then all the selected vertices are eliminated simultaneously. In subsequent rounds this procedure is repeated in the subgraph induced by those vertices not yet eliminated. The scheme continues until there are no vertices left, or until an isolated vertex is obtained (since an isolated vertex will never be eliminated). A graph is called *KO-reducible* if there exists a parallel knock-out scheme that eliminates the whole graph. The *parallel knock-out number* of a graph G, denoted by $\text{pko}(G)$, is the minimum number of rounds in a parallel knock-out scheme that eliminates every vertex of G. If G is not KO-reducible, then $\text{pko}(G) = \infty$.

Examples. Every graph G with a hamiltonian cycle has $\text{pko}(G) = 1$, as each vertex can select its successor on a hamiltonian cycle C of G after fixing some orientation of C. Also every graph G with a perfect matching has $\text{pko}(G) = 1$, as each vertex can select its matching neighbour in the perfect matching. In fact it is not difficult to see [2] that a graph G has $\text{pko}(G) = 1$ if and only if G contains a *[1,2]-factor*, that is, a spanning subgraph in which every component is either a cycle or an edge.

We study the computational complexity of the Parallel Knock-Out problem, which is the problem of deciding whether a given graph is KO-reducible. Our

* An extended abstract of this paper appeared in the proceedings of MFCS 2014 [15].
** Supported by EPSRC grant EP/K025090/1.

main motivation is the close relationship with cycles and matchings as illustrated by the above examples. We also consider the variant in which the number of rounds permitted is fixed. This problem is known as the k-Parallel Knock-Out problem, which has as input a graph G and ask whether $\mathrm{pko}(G) \leq k$ for some fixed integer k (i.e. that is not part of the input).

Known Results. The 1-Parallel Knock-Out problem is equivalent [2] to testing whether a graph has a $[1,2]$-factor, which is well-known to be polynomial-time solvable (see e.g. [3] for a proof). However, both the problems Parallel Knock-Out and k-Parallel Knock-Out with $k \geq 2$ are NP-complete even for bipartite graphs [3]. On the other hand, it is known that Parallel Knock-Out and k-Parallel Knock-Out (for all $k \geq 1$) can be solved in $O(n^{3.5} \log^2 n)$ time on trees [2]. These results were later extended to graph classes of bounded treewidth [3]. It remains *open* whether a further generalization is possible to graph classes of bounded clique-width. Broersma et al. in [4] gave an $O(n^{5.376})$ time algorithm for solving Parallel Knock-Out on n-vertex claw-free graphs. Later this was improved to an $O(n^2)$ time algorithm for almost claw-free graphs (which generalize the class of claw-free graphs) [16]. The latter paper also gives a full characterization of connected almost claw-free graphs that are KO-reducible. In particular it shows that every KO-reducible almost claw-free graph has parallel knock-out number at most 2. In general, KO-reducible graphs (even KO-reducible trees [2]) may have an arbitrarily large parallel knock-out number. Broersma et al. [4] showed that a KO-reducible n-vertex graph G has $\mathrm{pko}(G) \leq \min\{-\frac{1}{2} + (2n - \frac{7}{4})^{\frac{1}{2}}, \frac{1}{2} + (2\alpha - \frac{7}{4})^{\frac{1}{2}}\}$ (where α denotes the size of a largest independent set in G). This bound is asymptotically tight for complete bipartite graphs [2]. Broersma et al. [4] also showed that every KO-reducible graph with no induced $(p+1)$-vertex star $K_{1,p}$ has parallel knock-out number at most $p-1$.

Our Results. We address the open problem of whether Parallel Knock-Out is polynomial-time solvable on graph classes whose clique-width is bounded by a constant. This seems a very challenging problem, and in this paper we focus on graphs of clique-width at most 2. It is known that a graph has clique-width at most 2 if and only if it is a cograph [7]. Cographs are also known as P_4-free graphs (a graph is called P_k-free if it has no induced k-vertex path).

In Section 3 we give a linear-time algorithm for solving the Parallel Knock-Out problem on cographs. The first step of the algorithm is to compute the cotree of a cograph. It then traverses the cotree twice. The first time to compute to what extent "large" subgraphs can be reduced by themselves and how many free "firings" from outside are available. The second time to check whether the number of free external firings is sufficient to knock them out. In this way it will be verified whether the whole graph is KO-reducible. In Section 4 we prove that both the Parallel Knock-Out problem and the k-Parallel Knock-Out problem ($k \geq 2$) are NP-complete even for split graphs. Because split graphs are P_5-free, our results imply a dichotomy result for the computational complexity of the Parallel Knock-Out problem restricted to P_k-free graphs, as shown in Section 5, where we also give some (other) open problems.

2 Preliminaries

We denote a graph by $G = (V(G), E(G))$ and write $|G| = |V(G)|$ to denote the order of G. An edge joining vertices u and v is denoted by uv. If not stated otherwise a graph is assumed to be finite, undirected and simple.

Let $G = (V, E)$ be a graph. The *neighbourhood* of $u \in V$, that is, the set of vertices adjacent to u is denoted by $N_G(u) = \{v \,|\, uv \in E\}$. For a subset $S \subseteq V$, we let $G[S]$ denote the *induced* subgraph of G, which has vertex set S and edge set $\{uv \in E \mid u, v \in S\}$. A set $I \subseteq V$ is called an *independent set* of G if no two vertices in I are adjacent to each other. A subset $C \subseteq V$ is called a *clique* of G if any two vertices in C are adjacent to each other. A subset $D \subseteq V$ is a *dominating set* of a graph $G = (V, E)$ if every vertex of G is in D or adjacent to a vertex in D.

The *union* of two graphs G and H is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. If $V(G) \cap V(H) = \emptyset$, then we say that the union of G and H is *disjoint* and write $G + H$. We denote the disjoint union of r copies of G by rG.

For $n \geq 1$, the graph P_n denotes the *path* on n vertices, that is, $V(P_n) = \{u_1, \dots, u_n\}$ and $E(P_n) = \{u_i u_{i+1} \mid 1 \leq i \leq n-1\}$. For $n \geq 3$, the graph C_n denotes the *cycle* on n vertices, that is, $V(C_n) = \{u_1, \dots, u_n\}$ and $E(C_n) = \{u_i u_{i+1} \mid 1 \leq i \leq n-1\} \cup \{u_n u_1\}$. The graph K_n denotes the *complete graph* on n vertices, that is, the n-vertex graph whose vertex set is a clique. A graph is *complete bipartite* if its vertex set can be partitioned into two classes such that two vertices u and v are adjacent if and only if u and v belong to different classes. The graph $K_{p,q}$ is the *complete bipartite graph* with partition classes of sizes p and q, respectively; the graph $K_{1,q}$ is also called the *star* on $q+1$ vertices.

Let G be a graph and let $\{H_1, \dots, H_p\}$ be a set of graphs. We say that G is $(H_1, \dots, H_p)$*-free* if G has no induced subgraph isomorphic to a graph in $\{H_1, \dots, H_p\}$. If $p = 1$ we may write H_1-free instead of (H_1)-free. A P_4-free graph is also called a *cograph*. A graph G is a *split graph* if its vertex set can be partitioned into a clique and an independent set. Split graphs coincide with $(2K_2, C_4, C_5)$-free graphs [10]; note that this implies that every split graph is $2K_2$-free and thus P_5-free.

We also need some formal terminology for parallel knock-out schemes. For a graph $G = (V, E)$, a *KO-selection* is a function $f : V \to V$ with $f(v) \in N(v)$ for all $v \in V$. If $f(v) = u$, we say that vertex v *fires at* vertex u, or that u *is knocked out* by a *firing* of v. If $u \in U$ for some $U \subseteq V$ then the firing is said to be *internal* with respect to U if $v \in U$; otherwise it is said to be *external* (with respect to U).

For a KO-selection f, we define the corresponding *KO-successor* of G as the subgraph of G that is induced by the vertices in $V \setminus f(V)$; if G' is the KO-successor of G we write $G \rightsquigarrow G'$. Note that every graph without isolated vertices has at least one KO-successor. A sequence

$$G \rightsquigarrow G^1 \rightsquigarrow G^2 \rightsquigarrow \cdots \rightsquigarrow G^s,$$

is called a *parallel knock-out scheme* or *KO-scheme*. A KO-scheme in which G^s is the null graph $(\emptyset, \emptyset)$ is called a *KO-reduction scheme*; in that case G is also called *KO-reducible*. A single step in a KO-scheme is called a *(firing) round*. Recall that the parallel knock-out number of G, $\text{pko}(G)$, is the smallest number of rounds of any KO-reduction scheme, and that if G is not KO-reducible then $\text{pko}(G) = \infty$.

We will use the following result of Broersma et al. [2].

Lemma 1 ([2]). *Let p and q be two integers with $0 < p \leq q$. Then $K_{p,q}$ is KO-reducible if and only if $pko(K_{p,q}) \leq p$ if and only if $q \leq \frac{1}{2}p\,(p+1)$.*

3 Cographs

In this section we show that PARALLEL KNOCK-OUT can be solved in linear time for cographs. For doing so we need to introduce some extra notation and terminology.

Let G_1 and G_2 be two disjoint graphs. The *join* operation $\otimes$ adds an edge between every vertex of G_1 and every vertex of G_2. The *union* operation $\oplus$ creates the disjoint union of G_1 and G_2 (note that we may also write $G_1 + G_2$ instead of $G_1 \oplus G_2$).

It is well known (see, for example, [1]) that a graph G is a cograph if and only if G can be generated from K_1 by a sequence of operations, where each operation is either a join or a union. Such a sequence corresponds to a decomposition tree, which has the following properties:

1. its root r corresponds to the graph $G_r = G$;
2. every leaf x of it corresponds to exactly one vertex of G, and vice versa, implying that x corresponds to a unique single-vertex graph G_x;
3. every internal node x has at least two children, is either labeled $\oplus$ or $\otimes$, and corresponds to an induced subgraph G_x of G defined as follows:
   - if x is a $\oplus$-node, then G_x is the disjoint union of all graphs G_y where y is a child of x;
   - if x is a $\otimes$-node, then G_x is the join of all graphs G_y where y is a child of x.

A cograph G may have more than one such tree but has exactly one unique tree [5], called a *cotree*, if the following additional property is required:

4. Labels of internal nodes on the (unique) path from any leaf to r alternate between $\oplus$ and $\otimes$.

We denote the cotree of a cograph G by T_G; see Figures 1 and 2 for an example of a cograph with its corresponding cotree.

We use the following result of Corneil, Perl and Stewart [6] as a lemma.

Lemma 2 ([6]). *Let G be a graph with n vertices and m edges. Deciding if G is a cograph and constructing T_G (if it exists) can be done in time $O(n+m)$.*

We now present our algorithm, which we call `Cograph-PKO`, for solving Parallel Knock-Out on cographs.

Sketch We start by giving some intuition. Let G be a cograph. We may assume without loss of generality that G is connected, as otherwise we could consider each connected component of G separately. We first construct the cotree T_G. Because G is connected, the root r of T_G is a $\otimes$-node. Recall that $G_r = G$ by definition. Consider a partition (X, Y) of the set of children of r such that

$$p = \sum_{x \in X} |G_x| \leq \sum_{y \in Y} |G_y| = q.$$

Note that G has a spanning complete bipartite graph with partition classes $\bigcup_{x \in X} V(G_x)$ and $\bigcup_{y \in Y} V(G_y)$. Hence, if $q \leq \frac{1}{2}p(p+1)$ then G is KO-reducible by Lemma 1. However, such a partition (X, Y) need not exist, but G might still be KO-reducible. In order to find out, we must analyze the cotree of G at lower levels.

The main idea behind our algorithm is as follows. As mentioned above, the graph G_x corresponding to a join node x has at least one spanning complete bipartite subgraph. We will show that it is sufficient to consider only bipartitions, in which one bipartition class corresponds to a single child z of x. We chose z in such a way that if the corresponding complete bipartite subgraph is unbalanced (with respect to the ratio prescribed in Lemma 1) then the vertices of G_z correspond to a "large" bipartition class. We will then try to reduce G_z as much as possible by internal firings only. If G_z cannot be reduced to the empty graph, then external firings are needed. In particular, some of these external firings will be internal firings for supergraphs of G_z. Hence, we first traverse T_G from top to bottom, starting with the root r, to determine the number of external firings for each graph G_z. Afterwards we can then use a bottom-up approach, starting with the leaves of T_G, to determine the number of vertices a graph G_z can be reduced to by internal firings only. If this number is zero for r then G is KO-reducible; otherwise it is not.

Full Description Let G be a connected cograph, and let $x \in V(T_G)$. We say that $|G_x|$ is the *size* of x. We fix a *largest* child of x, that is, a child of x with largest size over all children of x. We denote this child by $z(x)$ (if there is more than one largest child we pick an arbitrary largest one). Let $C(x)$ consist of all other children of x in T_G (so excluding z). We write $F(x) = \sum_{y \in C(x)} G_y$.

In our algorithm we recursively define two functions f and l that assign a positive integer to the nodes of $V(T_G)$. We write $f(x) = \perp$ or $l(x) = \perp$ if we have not yet assigned an integer $f(x)$ or $l(x)$ to node x; for some nodes x our algorithm might never do this (as we shall see, l will define an integer to a node x if and only if f has previously done so). The meaning of these two functions will be made more clear later. In particular, we will show that $f(x)$ (if defined) is the the number of vertices in $V(G) \setminus V_x$ adjacent to each vertex of V_x. This function will help us in determining how many additional internal firing rounds we have when we expand G_x to a larger subgraph of G by moving up the tree.

The integer $l(x)$ (if defined) is, as we will prove, equal to the smallest number of vertices in G_x that cannot be knocked out internally (that is, within G_x) by any KO-scheme of G. We will show that $l(r)$ is defined, that is, $l(r) \neq \perp$. Hence, there exists a KO-scheme that knocks out all vertices of $V(G_r) = V(G)$ if and only if $l(r) = 0$.

`Cograph-PKO`

input : a connected cograph G
output : `yes` if G is KO-reducible; `no` otherwise

Step 1. Compute the size $|G_x|$ for all $x \in V(T_G)$.

Step 2. Recursively define a function f. Initially set $f(x) := \perp$ for all $x \in V(T_G)$. Set $f(r) := 0$. Now let x be a vertex in T_G with $f(x) \neq \perp$.

2a. If x is a $\oplus$-node: $f(y) := f(x)$ for all $y \in C(x) \cup \{z(x)\}$.

2b. If x is a $\otimes$-node: $f(z(x)) := f(x) + |F(x)|$.

Step 3. Let $B = \{\ell \mid \ell \text{ is a leaf of } T_G \text{ with } f(\ell) \neq \perp\}$.

Step 4. Recursively define a function l. Initially set $l(x) := \perp$ for all $x \in V(T_G)$. Set $l(\ell) := 1$ for all $\ell \in B$. Now let x be a vertex in T that is either a $\oplus$-node with $l(y) \neq \perp$ for all $y \in C(x) \cup \{z(x)\}$ or a $\otimes$-node with $l(z(x)) \neq \perp$.

4a. If x is a $\oplus$-node: $l(x) := l(z(x)) + \sum_{y \in C(x)} l(y)$.

4b. If x is a $\otimes$-node: $l(x) := \max\{0,\ l(z(x)) - f(x) \cdot |F(x)| - \frac{1}{2}|F(x)|(|F(x)|+1)\}$.

Step 5. If $l(r) = 0$ then return `yes`; otherwise return `no`.

Note that for some $x \in V(T_G)$, it may happen indeed that $f(x) = \perp$ or $l(x) = \perp$ holds (for example, if x is a leaf node not in B then $l(x) = \perp$).

Example. We illustrate the working of `Cograph-PKO` by applying it on the graph G of Figure 1 gives Figure 2, in which the size, f-value and l-value are displayed except for the leaves and all y-nodes except y_3. The other y-nodes have the same f-value and l-value as y_3, namely equal to $\perp$ (our algorithm does not need to compute f and l for these nodes). Because $l(r) = 0$ we conclude that G is KO-reducible. Indeed this can also be seen as follows.

Round 1: $v_1, \ldots, v_6$ fire at v_{17}; v_7 fires at v_1; v_8 fires at v_2; v_9 and v_{13} fire at each other; v_{10} and v_{14} fire at each other; v_{11} and v_{15} fire at each other; v_{12} and v_{16} fire at each other; v_{17} fires at v_3.

Round 2: v_4, v_5, v_6 fire at v_7; v_7 fires at v_4; v_8 fires at v_5.

Round 3: v_6 and v_8 fire at each other.

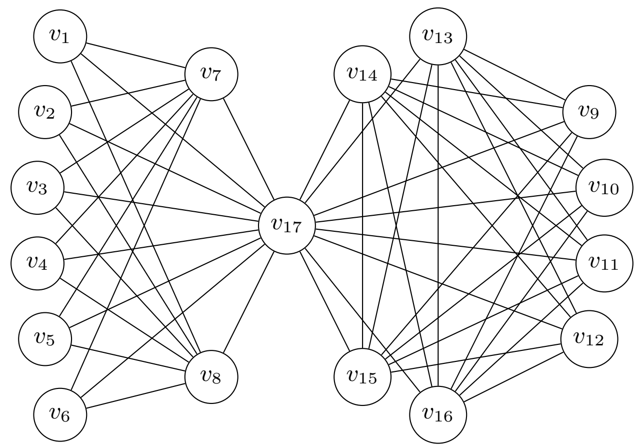

Fig. 1. An example of a cograph G.

In order to prove correctness of `Cograph-PKO` we need some new terminology and a number of lemmas. Let x be a node in T_G. From now on we write $V_x = V(G_x)$. We say that a vertex $v \in V(G)$ is *complete* to a set $U \subseteq V(G)$ with $v \notin U$ if v is adjacent to all vertices of U.

Lemma 3. *Let $x \in V(T_G)$ with $f(x) \neq \bot$. The following two statements hold:*

(i) any vertex in $V(G) \setminus V_x$ adjacent to a vertex of V_x is complete to V_x;
(ii) the number of vertices in $V(G) \setminus V_x$ complete to V_x is equal to $f(x)$.

Proof. Let $x \in V(T_G)$ with $f(x) \neq \bot$. Statement (i) follows from the definition of T_G. We prove (ii) as follows. Let $\mathrm{dist}(x, r)$ denote the distance between x and r in T_G. We use induction on $\mathrm{dist}(x, r)$. The claim is true for $\mathrm{dist}(x, r) = 0$ because in that case $x = r$ and $V(G) \setminus V_x = \emptyset$.

Let $\mathrm{dist}(x, r) \geq 1$. Then x has a parent in T_G. Denote this parent by x'. By the induction hypothesis, $f(x')$ is equal to the number of vertices not in $G_{v'}$ that are complete to $V_{x'}$. Because V_x is contained in $V_{x'}$, these vertices are complete to V_x as well. Suppose that x is a $\oplus$-node. Then x' is a $\otimes$-node. This means that all vertices in $F(x')$ are complete to V_x. Hence, the total number of vertices in $V(G) \setminus V_x$ that are complete to V_x is equal to $f(x') + |F(x')| = f(x)$. Suppose that x is a $\otimes$-node. Then x' is a $\oplus$-node. This means that no vertex in $F(x')$ is adjacent to a vertex in V_x. Hence, the total number of vertices in $V(G) \setminus V_x$ that are complete to V_x is equal to $F(x') = f(x)$. $\square$

The following lemma follows directly from the construction of our algorithm.

Lemma 4. *Let $x \in V(T_G)$. Then $l(x) \neq \bot$ if and only if $V(G_x) \cap B \neq \emptyset$.*

We are now ready to give a new definition that plays an important role in proving correctness of our algorithm. Let x be a node in T_G. An *x-pseudo-KO-selection* of G is a function $f : V_x \to V(G)$ with $f(v) \in N(v)$ for all $v \in V$.

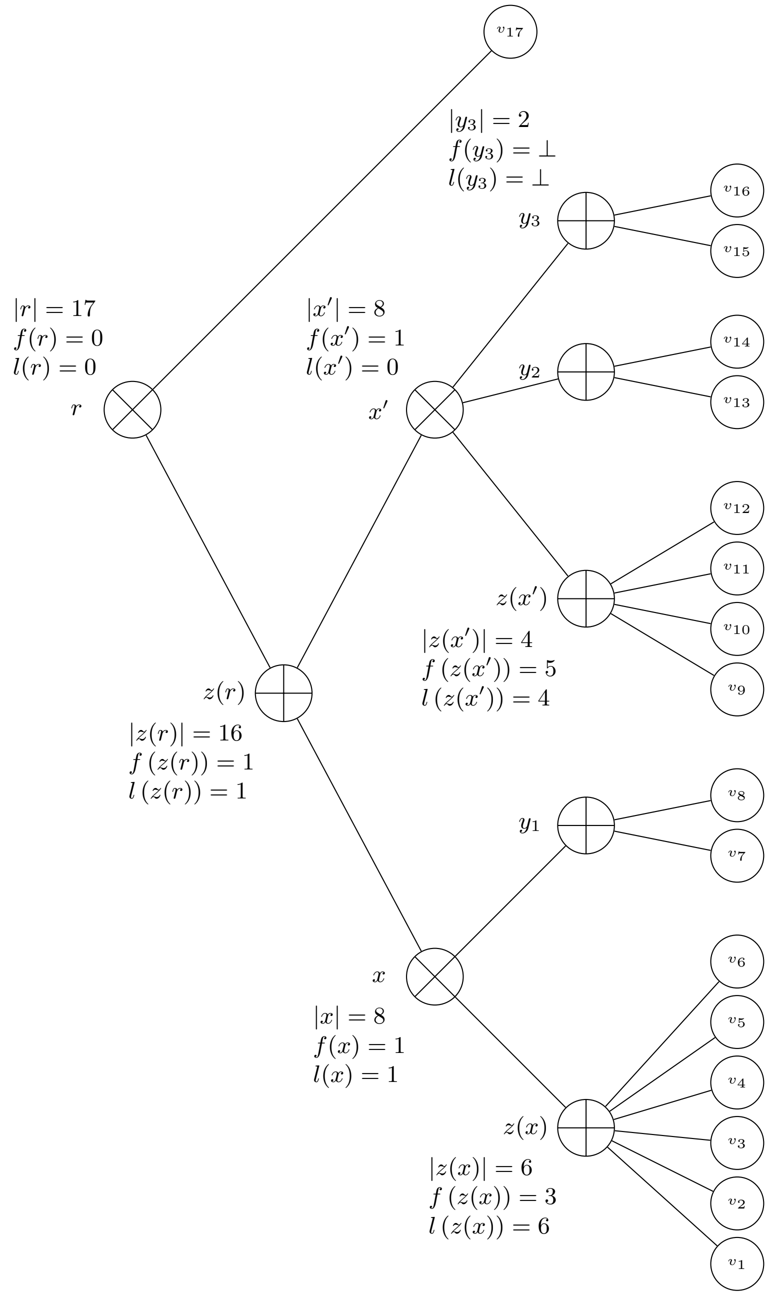

Fig. 2. The cotree T_G of the graph G of Figure 1 with f-values and l-values obtained from executing `Cograph-PKO`. Note that $B = \{v_1, v_2, v_3, v_4, v_5, v_6\} \cup \{v_9, v_{10}, v_{11}, v_{12}\}$.

We copy some terminology. If $f(v) = u$, we say that v fires at u, or that u is knocked-out by a firing of v. Note that every KO-selection of G is an x-pseudo-KO-selection of G. However, the reverse implication is not true if $x \neq r$, because we do not let any vertices in $V(G) \setminus V_x$ fire according to this definition.

For an x-pseudo-KO-selection, we define the *x-pseudo-KO-successor* of G_x as the subgraph of G induced by $V(G) \setminus f(V)$. We write $G \leadsto^x G'$ to denote that G' is an x-pseudo-KO-successor of G. We call a sequence

$$G \leadsto^x G^1 \leadsto^x \cdots \leadsto^x G^s$$

an *x-pseudo-KO-scheme* (where each single step is called a round) of G if in addition there is no vertex of V_x that fires at a vertex of V_x in some round i and at a vertex in $V(G) \setminus V_x$ in some round $j > i$. Let G_x^s be the subgraph of G^s induced by V_x. Then we say that G_x is *x-pseudo-reducible* to G_x^s. Define $\mathrm{pseudo}(x)$ as the number of vertices in a smallest graph to which G_x is x-pseudo-reducible and say that a corresponding x-pseudo-KO-scheme of G is *optimal*.

Lemma 5. *The cograph G is KO-reducible if and only if* $\mathrm{pseudo}(r) = 0$.

Proof. Recall that $V_r = V(G)$. Then the statement of the lemma holds because every KO-reduction scheme of G (if there exists one) is an r-pseudo-KO-scheme with $\mathrm{pseudo}(r) = 0$, and vice versa. $\square$

The following lemma is crucial for the correctness of our algorithm.

Lemma 6. *Let $x \in V(T_G)$ be a $\otimes$-node with $l(x) \neq \perp$. Then $l(x) = \mathrm{pseudo}(x)$.*

Proof. Let $x \in V(T_G)$ be a $\otimes$-node with $l(x) \neq \perp$. By Lemma 4, $V(G_x) \cap B \neq \emptyset$. We write $z = z(x)$. Let $|V_z| = q$ and $|F(x)| = p$. This enables us to write:

$$\begin{aligned} l(x) &= \max\{0, l(z) - f(x) \cdot |F(x)| - \tfrac{1}{2}|F(x)|(|F(x)| + 1)\} \\ &= \max\{0, l(z) - f(x) \cdot p - \tfrac{1}{2}p(p + 1)\}. \end{aligned}$$

Note that $q \geq 1$ and $p \geq 1$ by the definition of a $\otimes$-node.

Let d denote the number of internal $\otimes$-nodes on the longest path from x to a leaf in the subtree of T_G rooted at x. We prove the lemma by induction on d. Let $d = 0$. Then every child of x is either a leaf itself or all its children are leaves.

First suppose z is a leaf. Because $V(G_x) \cap B \neq \emptyset$, we find that $z \in B$. Hence, $l(z) = 1$. Then, as $p \geq 1$, we find that $l(z) - f(x) \cdot p - \frac{1}{2}p(p + 1) \leq 0$. Hence, $l(x) = 0$. Note that $q = 1$. Because z is a largest child of x, all children of x are leaves. Hence, G_x is a complete graph on $p + 1$ vertices. This means that G_x is KO-reducible. We conclude that $\mathrm{pseudo}(x) = 0 = l(x)$.

Now suppose that z is not a leaf. Then z has at least two children (which are all leaves since $d = 0$). Hence, $q \geq 2$. Because $V(G_x) \cap B \neq \emptyset$, every child of z is in B, that is, $V_z = B$ is an independent set, in particular, $q = |B|$. Because $l(\ell) = 1$ for every $\ell \in B$, this means that $l(z) = |B| = q$. We distinguish three cases.

Case 1. $q < p$.
Then $l(z) - f(x) \cdot p - \frac{1}{2}p(p+1) = q - f(x) \cdot p - \frac{1}{2}p(p+1) \leq 0$. Hence, $l(x) = 0$.

Let $y_1, \ldots, y_r, z$ be the children of x for some $r \geq 1$. In fact, because $2 \leq q < p$ and z is the largest child of x, we find that $r \geq 2$. Assume that $|V_{y_1}| \geq \cdots \geq |V_{y_r}|$. By definition, $q = |V_z| \geq |V_{y_1}|$. Because $q < p$, we can pick a set D of $q - |V_{y_1}| \geq 0$ vertices of $V(F(x)) \setminus V_{y_1}$. We define $T_1 = V_{y_1} \cup D$ and $T_i = V_{y_i} \setminus D$ for $i = 2, \ldots, r$. Note that $|T_1| = q$.

Let $\{|T_1|, \ldots, |T_r|\} = \{j_1, \ldots, j_s\}$ for some $s \leq r$, where $j_1 \geq \cdots \geq j_s$. Because $|T_1| = q$, we find that $j_1 = q$. We partition $V(G_x)$ into s subsets. For the first subset we pick j_s vertices from V_z and also j_s vertices from each non-empty T_i. The graph induced by the union of all these vertices has a hamilton cycle, as $q < p$, so besides T_1 at least one other set T_i is nonempty. We remove all chosen vertices. Then, for the second subset of our partition, we pick $j_{s-1} - j_s$ vertices from V_z and also $j_{s-1} - j_s$ vertices from each T_i that is not yet empty. The graph induced by the union of all chosen vertices has a hamilton cycle if there are two non-empty sets T_i and a perfect matching otherwise. We repeat this procedure until all sets T_i are empty. In this way we have found a $[1, 2]$-factor of G_x. Consequently, $\text{pko}(G_x) = 0$. Hence, $\text{pseudo}(x) = 0 = l(x)$.

Case 2. $q \geq p$ and $l(x) = 0$.
As $l(z) = q$, the assumption that $l(x) = 0$ implies that $q - f(x) \cdot p \leq \frac{1}{2}p(p+1)$. By Lemma 3, all vertices in $V(G) \setminus V_x$ that are adjacent to V_x are complete to V_x and moreover, the number of such vertices is equal to $f(x)$. This enables us to define the following x-pseudo-KO-scheme. Let all vertices of $F(x)$ fire at different vertices in V_z for the first $f(x)$ rounds. Let all vertices in V_z fire at the same vertex of $V(G) \setminus V_x$ for the first $f(x)$ rounds. Note that q decreases in this way. However, we may not need to perform all these rounds: after each round we check whether $p \leq q \leq \frac{1}{2}p(p+1)$. Because $q - f(x) \cdot p \leq \frac{1}{2}p(p+1)$, it will eventually happen that $q \leq \frac{1}{2}p(p+1)$. If it turns out that $q < p$, we slightly adjust the previous round by letting a sufficient number of vertices of $F(x)$ fire at the same vertex in V_z instead of at different vertices, in order to get $p \leq q \leq \frac{1}{2}p(p+1)$. We then apply Lemma 1 to knock out the remaining vertices of V_x in at most p additional rounds. Hence $\text{pseudo}(x) = 0 = l(x)$.

Case 3. $q \geq p$ and $l(x) > 0$.
As $l(z) = q$, the assumption that $l(x) > 0$ implies that $q > f(x) \cdot p + \frac{1}{2}p(p+1)$. Recall that, by Lemma 3, all vertices in $V(G) \setminus V_x$ that are adjacent to V_x are complete to V_x, and moreover, the number of such vertices is equal to $f(x)$. This enables us to define the following x-pseudo-KO-scheme. Let all vertices of $F(x)$ fire at different vertices in V_z for the first $f(x)$ rounds. Let all vertices in V_z fire at the same vertex of $V(G) \setminus V_x$ for the first $f(x)$ rounds. Afterwards we can reduce the number of vertices of V_x by at most $\frac{1}{2}p(p+1)$ by letting all vertices of $F(x)$ fire at different vertices in V_z, whereas all vertices in V_z fire at the same vertex of $F(x)$ until $F(x) = \emptyset$. Because $F(x) = \emptyset$ in the end, and in each round we have reduced the maximum number of vertices of the independent set V_z, we find that $\text{pseudo}(x) = q - f(x) \cdot p - \frac{1}{2}p(p+1) = l(z) - f(x) \cdot p - \frac{1}{2}p(p+1) = l(x)$.

Let $d \geq 1$. Then z is not a leaf as otherwise all children of x are leaves, which contradicts $d \geq 1$. Consequently, z is a $\oplus$-node. We distinguish two cases.

Case 1. $q < p$.
Observe that $l(z) \leq q$. Then $l(z) - f(x) \cdot p - \frac{1}{2}p(p+1) \leq q - f(x) \cdot p - \frac{1}{2}p(p+1) \leq 0$. Hence, $l(x) = 0$. We repeat the same arguments as for the corresponding case for $d = 0$ to obtain that $\text{pseudo}(x) = 0 = l(x)$. So Case 1 is proven.

Before we consider Case 2, we first analyze the subtree of T_G rooted at x. Let $s_1, \ldots, s_p$ be the children of z with $l(s_i) > 0$ for $i = 1, \ldots, p$ (if such children exist) and let $t_1, \ldots, t_q$ be the children of z with $l(t_i) = 0$ for $i = 1, \ldots, q$ (if such children exist). Note that all children of z are either leaves or $\otimes$-nodes. Let z' be a child of z. If z' is a leaf, then $\text{pseudo}(z') = 1 = l(z')$. If z' is a $\otimes$-node, we may apply the induction hypothesis to find that $\text{pseudo}(z') = l(z')$. In other words, $\text{pseudo}(s_i) = l(s_i)$ for $i = 1, \ldots, p$ and $\text{pseudo}(t_i) = l(t_i)$ for $i = 1, \ldots, q$. Then, because $G_z = G_{s_1} + \cdots + G_{s_p} + G_{t_1} + \cdots + G_{t_q}$, we find that an optimal z-pseudo-KO-scheme mimics the optimal s_i-pseudo-KO-schemes and optimal t_j-pseudo-KO-schemes (we may assume without loss of generality that all external firings outside G_z in a round are always at a single vertex). Hence, $\text{pseudo}(z) = l(s_1) + \cdots + l(s_p) + l(t_1) + \cdots + l(t_q) = l(z)$.

Case 2. $q \geq p$.
We define the following x-pseudo-KO-selection scheme. The firing rounds for the vertices in G_z are according to an optimal z-pseudo-KO-scheme under the following conditions. For the first $f(x)$ rounds any external firings outside G_z are at a single vertex, which is not in G_x. Note that this is possible by Lemma 3. Afterwards any external firing outside G_z must be in $F(x)$ and also for such firings we require that they are at a single vertex in every round. The vertices in $F(x)$ fire in each round at different vertices of G_x that are in $G_{s_1} + \cdots + G_{s_p}$ and that are not being fired at by vertices in G_x. They stop firing in a graph G_{s_i} as soon as they have knocked out $l(s_i)$ of its vertices. Note that we are guaranteed a budget of exactly $f(x) \cdot p + \frac{1}{2}p(p+1)$ firings from vertices outside G_z into G_z.

First suppose that $l(z) - f(x) \cdot p \leq \frac{1}{2}p(p+1)$, so $l(x) = 0$. Then we can knock out all $l(z)$ vertices of G_z that cannot be knocked out by internal firings inside G_z. As we still need to knock out the vertices of $F(x)$, we check after each round whether q has decreased such that $p \leq q \leq \frac{1}{2}p(p+1)$ holds. Because $q - f(x) \cdot p \leq \frac{1}{2}p(p+1)$, it will eventually happen that $q \leq \frac{1}{2}p(p+1)$. If it turns out that $q < p$, we slightly adjust the previous round as we did in Case 2 for $d = 0$, in order to get $p \leq q \leq \frac{1}{2}p(p+1)$. We then apply Lemma 1 to knock out the remaining vertices of V_x in at most p additional rounds. We conclude that $\text{pseudo}(x) = 0 = l(x)$.

Now suppose that $l(z) - f(x) \cdot p > \frac{1}{2}p(p+1)$, so $l(x) > 0$. Then, by the definition of our x-pseudo-KO-reduction scheme, all vertices in $F(x)$ have fired at different vertices in every round for $f(x) \cdot p + \frac{1}{2}p(p+1)$ rounds. Moreover, all vertices in $F(x)$ are knocked out afterwards. Because $\text{pseudo}(z) = l(z)$ and we mimicked an optimal z-pseudo-KO-scheme as regards the firings of the vertices of G_z in each round, we cannot improve. We conclude that pseudo$(x) = l(z) - f(x) \cdot p - \frac{1}{2}p(p+1) = l(x)$. This completes the proof of Lemma 6. ☐

Theorem 1. *The* Parallel Knock-Out *problem can be solved in $O(n+m)$ time on cographs with n vertices and m edges.*

Proof. Let G be a cograph with n vertices and m edges. If G is disconnected we consider each connected component of G separately. Hence, assume that G is connected.

We construct T_G. Run `Cograph-PKO` with input G. By Lemma 4, we find that $l(r) \neq \perp$. Hence, we may apply Lemma 6 to find that $l(r) =$ pseudo(r). By Lemma 5, we find that G is KO-reducible if and only if pseudo$(r) = 0$. As `Cograph-PKO` outputs a yes-answer if and only if $l(r) = 0$, we find it is correct.

It remains to show that `Cograph-PKO` runs in linear time. We can perform Step 1 in a bottom-up approach starting from the leaves of T_G. So, Steps 1-3 each visit each node at most once. This means that every node of x is visited at most three times in total. Because every co-tree has at most $n+n-1 = 2n-1$ vertices, we find that the running time of `Cograph-PKO` is $O(n)$. Because constructing T_G costs time $O(n+m)$ by Lemma 2, the total running time is $O(n+m)$. ☐

4 Split Graphs

We show the following result, the proof of which is (partially) based on the NP-hardness proof of 2-Parallel Knock-Out for bipartite graphs from [3].

Theorem 2. *The* Parallel Knock-Out *problem and, for any $k \geq 2$, the k-*Parallel Knock-Out *problem are* NP*-complete for split graphs.*

Proof. First consider the Parallel Knock-Out problem. We reduce from the Dominating Set problem, which is well known to be NP-complete (see [11]). This problem takes as input a graph $G = (V, E)$ and a positive integer p. We may assume without loss of generality that $p \leq |V|$. The question is whether G has a dominating set of cardinality at most p.

From an instance (G, p) of Dominating Set we construct a split graph G' as follows. Let $V(G) = \{v_1, \ldots, v_n\}$. We let $V(G')$ consist of three mutually disjoint sets: the set $V = \{v_1, \ldots, v_n\}$, a set $V' = \{v'_1, \ldots, v'_n\}$ and a set $W = \{w_1, \ldots, w_r\}$ where $r = \frac{1}{2}(n-p)(n-p+1)$. We define $E(G')$ as follows. First we add the edges $v_iv'_i$ for $i = 1, \ldots, n$. For all $i \neq j$, we add the edges $v_iv'_j$ and $v_jv'_i$ if and only if v_iv_j is an edge in $E(G)$. We also add an edge between every v_i and every w_j. Finally, we add an edge between any two vertices in V. Observe that G' is indeed a split graph in which V is a clique of size n and $V' \cup W$ is an independent set of size $n + r$. We claim that G has a dominating set of size at most p if and only if G' is KO-reducible.

First suppose G has a dominating set D of size at most p. Because $p \leq |V|$, we may assume without loss of generality that $D = \{v_1, \ldots, v_p\}$. We construct a KO-reduction scheme of G' as follows. In the first round let every vertex $v_i \in V$

fire at $v_i' \in V'$. For $i = 1, \ldots, p$, let v_i' fire at v_i. For $i = p+1, \ldots, n$ let v_i' fire at an arbitrary vertex in D, which is possible because D is a dominating set of G. Finally, let every vertex in W fire at an arbitrary vertex in D as well; this is possible by the construction of G'. The resulting (split) graph G'' consists of a clique $V \setminus D$ of size $n-p$ and the independent set W of size $\frac{1}{2}(n-p)(n-p)$. Because there is an edge between every vertex in V and every vertex in W, we find that G'' is KO-reducible by Lemma 1.

Now suppose G' is KO-reducible. Consider a KO-reduction scheme of G'. Let D be the subset of vertices that are knocked out in the first round. Because each vertex must fire at a neighbour, D is a dominating set of G. We claim that $|D| \leq p$. For contradiction, suppose that $|D| \geq p+1$. Let $V_1 = V \setminus D$ be the subset of V consisting of vertices not knocked out in the first round. Because $|D| \geq p+1$, we obtain $|V_1| = |V| - |D| \leq n-p-1$. Let V^* and W^* be the subsets of V' and W, respectively, that consist of vertices not knocked out in the first round. Vertices in $V' \cup W$ can only be knocked out by vertices of V. Moreover, the total number of vertices that V can knock out in the first round is at most $|V| = n$. This means that $V^* \cup W^*$ is an independent set of size

$$|V^* \cup W^*| = |V^*| + |W^*| \geq |V'| + |W| - n = \tfrac{1}{2}(n-p)(n-p+1).$$

However, as in every round the size of V_1 is reduced by at least 1, the maximum number of vertices in $V^* \cup W^*$ that V_1 can knock out is at most

$$(n-p-1) + (n-p-2) + \cdots + 1 < \frac{1}{2}(n-p)(n-p+1).$$

Hence, the scheme is not a KO-reduction scheme of G'. This is a contradiction, and we have completed the proof for Parallel Knock-Out.

Now let $k \geq 2$ and consider the k-Parallel Knock-Out problem. We use the same reduction and the same arguments as for Parallel Knock-Out after changing the size of W into $r := (n-p) + (n-p-1) + \cdots + (n-p-k+2)$. □

5 Conclusions

We have shown in Theorem 1 that Parallel Knock-Out is linear-time solvable for P_4-free graphs. We have also shown in Theorem 2 that Parallel Knock-Out and, for any $k \geq 2$, k-Parallel Knock-Out are NP-complete for split graphs. Because split graphs are $(2K_2, C_4, C_5)$-free [10], they are P_5-free. Hence, Theorems 1 and 2 have the following consequence.

Corollary 1. *The* Parallel Knock-Out *problem restricted to P_r-free graphs is linear-time solvable if $r \leq 4$ and* NP*-complete if $r \geq 5$.*

Whether it is possible to compute $\text{pko}(G)$ in polynomial time for cographs is still an open problem. It is natural to ask whether this can be solved by adjusting our algorithm `Cograph-PKO`. We pursued this approach, but encountered the following problem. With unlimited rounds, either all the vertices in $F(x)$ are used up or $G_{z(x)}$ can be reduced entirely so an adjustment can be made such that $F(x)$ and $G_{z(x)}$ knock each other out in the final round. With a restriction on the number of rounds, the assumption that $F(x)$ will always be entirely eliminated is no longer valid, since there may be survivors in both $F(x)$ and $G_{z(x)}$ and determining their optimal firing is not trivial.

We believe, however, that the above open problem is not the most interesting direction for future work; recall that our long-standing goal is to determine the complexity of Parallel Knock-Out on graph classes of bounded clique-width.

Also recall that cographs are exactly those graphs that have clique-width at most 2 [7]. Can we solve Parallel Knock-Out in polynomial time for graphs of clique-width at most 3? For this we could start by considering the class of distance-hereditary graphs, which have clique-width at most 3 [12]. Distance-hereditary graphs are completely decomposable with respect to a so-called split decomposition [13], a graph decomposition introduced by Cunningham and Edmonds [8] which may be useful for our purposes.

We also do not know whether there is a constant c such that Parallel Knock-Out is NP-complete for graphs of clique-width at most c. However, it is known that the related NP-complete problem Hamilton Cycle, which tests whether a graph has a hamiltonian cycle, is polynomial-time solvable on any graph class whose clique-width is bounded by a constant (this follows from combining results of [14, 18], also see [9]).

A different direction from above for extending our results would be to classify the complexity of Parallel Knock-Out restricted to H-free graphs. The complexity status is open even for small graphs $H \in \{4P_1, 2P_1 + 2P_2, P_1 + P_3, K_{1,4}\}$.

References

1. A. Brandstädt, V.B. Le and J. Spinrad, Graph Classes: A Survey, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics (SIAM), 1999.
2. H.J. Broersma, F.V. Fomin, R. Královič, and G.J. Woeginger, Eliminating graphs by means of parallel knock-out schemes, Discrete Applied Mathematics 155 (2007) 92–102.
3. H.J. Broersma, M. Johnson, D. Paulusma, and I.A. Stewart, The computational complexity of the parallel knock-out problem, Theoretical Computer Science 393 (2008) 182–195.
4. H.J. Broersma, M. Johnson, and D. Paulusma, Upper bounds and algorithms for parallel knock-out numbers, Theoretical Computer Science 410 (2008) 1329–1327.
5. D.G. Corneil, H. Lerchs and L. Stewart Burlingham, Complement reducible graphs, Discrete Applied Mathematics 3 (1981) 163–174.
6. D.G. Corneil, Y. Perl and L.K. Stewart, A linear recognition algorithm for cographs, SIAM Journal on Computing 14 (1985) 926–934.
7. B. Courcelle and S. Olariu, Upper bounds to the clique width of graphs, Discrete Applied Mathematics 101 (2000) 77–144.
8. W.H. Cunnigham and J. Edmonds, A combinatorial decomposition theory, Canadian Journal of Mathematics 32 (1980) 734–765.

9. W. Espelage, F. Gurski and E. Wanke, How to solve NP-hard graph problems on clique-width bounded Graphs in polynomial time, Proc. WG 2001, Lecture Notes in Computer Science 2204, 117–128
10. S. Földes and P.L. Hammer, Split graphs, 8th South–Eastern Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium 19 (1977) 311–315.
11. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.
12. M.C. Golumbic and U. Rotics, On the clique-width of some perfect graph classes, International Journal of Foundations of Computer Science 11 (2000) 423–443.
13. P. Hammer and F. Maffray, Completely separable graphs, Discrete Applied Mathematics 27 (1990) 85–99.
14. O. Johansson, Clique-decomposition, NLC-decomposition, and modular decomposition – relationships and results for random graphs, Congressus Numerantium 132 (1998) 39–60.
15. M. Johnson, D. Paulusma and A. Stewart, Knocking out P_k-free graphs, Proc. MFCS 2014, Lecture Notes in Computer Science 8635, 396–407.
16. M. Johnson, D. Paulusma and C. Wood, Path factors and parallel knock-out schemes of almost claw-free graphs, Discrete Mathematics 310 (2010) 1413–1423.
17. D.E. Lampert and P.J. Slater, Parallel knockouts in the complete graph, American Mathematical Monthly 105 (1998) 556–558.
18. E. Wanke, k-NLC graphs and polynomial algorithms, Discrete Applied Mathematics 54 (1994) 251–266.