Reduction Algorithms for Graphs with Small Treewidth*

Hans L. Bodlaender and Babette de Fluiter
Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
e-mail: {hansb,babette}@cs.ruu.nl

Abstract

This paper presents some new ideas and results on graph reduction applied to graphs with bounded treewidth. Arnborg et al. [2] have shown that many decision problems on graphs can be solved in linear time on graphs with bounded treewidth, by using a finite set of reduction rules. We show that this method can also be used to solve the construction variants of many of these problems, and to solve a number of optimization problems, and to solve construction variants of many of these optimization problems. For example, the construction variants of decision problems that are definable in monadic second order logic can be solved in this way. Examples of optimization problems that can be solved in this way are INDEPENDENT SET, INDUCED BOUNDED DEGREE SUBGRAPH, PARTITION INTO CLIQUES and HAMILTONIAN COMPLETION NUMBER.

We also show that the results of [6] can be applied to these reduction algorithms, which results in parallel algorithms that use $O(n)$ operations and $O(\log n \log^* n)$ time on an EREW PRAM, or $O(\log n)$ time on a CRCW PRAM (where n is the number of vertices of the graph).

1 Introduction

In this paper, new ideas and results are presented on graph reduction, applied to graphs with bounded treewidth. We consider reduction rules, where a subgraph of a graph G is to be replaced by another smaller subgraph (under some additional rules, see Section 2 for the precise definitions).

A graph property P is a function which assigns to each graph the value true or false. Arnborg et al. [2] have shown that for each graph property P which is of 'finite index', and each constant k, there exists a finite, complete, safe, and decreasing set of reduction rules for graphs with treewidth at most k. This set of reduction rules can be used to reduce a graph G by a series of applications of reduction rules from the set to a graph from some finite set of 'small' graphs if

*This research was partially supported by the Foundation for Computer Science (S.I.O.N) of the Netherlands Organization for Scientific Research (N.W.O.) and partially by the ESPRIT Basic Research Actions of the EC under contract 7141 (project ALCOM II). Some results of this paper have appeared in [5].

and only if $P(G)$ holds and the treewidth of G is at most k. The set of finite index graph properties includes many interesting properties, including all graph properties expressible in monadic second order logic.

Arnborg et al. use this result to show the existence of linear time algorithms that decide whether property P holds for a given graph G with bounded treewidth, without the need of using a tree decomposition of G. It should be noted that these algorithms use more than linear memory. The algorithm does not depend on the structure of the reduction rules: it can be applied for all sets of reduction rules that are finite, safe, complete and decreasing.

Bodlaender and Hagerup [6] give parallel algorithms based on finite, safe, complete and decreasing sets of reduction rules. These algorithms use $O(n)$ operations, $O(n)$ memory and $O(\log n \log^* n)$ time on the EREW PRAM, or $O(\log n)$ time on the CRCW PRAM (n is the number of vertices of the input graph). However, their algorithm only works if the reduction rules have some predefined structure. They show that for each graph property P which is of finite index, and each constant k, there exists a finite, complete, safe, and decreasing set of reduction rules for graphs with treewidth at most k, such that each reduction rule in this set has the desired structure. The algorithm of Bodlaender and Hagerup [6] can be simulated on one processor to get a sequential linear time algorithm which uses linear memory.

In this paper, we extend the results of Arnborg et al. [2] in two ways.

- We discuss a method to solve in many cases not only decision problems (i.e. properties) on graphs with bounded treewidth, but also the construction variants of these problems. We show that this method works for all construction variants of properties that are expressible in monadic second order logic.

- We show that a variant of the method of Arnborg et al. can be used to solve several optimization problems on graphs with bounded treewidth, and we give a way to prove for a given optimization problem that this method works.

We also show that a combination of these two results is possible: the construction variants of several graph optimization problems of graphs can be solved using graph reduction algorithms on graphs with bounded treewidth.

Furthermore, we show that the parallel algorithm of Bodlaender and Hagerup [6] can be applied to our results.

This paper is organized as follows. In Section 2, some definitions and preliminary results are given. In Section 3, a method to use reduction algorithms for the construction variants of decision problems is discussed. In Section 4 it is shown that graph reduction can also used to solve certain optimization problems, and in Section 5, the results of Sections 3 and 4 are combined. Section 6 discusses parallel algorithms, based on reduction, and Section 7 concludes the paper, and gives some ideas for further research.

2 Preliminaries

In this paper, the graphs we consider are undirected, do not contain self-loops or multiple edges. (Similar results can be derived for directed graphs. For simplicity, we concentrate on undirected graphs.)

We say a property is *effectively decidable* if an algorithm is *known* that decides on the property. Similarly, we say a function is *effectively computable* if an algorithm is known that computes the function value for a given element of the domain.

The notion of treewidth was introduced by Robertson and Seymour [11].

Definition 2.1. *A* tree decomposition TD *of a graph* $G = (V, E)$ *is a pair* $(\{X_i \mid i \in I\}, T)$ *with* $T = (I, F)$ *a tree, and* $\{X_i \mid i \in I\}$ *a family of subsets of* V*, one for each node of* T*, such that*

- $\bigcup_{i \in I} X_i = V$*,*
- *for all edges* $\{v, w\} \in E$*, there exists an* $i \in I$ *with* $v \in X_i$ *and* $w \in X_i$*, and*
- *for all* $i, j, k \in I$*: if* j *is on the path from* i *to* k *in* T*, then* $X_i \cap X_k \subseteq X_j$*.*

The treewidth *of a tree decomposition* $(\{X_i \mid i \in I\}, (I, F))$ *is* $\max_{i \in I} |X_i| \Leftrightarrow 1$*. The treewidth of a graph* G*, denoted by* $\mathrm{tw}(G)$*, is the minimum treewidth over all possible tree decompositions of* G*.*

Definition 2.2. *A* terminal graph *G is a triple (V, E, X) with (V, E) an undirected graph, and $X \subseteq V$ is an ordered subset of the vertices, denoted by $\langle x_1, \ldots, x_l \rangle$, $l \geq 0$, called the set of* terminals*. Vertices in $V \Leftrightarrow X$ are called* inner vertices*. A terminal graph (V, E, X) is called an l-terminal graph if $|X| = l$. A terminal graph (V, E, X) is said to be* open*, if there are no edges between terminals (for all $v, w \in X$, $\{v, w\} \notin E$).*

The usual undirected graphs (i.e., without terminals) will be simply called *graph*.

Definition 2.3. *The operation $\oplus$ maps two terminal graphs G and H with the same number l of terminals to a graph $G \oplus H$, by taking the disjoint union of G and H, then identifying the corresponding terminals, i.e., for $i = 1, \ldots, l$, the ith terminal of G is identified with the ith terminal of H, and then removing multiple edges.*

For an example of the $\oplus$-operation, see Figure 1.

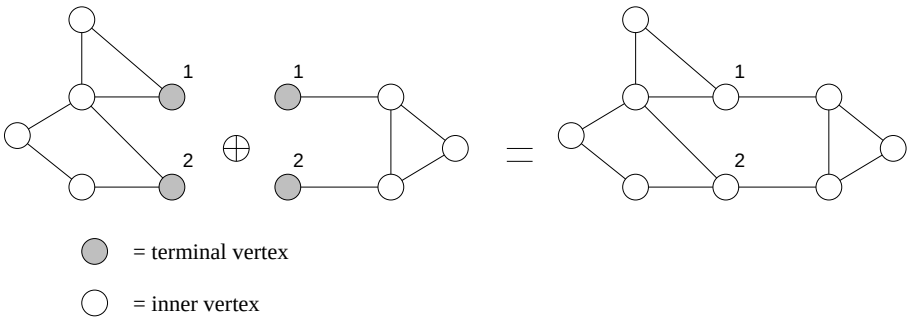

Figure 1: Example of $\oplus$-operation

Two terminal graphs $(V_1, E_1, \langle x_1, \cdots, x_k \rangle)$ and $(V_2, E_2, \langle y_1, \cdots, y_l \rangle)$ are said to be *isomorphic*, if $k = l$ and there exists a bijective function $f : V_1 \to V_2$ with for all $v, w \in V_1$, $\{v, w\} \in E_1 \Leftrightarrow \{f(v), f(w)\} \in E_2$ and for all i, $1 \leq i \leq k$, $f(x_i) = y_i$. The main difference with the usual definition of graph isomorphism is that we require that the corresponding terminals are mapped to each other.

Definition 2.4. *A* reduction rule r *is an ordered pair* (H_1, H_2)*, where* H_1 *and* H_2 *are* l*-terminal graphs for some* $l \geq 0$*. An application of reduction rule* (H_1, H_2) *is the operation, that takes a graph* G *of the form* $G_1 \oplus G_3$*, with* G_1 *isomorphic to* H_1*, and replaces it by the graph* $G_2 \oplus G_3$*, with* G_2 *isomorphic to* H_2*. We write* $G \xrightarrow{r} G_2 \oplus G_3$*.*

An example of the application of a reduction rule is given in Figure 2.

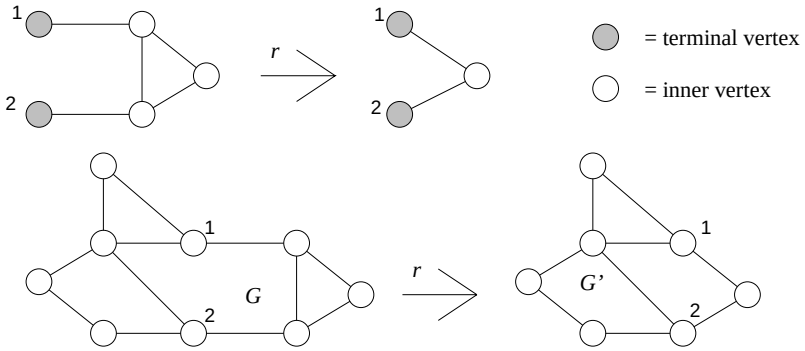

Figure 2: Applying rule r to G yields G'

For two graphs G and G', and a set of reduction rules $\mathcal{R}$, we write $G \xrightarrow{\mathcal{R}} G'$, if there exists an $r \in \mathcal{R}$ with $G \xrightarrow{r} G'$.

Definition 2.5. *Let* P *be a graph property, i.e. for each graph* G*,* $P(G) \in \{\text{true}, \text{false}\}$*. Let* $\mathcal{R}$ *be a set of reduction rules.*

- $\mathcal{R}$ *is* safe *for* P *if, whenever* $G \xrightarrow{\mathcal{R}} G'$*, then* $P(G) \Leftrightarrow P(G')$*.*
- $\mathcal{R}$ *is* complete *for* P *if the set of graphs* $\{G \mid P(G) \wedge \neg \exists G' : G \xrightarrow{\mathcal{R}} G'\}$ *is finite.*
- $\mathcal{R}$ *is* terminating *if there does not exist an infinite sequence* $G_1 \xrightarrow{\mathcal{R}} G_2 \xrightarrow{\mathcal{R}} G_3 \xrightarrow{\mathcal{R}} \cdots$*.*
- $\mathcal{R}$ *is* decreasing *if, whenever* $G \xrightarrow{\mathcal{R}} G'$*, then* G' *contains fewer vertices than* G*.*

Clearly, a decreasing set of rules is terminating.

A set of reduction rules $\mathcal{R}$, that is finite, safe, complete, and terminating for a property P corresponds to an algorithm that decides whether property P holds on a given graph: repeat applying rules from $\mathcal{R}$ starting with the input graph, until no rule from $\mathcal{R}$ can be applied anymore. If the resulting graph belongs to the finite set $\{G \mid P(G) \wedge \neg \exists G' : G \xrightarrow{\mathcal{R}} G'\}$, then P holds on the input graph, otherwise it does not. In [2] it has been shown how, when the set is decreasing, this algorithm can be implemented such that it takes linear time and polynomial space.

Definition 2.6. *For a graph property P the equivalence relation $\sim_{P,l}$ on l-terminal graphs, is defined as follows.*

$$G_1 \sim_{P,l} G_2 \Leftrightarrow (\forall_{l\text{-terminal graphs } H}\ P(G_1 \oplus H) \Leftrightarrow P(G_2 \oplus H))$$

Property P is of finite index, *if for all $l \geq 1$, $\sim_{P,l}$ has finitely many equivalence classes.*

It appears that many important graph properties are of finite index. For instance, all properties that can be formulated in monadic second order logic, i.e. that are *MS-definable*, are of finite index (for a definition, see e.g. [3]). These include HAMILTONIAN CIRCUIT, k-COLORABILITY (for fixed k), and many others.

An equivalence relation $\sim'$ is a *refinement* of an equivalence relation $\sim$ if each equivalence class of $\sim'$ is a subset of an equivalence class of $\sim$.

For each integer $k \geq 1$, let TW_k be the graph property defined as follows: for each graph G, $TW_k(G) = \mathrm{tw}(G) \leq k$. For a property P and an integer k, the property P_k is defined as $P_k(G) = P(G) \wedge TW_k(G)$.

Lemma 2.1. *If P is of finite index, then P_k is of finite index for each $k \geq 1$.*

Proof. The property TW_k is of finite index, for each $k \geq 1$, since it is MS-definable (see e.g. [3]). For each l, let $\sim_l$ be the equivalence relation on l-terminal graphs which is defined as follows.

$$G_1 \sim_l G_2 \Leftrightarrow G_1 \sim_{P,l} G_2 \wedge G_1 \sim_{TW_k,l} G_2$$

If $G_1 \sim_l G_2$, then $G_1 \sim_{P_k,l} G_2$, so $\sim_l$ is a refinement of $\sim_{P_k,l}$. Hence the number of equivalence classes of $\sim_{P_k,l}$ is at most the number of equivalence classes of $\sim_l$, which is at most the number of equivalence classes of $\sim_{P,l}$ times the number of equivalence classes of $\sim_{TW_k,l}$. Hence P_k is of finite index. □

(A similar lemma holds, if we pose an additional constant upper bound on the maximum degree of vertices in the graph.)

Finite index corresponds to 'finite state': there exists a linear time algorithm that decides the property on graphs, given with a tree decomposition of bounded treewidth. Moreover, this algorithm is of a special, well described structure. See e.g. [1].

Note that a reduction rule $(H_1, H_2) \in \mathcal{R}$ is safe for a property P if and only if $H_1 \sim_{P,l} H_2$ (if H_1 and H_2 are l-terminal graphs).

Below, we give a lemma on the existence of subgraphs of a certain size and type in graphs with bounded treewidth. This lemma will be used to show that there is a finite, safe, complete and decreasing set of reduction rules for a property P_k if property P is of finite index.

Lemma 2.2. *Let k and r be positive integers. If $G = (V, E)$ is a graph with n vertices and treewidth at most k, and $n \geq r+1$, then G can be written as $G_1 \oplus G_2$, with G_1 and G_2 terminal graphs with at most $k+1$ terminals, and G_1 has at least $r+1$ and at most $2(r+1)(k+1)$ vertices, and, if G_1 has one or more terminals, then G_1 is open and connected.*

Proof. Let $H_1, \ldots, H_l$ be the connected components of G. If, for each i, $|V(H_i)| \leq r$, then there is a set $I' \subseteq \{1, \ldots, l\}$, such that $r+1 \leq \sum_{i \in I'} |V(H_i)| \leq 2(r+1)$. In this case, let G_1 denote the zero-terminal graph consisting of all components H_i with $i \in I'$, and let G_2 denote the zero-terminal graph consisting of all components H_i with $i \notin I'$.

Suppose that there is an i, $1 \leq i \leq l$, such that $|V(H_i)| \geq r+1$. We show that the lemma holds for H_i. We can then extend G_2 with the other components of G to prove the lemma. From now on, let G denote H_i.

Let $TD = (\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of width at most k of G. Take $s \in I$ arbitrarily, and let X_s be the root node of the tree decomposition. For each node X_i, let V_i denote the set of all vertices of G which occur in X_i or in a node X_j, where j is a descendant of i in T, and let G'_i be the graph obtained from $G[V_i]$ by deleting al edges between vertices in X_i. Let TD_i denote the tree decomposition obtained from TD by taking the subtree rooted at X_i. Note that TD_i is a tree decomposition of $G[V_i]$. Note furthermore that for each $v \in V_i$, there is a path in $G[V_i]$, and also in G'_i, to some $w \in X_i$, since G is connected. Hence G'_i has at most $k+1$ connected components, and each component contains at least one vertex of X_i.

If G has less then $(r+1)(k+1)$ vertices, then let G_1 be the zero-terminal graph G, and let G_2 be the empty zero-terminal graph. If G has at least $(r+1)(k+1)$ vertices, then apply the following algorithm to find G_1 (the definition of G_2 then follows directly).

Algorithm Find

1. $(* \; |V(G)| \geq (r+1)(k+1)$ and there is a path from each $v \in V_s$ to a $w \in X_s \; *)$
2. **if** $|V(G)| \leq 2(r+1)(k+1)$
3. **then** Let $H_1, \ldots, H_l$ be the components of G'_s. $(* \; l \leq k+1 \; *)$
4. Let H_j be a component with at least $r+1$ vertices.
5. Let G_1 be the terminal graph obtained from H_j by taking as terminals the vertices in $V(H_j) \cap X_s$, and leaving out all edges between terminals.
6. **return** G_1
7. **else** $(* \; |V| > 2(r+1)(k+1) \; *)$
8. Let j be an arbitrary child of s in T.
9. **if** $|V_j| \geq (r+1)(k+1)$
10. **then** Let G denote the graph $G[V_j]$.
11. Let TD be TD_j.
12. Let $s = j$.
13. Go to step 1
14. **else** $(* \; |V_j| < (r+1)(k+1) \; *)$
15. Let $V'_j = \bigcup \{V_i \mid i$ is child of s in T and $i \neq j \; \}$.
16. $(* \; |V'_j| \geq |V| \Leftrightarrow |V_j| \geq (r+1)(k+1) \; *)$
17. Let G denote the graph $G[V'_j]$.
18. Let TD be the tree decomposition of G obtained from the old TD by leaving out the subtree rooted at X_j
19. Go to step 1

Note that a terminal graph G_1 that is returned has at most $k+1$ terminals, is open and connected, and has at least $r+1$ and at most $2(r+1)(k+1)$ vertices. □

The following theorem has been proved in a slightly different form in [2], but we give a proof which may be somewhat easier to follow.

Theorem 2.1. *Let $k \geq 1$ be a constant, P a graph property, and suppose P is of finite index. There exists a finite, safe, complete and decreasing set of reduction rules $\mathcal{R}$ for P_k. Moreover, for each reduction rule $(H, H') \in \mathcal{R}$, H and H' are open, and if H has one or more terminals, then H is connected.*

If there is also an equivalence relation $\sim_l$ for each $l \geq 1$, which is a refinement of $\sim_{P,l}$, is effectively decidable, and has a finite number of equivalence classes, then such a set of reduction rules can effectively be constructed.

Proof. Note that we only have to construct reduction rules (H, H'), for which there is a terminal graph G such that $P_k(H \oplus G)$ holds (and consequently, $P_k(H' \oplus G)$ holds).

For every $l \leq k+1$, and every equivalence class c of $\sim_{P_k,l}$, do the following. If $l = 0$ and c contains graphs with treewidth at most k, then take a representing graph H_c from c which has treewidth at most k. If $l \geq 1$ and c contains at least one open and connected l-terminal graph which has treewidth at most k, choose a representing open, connected l-terminal graph $H_c \in c$ with treewidth at most k. Let r be the maximum number of vertices of all chosen graphs H_c.

Let $\mathcal{R}$ denote the set of reduction rules to build. First, for all zero-terminal graphs H with at least $r+1$ and at most $2(r+1)(k+1)$ vertices, if we have a representative for the class c which contains H, then add reduction rule (H, H_c) to $\mathcal{R}$. Next, for all l, $1 \leq l \leq k+1$, for all open connected l-terminal graphs H with at least $r+1$ and at most $2(r+1)(k+1)$ vertices, if we have a representative for the equivalence class c in which H is contained, then add the reduction rule (H, H_c) to $\mathcal{R}$. Note that if we do not have such a representative, then H must have treewidth $k+1$ or more, and hence there is no terminal graph G for which $P_k(H \oplus G)$ holds.

It is easy to see that $\mathcal{R}$ is finite: there are finitely many l-terminal graphs with at most $2(r+1)(k+1)$ vertices. Safeness of the resulting set $\mathcal{R}$ follows directly from the fact that each left- and right-hand-side of a rule in $\mathcal{R}$ belong to the same equivalence class of the relation $\sim_{P_k,l}$. As, by Lemma 2.2, each graph with treewidth at most k and at least $r+1$ vertices, has an applicable rule from the set $\mathcal{R}$, completeness follows directly: the set $\{H \mid P(H) \wedge \neg\exists H' : H \xrightarrow{\mathcal{R}} H'\}$ contains only graphs with at most r vertices. It is obvious that $\mathcal{R}$ is decreasing.

We now show how we can effectively construct such a set of reduction rules. Note that the non-constructive parts in the proof until now are the part of finding a representative for each equivalence class which contains open terminal graphs with treewidth at most k, and the part of testing in which equivalence class a graph is contained. For each l, let $\sim_l$ be an effectively decidable equivalence relation on l-terminal graphs that is a refinement of $\sim_{P,l}$ and has a finite number of equivalence classes.

Arnborg et al. [2] give a way to construct, for a given integer m, a representative of each equivalence class of $\sim_l$ ($0 \leq l \leq m+1$) which contains a graph for which there exists a tree decomposition of width m with all terminals in the same node.

Furthermore, Lagergren and Arnborg [10] give an effectively decidable equivalence relation $\sim'_{TW_k,l}$, which has a finite number of equivalence classes for each k and l, and is a refinement of $\sim_{TW_k,l}$. This gives us enough ingredients to show how to construct reduction rules. First consider the construction of representatives.

For each l and k, let $\sim_{k,l}$ and $\sim'_{k,l}$ be equivalence relations on l-terminal graphs which are defined as follows.

$$\begin{aligned} G_1 \sim_{k,l} G_2 &\Leftrightarrow G_1 \sim_l G_2 \wedge G_1 \sim'_{TW_k,l} G_2 \\ G_1 \sim'_{k,l} G_2 &\Leftrightarrow G_1 \sim_{k,l} G_2 \wedge (G_1 \text{ is open } \Leftrightarrow G_2 \text{ is open}) \end{aligned}$$

It is clear that both $\sim_{k,l}$ and $\sim'_{k,l}$ are effectively decidable, have a finite number of equivalence classes, and are a refinement of $\sim_{P_k,l}$. Furthermore, $\sim'_{k,l}$ is a refinement of $\sim_{k,l}$.

Let G be an l-terminal graph with $l \leq k+1$, suppose G has treewidth at most k. There is a tree decomposition of width $2k+1$ of G in which all terminals are in one node: take an arbitrary tree decomposition of width k of G, append a node containing all terminals at an arbitrary place, and add all terminals to all other nodes.

Use the result from [2] to generate a representative for each equivalence class of $\sim'_{k,l}$ (for each $l \leq 2k+1$) which contains a graph for which there is a tree decomposition of width $2k+1$ with all terminals in one node. After the generation, throw away all representatives with more than $k+1$ terminals or with treewidth $k+1$ or more. The resulting set contains a representative for each equivalence class of $\sim_{k,l}$, $0 \leq l \leq k+1$, which contains a graph of treewidth at most k. Let R denote this set.

Now delete all graphs from R which are not open. The resulting set contains a representative for each equivalence class of $\sim_{k,l}$ which contains open l-terminal graphs of treewidth at most k, and hence this is the set we need.

Now it is easy to construct a finite, safe, complete and decreasing set of reduction rules. Let r again be the maximum number of vertices of any graph in R. For all $l \leq k+1$, for all open and, if $l \geq 1$, connected l-terminal graphs H with at least $r+1$ and at most $2(r+1)(k+1)$ vertices, find an $H' \in R$ for which $H \sim_{k,l} H'$ (using the algorithm for deciding $\sim_{k,l}$). If an H' is found, then add the reduction rule (H, H') to an initially empty set of reduction rules $\mathcal{R}$. □

The open and connectedness properties of the reduction rules in Theorem 2.1 are not needed for the algorithm of Arnborg et al [2], but they are used for the parallel algorithm of Bodlaender and Hagerup [6], see also Section 6. As each right-hand-side of a rule in $\mathcal{R}$ is open, applying a rule in $\mathcal{R}$ can never give multiple edges between a pair of vertices. The connectedness of the left-hand-sides of the reduction rules is used to obtain a more efficient way to find occurrences of left-hand-sides of reduction rules in a given graph.

From the proof of Theorem 2.1, we can also conclude the following.

Corollary 2.1. *Let P be a graph property, and for each $l \geq 0$, let $\sim_l$ be a refinement of $\sim_{P,l}$. Let $k \geq 1$. If $\sim_l$ has a finite number of equivalence classes for each $l \geq 0$, then there is a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules for P_k, such that for each $(H, H') \in \mathcal{R}$, $H \sim_l H'$. Moreover, if $\sim_l$ is effectively decidable, then such a set $\mathcal{R}$ can effectively be constructed.*

More background information about graph reduction and graphs of bounded treewidth can be found in [4, 8].

3 Constructing Solutions

Many graph properties are of the form $P(G) = \exists_{S \in D(G)} Q(G, S)$, where $D(G)$ is a *solution domain* (or shortly domain), which is some set depending on G, and Q is a property of G and S, i.e. $Q(G, S) \in \{\text{true}, \text{false}\}$ for all graphs G and all $S \in D(G)$. An $S \in D(G)$ for which $Q(G, S)$ holds is called a *solution* for G. For example, for the perfect matching problem on a graph G, $D(G)$ can be $\mathcal{P}(E)$, the power set of E, and for $S \in D(G)$, $Q(G, S)$ holds if and only if every vertex in G is end point of exactly one edge in S. Hence S is a solution for G if S is a perfect matching of G.

Often we are not only interested in whether $P(G)$ holds, but we are also interested in a solution $S \in D(G)$ for which $Q(G, S)$ holds (if $P(G)$ holds). However, such a solution is not constructed if reduction algorithms are used: these algorithms only compute whether $P(G)$ holds or not. For instance, 3-COLORABILITY is of finite index, so there is a finite, safe, complete and decreasing set of reduction rules for this property on graphs with bounded treewidth. However, by reducing a given graph with these rules, we do not find a three-coloring for it if one exists.

In this section we give an idea how to construct solutions in reduction algorithms, and we give a condition for graph properties such that this idea can be used. We also show that constructive versions of graph properties that are MS-definable satisfy this condition.

The idea is to solve the construction versions of problems as follows. First apply a reduction algorithm and store the applied reductions and the place at which they are applied. Then, if $P(G)$ holds, construct a solution for the reduced graph. After that, undo the reductions one by one in reversed order, and after each undo-action, reconstruct the solution for the old graph into a solution of the new graph.

To keep the total running time of the algorithm linear in the number of vertices of the graph, the total time for all reconstructions of solutions must be linear. This is possible if a solution for the reduced graph can be constructed in constant time, and, for each undo-action, a solution for the new graph can be computed from the old solution in constant time. This may for example be possible if the new solution only differs from the old solution in the part of the graph that was involved in the undone reduction. This gives rise to the following algorithm for a given set $\mathcal{R}$ of reduction rules for a graph property P with $P(G) = \exists_{S \in D(G)} Q(G, S)$.

Algorithm ConstructSolution
Input: A graph G
Output: An $S \in D(G)$ such that $Q(G, S)$ holds if $P(G)$ holds, false otherwise
1. $i \leftarrow 0$
2. **while** there is applicable reduction rule $r_i = (H_i, H_i') \in \mathcal{R}$
3. **do** apply r_i to G and store place of application of r_i
4. $i \leftarrow i + 1$
5. **if** $\neg P(G)$
6. **then return** false
7. **else** ($*$ construct initial solution $*$)
8. let $S \in D(G)$ be such that $Q(G, S)$
9. **while** $i > 0$

10. **do** ($*$ undo reduction r_{i-1} and reconstruct solution $*$)
11. $i \leftarrow i \Leftrightarrow 1$
12. undo reduction r_i, let G' denote new graph
13. let H be such that $G = H_i' \oplus H$ and $G' = H_i \oplus H$
14. construct $S' \in D(G')$ from S such that $Q(G', S')$ holds, and S' only differs from S in part H_i
15. $G \leftarrow G'$;$S \leftarrow S'$
16. **return** S

In analogy with $\sim_{P,l}$, we define $\sim_{Q,l}$. We show that if Q is of finite index, then there is a finite, safe, complete and decreasing set of reduction rules for P_k ($k \geq 1$), and furthermore, with this set of reduction rules, it is possible to use algorithm ConstructSolution for constructing solutions. What remains after that is that, to keep the algorithm running in linear time, the construction of a solution for the reduced graph must be done in constant time, and the construction of a new solution from an old solution in the reduced part of the graph must be done in constant time.

Before being able to define $\sim_{Q,l}$, we first give a number of other definitions. The first definition we need is a definition of $\oplus$ for solutions of two l-terminal graphs.

Let D be some solution domain and let G_1 and G_2 be l-terminal graphs. Let $S \in D(G_1 \oplus G_2)$. We have to define $S[G_1]$ in such a way that it only depends on G_1, i.e. it may not contain vertices or edges which are not in G_1. For most domains this works if $S[G_1]$ is obtained from S by deleting all edges which are not in $E(G_2)$, and all vertices which are not in $V(G_2)$ from S. ($[\,]$ should be seen as a function, mapping the pair (S, G) to $S[G]$.)

Let D be a solution domain, and let a definition of $[\,]$ be given. For each $l \geq 0$, each l-terminal graph G, define

$$D_{[]}(G) = \{S[G] \mid S \in D(G \oplus H) \text{ for some } l\text{-terminal graph } H\}.$$

$D_{[]}(G)$ is called the domain of *partial solutions* of G. Note that $D(G) \subseteq D_{[]}(G)$.

Definition 3.1 (Inducibility). *Let D be some domain. D is* inducible *if there is a function $[\,]$ for D, such that for each graph G and for each pair of terminal graphs G_1 and G_2 such that $G_1 \oplus G_2 = G$, each $S \in D(G)$, there is no $S' \in D(G)$, $S' \neq S$, such that $S'[G_1] = S[G_1]$ and $S'[G_2] = S[G_2]$.*

Definition 3.2 ($\oplus$-Compatibility). *Let G and H be l-terminal graphs for some $l \geq 0$, let D be an inducible domain, and let $S \in D_{[]}(G)$ and $S' \in D_{[]}(H)$. (G, S) and $(H.S')$ are* $\oplus$-compatible *if there is an $S'' \in D(G \oplus H)$ such that $S''[G] = S$ and $S''[H] = S'$. If (G, S) and (H, S') are $\oplus$-compatible, then we write $S \oplus S' = S''$.*

Note that $S \oplus S'$ is defined properly, since if there is an $S'' \in D(G \oplus H)$ such that $S = S''[G]$ and $S' = S''[H]$, then this S'' is unique, because D is inducible.

For example, if $D(G) = \mathcal{P}(V)$, then D is inducible with the common definition of $[\,]$, and $D_{[]}(G) = D(G)$ for all terminal graphs G. If $G = (V, E, \langle x_1, \ldots, x_l \rangle)$ and $H =$

$(V', E', \langle y_1, \ldots, y_l \rangle)$ are l-terminal graphs, and $S \in D_{[]}(G)$, $S' \in D_{[]}(H)$, then (G, S) and (H, S') are $\oplus$-compatible if and only if

$$\{i \mid 1 \leq i \leq l \wedge x_i \in S\} = \{i \mid 1 \leq i \leq l \wedge y_i \in S'\}.$$

In that case, $S \oplus S'$ is simply the union of S and S' in $G \oplus H$.

The following definition is necessary for the definition of $\sim_{Q,l}$.

Definition 3.3 (Compatibility). *Let D be an inducible domain, let G_1 and G_2 be l-terminal graphs for some $l \geq 0$, and let $S_1 \in D_{[]}(G_1)$ and $S_2 \in D_{[]}(G_2)$. (G_1, S_1) and (G_2, S_2) are* compatible *if for each l-terminal graph H, each $S \in D_{[]}(H)$, (G_1, S_1) is $\oplus$-compatible with (H, S) if and only if (G_2, S_2) is $\oplus$-compatible with (H, S).*

Note that compatibility is an equivalence relation. The set of all these equivalence classes is denoted by $C_{\mathrm{cmp},l}$, for each l, and the equivalence classes are also called compatibility classes. For two equivalence classes c and c' of some equivalence relation which is a refinement of compatibility, we say that c and c' are $\oplus$-compatible if, for each $(G, S) \in c$, $(H, S') \in c'$, (G, S) and (H, S') are $\oplus$-compatible.

Let P be a graph property, and suppose $P(G)$ can be written as $\exists_{S \in D(G)}\ Q(G, S)$, such that domain D is inducible.

Definition 3.4. *For each $l \geq 0$, $\sim_{Q,l}$ is an equivalence relation on pairs of l-terminal graphs G and partial solutions $S \in D_{[]}(G)$, which is defined as follows. Let G_1, G_2 be l-terminal graphs, and $S_1 \in D_{[]}(G_1)$ and $S_2 \in D_{[]}(G_2)$.*

$$\begin{aligned}(G_1, S_1) \sim_{Q,l} (G_2, S_2) \quad \Leftrightarrow \quad & (G_1, S_1) \text{ and } (G_2, S_2) \text{ are compatible and} \\ & \forall_{l\text{-terminal graphs } H}\ \forall_{S \in D_{[]}(H)} \\ & (H, S) \oplus\text{-compatible with } (G_1, S_1) \text{ and } (G_2, S_2) \\ & \Rightarrow Q(G_1 \oplus H, S_1 \oplus S) = Q(G_2 \oplus H, S_2 \oplus S)\end{aligned}$$

Let $C_{Q,l}$ denote the set of equivalence classes of $\sim_{Q,l}$, and for each l-terminal graph G and $S \in D_{[]}(G)$, let $ec_{Q,l}(G, S) = c$, $c \in C_{Q,l}$, if and only if $(G, S) \in c$.

By $\sim_{rQ,l}$, we usually denote an equivalence relation which is a refinement of $\sim_{Q,l}$. By $C_{rQ,l}$ we denote the set of equivalence classes of $\sim_{rQ,l}$, and for each l-terminal graph G, each $S \in D_{[]}(G)$, $ec_{rQ,l}(G, S) = c$ if (G, S) is in equivalence class $c \in C_{rQ,l}$.

Definition 3.5. *For each $l \geq 0$, and for each refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$, let $\approx_{rQ,l}$ be an equivalence relation on l-terminal graphs, which is defined as follows. For each two l-terminal graphs G_1 and G_2,*

$$\begin{aligned}G_1 \approx_{rQ,l} G_2 \quad \Leftrightarrow \quad & \{\, ec_{rQ,l}(G_1, S_1) \in C_{rQ,l} \mid S_1 \in D_{[]}(G_1) \,\} \\ & = \{\, ec_{rQ,l}(G_2, S_2) \in C_{rQ,l} \mid S_2 \in D_{[]}(G_2) \,\}\end{aligned}$$

Note that $\approx_{rQ,l}$ is an equivalence relation.

Lemma 3.1. *For each $l \geq 0$, each refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$, $\approx_{rQ,l}$ is a refinement of $\sim_{P,l}$.*

Proof. Let G_1 and G_2 be l-terminal graphs, Suppose $G_1 \approx_{rQ,l} G_2$. We have to prove that for all l-terminal graphs H, $P(G_1 \oplus H) = P(G_2 \oplus H)$. Suppose $P(G_1 \oplus H)$ holds. Let $S \in D(G_1 \oplus H)$ such that $Q(G_1 \oplus H, S)$ holds. Let $S_1 = S[G_1]$ and $S' = S[H]$. Since $G_1 \approx_{rQ,l} G_2$, there is an $S_2 \in D_{[]}(G_2)$ such that $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$, and hence $(G_1, S_1) \sim_{Q,l} (G_2, S_2)$. Since domain D is inducible, (G_1, S_1) and (H, S) are $\oplus$-compatible. Furthermore, (G_1, S_1) and (G_2, S_2) are compatible, so (G_2, S_2) and (H, S') are also $\oplus$-compatible. Hence $Q(G_1 \oplus H, S_1 \oplus S') = Q(G_2 \oplus H, S_2 \oplus S')$, so $P(G_2 \oplus H)$ holds. By symmetry, this means that $P(G_1 \oplus H) = P(G_2 \oplus H)$. □

Let G be a graph. Suppose we run Algorithm ConstructSolution on G with a set of finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules. Furthermore, suppose that in line 12 we undo a rule $r_i = (H_i, H_i')$ in G, and $H_i \approx_{Q,l} H_i'$. Let G' and H be defined as in the algorithm. Then there exists an $S' \in D(G')$, such that $Q(G', S')$ holds and S' and S do not differ in H (i.e., $S'[H] = S[H]$): let $S'' \in D_{[]}(H_i)$ such that $(H_i, S'') \sim_{Q,l} (H_i', S[H_i'])$, and let $S' = S'' \oplus S[H]$. This gives us an algorithmic method to quickly construct a solution for the unreduced graph, given a solution for the reduced graph.

So what we need is a finite, safe, complete and terminating set $\mathcal{R}$ of reduction rules, such that for each rule $(G_1, G_2) \in \mathcal{R}$, $G_1 \approx_{Q,l} G_2$. We now show when this is possible.

Lemma 3.2. *If for each $l \geq 0$, $|C_{rQ,l}|$ is finite, then $\approx_{rQ,l}$ has a finite number of equivalence classes.*

Proof. For each $l \geq 0$, the number of equivalence classes of $\approx_{rQ,l}$ is at most $2^{|C_{rQ,l}|}$, which is finite if $|C_{rQ,l}|$ is finite. □

Note that for $|C_{rQ,l}|$ to be finite, $|C_{Q,l}|$ must be finite, and hence also $|C_{\mathrm{cmp},l}|$ must be finite.

Definition 3.6. *For each $k \geq 1$ and $l \geq 0$, each refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$, let $\approx_{rQk,l}$ be an equivalence relation on l-terminal graphs, which is defined as follows. For each two l-terminal graphs G_1 and G_2,*

$$G_1 \approx_{rQk,l} G_2 \Leftrightarrow G_1 \approx_{rQ,l} G_2 \ \wedge\ G_1 \sim_{TW_k,l} G_2.$$

The analogy of Lemma 2.1 also holds for $\approx_{rQk,l}$.

Lemma 3.3. *For each $l \geq 0$, if $|C_{rQ,l}|$ is finite, then so is the number of equivalence classes of $\approx_{rQk,l}$.*

Lemma 3.4. *For each $k \geq 1$, $l \geq 0$, each refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$, $\approx_{rQk,l}$ is a refinement of $\sim_{P_k,l}$.*

Proof. Let G_1 and G_2 be l-terminal graphs. Let H be an l-terminal graph, and suppose $P(G_1 \oplus H)$ holds and $\mathrm{tw}(G_1 \oplus H) \leq k$. Then $P(G_2 \oplus H)$ holds because of Lemma 3.1. Furthermore, $\mathrm{tw}(G_2 \oplus H) \leq k$ because of Definition 3.6. Hence for all l-terminal graphs H, $P(G_1 \oplus H) \wedge \mathrm{tw}(G_1 \oplus H) \leq k \Leftrightarrow P(G_2 \oplus H) \wedge \mathrm{tw}(G_2 \oplus H) \leq k$. □

We now come to the main result of this section.

Theorem 3.1. *Let P be a graph property. Suppose that the following conditions hold.*

1. *P can be written in the form*

$$P(G) = \exists_{S \in D(G)} Q(G, S),$$

   in such a way that domain D is inducible, Q is decidable, a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ is decidable, and $|C_{rQ,l}|$ is finite,

2. *There is a function s, which assigns to each terminal graph G a positive integer, such that for each $S \in D_{[]}(G)$, the number of bits needed to represent S is at most $s(G)$.*

3. *For each two fixed l-terminal graphs H and H', the following holds. For each l-terminal graph G, if $S \in D(G \oplus H)$, then $S[H]$ can be computed from S and H in constant time, and for each $S' \in D_{[]}(H')$, such that $(H, S[H]) \sim_{rQ,l} (H', S')$, $S' \oplus S[G]$ can be computed in constant time from S, S', H and H'.*

Then for each $k \geq 1$, there is a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules for P_k, and there is an implementation of Algorithm ConstructSolution *which can be used to compute for each graph G, in linear time, an $S \in D(G)$ such that $Q(G, S)$ holds, if $P_k(G)$ holds.*

If, in addition, Q and $\sim_{rQ,l}$ are effectively decidable, s is effectively computable, and in condition 3, $S[H]$ and $S' \oplus S[G]$ are effectively computable from S and H, then $\mathcal{R}$ and the implementation of Algorithm ConstructSolution *can be constructed.*

Proof. Since $|C_{rQ,l}|$ is finite, $\approx_{rQk,l}$ has a finite number of equivalence classes, and it is a refinement of $\sim_{P_k,l}$. Let $\mathcal{R}$ be a finite set of safe, complete and terminating reduction rules, such that for each rule $H_1 \to H_2$, $H_1 \approx_{rQk,l} H_2$. Note that this set can be constructed, if $\sim_{rQ,l}$ is effectively decidable, since in that case, $\approx_{rQk,l}$ is effectively decidable (Theorem 2.1).

For each reduction rule $H_1 \to H_2$ in $\mathcal{R}$, keep a table T for H_1, which contains for each possible equivalence class $c \in C_{rQk,l}$, a partial solution $S_1 \in D_{[]}(H_1)$ such that $ec_{rQk,l}(H_1, S_1) = c$, if such a solution exists, and false otherwise.

Let G be a graph. Algorithm ConstructSolution can now be further refined as follows. In line 8, an $S \in D(G)$ (G is the reduced graph here) for which $Q(G, S)$ holds can be constructed as follows. Each possible $S \in D(G)$ is tried, and if $Q(G, S)$ holds, then this solution is taken. Note that this can be done in constant time, because of condition 2, and we can actually do it if s is effectively computable and Q is effectively decidable.

In line 14 of Algorithm ConstructSolution, the construction of S' can be done as follows. First $S[H'_i]$ is computed. Then $c = ec_{rQk,l}(H'_i, S[H'_i])$ is computed. After that $S'' = T(c)$ is obtained, and $S' = S'' \oplus S[H]$ is computed. Note that all these steps can be done in constant time, and we can actually do them if the constant time algorithms to compute $S[H'_i]$ and $S'' \oplus S[H]$ are known.

This completes the proof. □

As an important special case, we consider the graph properties that are MS-definable (see e.g. [9] or [3]). Let $k \geq 1$. Suppose we have a graph property P which can be written as $P(G) = \exists_{S \in D(G)} Q(G, S)$, where $D(G) = D_1(G) \times D_2(G) \times \cdots \times D_t(G)$ for some $t \geq 1$, each $D_i(G)$ is either equal to $V(G)$, to $E(G)$, to $\mathcal{P}(V(G))$ or to $\mathcal{P}(E(G))$, and we have a definition of Q in monadic second order logic. We show that Algorithm ConstructSolution can be used to find for a given graph G an $S \in D(G)$ such that $Q(G, S)$ holds, if $P_k(G)$ holds, and that we can construct the finite, safe, complete and decreasing set of reduction rules that is needed for the algorithm.

For each two l-terminal graphs G and H, each $S = (S_1, \ldots, S_t) \in D(G \oplus H)$, let $S[G] = (S_1[G], \ldots, S_t[G])$, where for each i, $S_i[G]$ is defined as follows.

$$S_i[G] = \begin{cases} S_i \cap V(G) & \text{if } D_i(G) = \mathcal{P}(V(G)) \\ S_i \cap E(G) & \text{if } D_i(G) = \mathcal{P}(E(G)) \\ S_i & \text{if } D_i(G) = V(G) \wedge S_i \in V(G) \\ \epsilon & \text{if } D_i(G) = V(G) \wedge S_i \notin V(G) \\ S_i & \text{if } D_i(G) = E(G) \wedge S_i \in E(G) \\ \epsilon & \text{if } D_i(G) = V(G) \wedge S_i \notin E(G) \end{cases}$$

Hence if $D_i(G \oplus H)$ is $\mathcal{P}(V(G \oplus H))$ or $\mathcal{P}(E(G \oplus H))$, then $D_{[\,],i}(G) = D_i(G)$. If $D_i(G \oplus H)$ is $V(G \oplus H)$ or $E(G \oplus H)$, then $D_{[\,],i}(G) = D_i(G) \cup \{\epsilon\}$.

With this definition of $[\,]$, D is inducible, and $|C_{\mathrm{cmp},l}|$ is finite, for each $l \geq 0$.

Borie et al. [7] have shown that for each $k \geq 1$, there is a homomorphism h, mapping each pair (G, S), where G is an l-terminal graph, $l \leq k$, and $S \in D_{[\,]}(G)$, to an element of a finite set A_k, such that the following conditions hold.

1. For each $l, l' \leq k$, each l-terminal graph G_1 and l'-terminal graph G_2, each $S_1 \in D_{[\,]}(G_1)$ and $S_2 \in D_{[\,]}(G_2)$, if $h(G_1, S_1) = h(G_2, S_2)$, then $Q(G_1, S_1) = Q(G_2, S_2)$.

2. There is a function $f_\oplus : A_k \times A_k \to A_k$, such that for each $l \leq k$, each two l-terminal graphs G and H, each $S \in D_{[\,]}(G)$ and $S' \in D_{[\,]}(H)$, if (G, S) and (H, S') are $\oplus$-compatible, then

$$h(G \oplus H, S \oplus S') = f_\oplus(h(G, S), h(H, S')).$$

This homomorphism can be computed if we have a definition of Q in monadic second order logic.

For each $l \geq 0$, each l-terminal graph G and $S \in D_{[\,]}(G)$, let $ec_l(G, S) = (h(G, S), c)$, where $c \in C_{\mathrm{cmp},l}$ is such that (G, S) belongs to compatibility class c. Furthermore, let $C_l = A_k \times C_{\mathrm{cmp},l}$, and let $(G_1, S_1) \sim_l (G_2, S_2)$ if and only if $ec_l(G_1, S_1) = ec_l(G_2, S_2)$. Since $|A_k|$ and $|C_{\mathrm{cmp},l}|$ are both finite, $|C_l|$ is also finite. We now show that $\sim_l$ is a refinement of $\sim_{Q,l}$.

Let $l \geq 0$, let G_1 and G_2 be l-terminal graphs, let $S_1 \in D_{[\,]}(G_1)$, $S_2 \in D_{[\,]}(G_2)$, such that $G_1 \sim_{Q,l} G_2$. We have to show that for all l-terminal graphs H, all $S \in D_{[\,]}(H)$ such that (G_1, S_1) and (H, S) are $\oplus$-compatible, $Q(G_1 \oplus H, S_1 \oplus S) = Q(G_2 \oplus H, S_2 \oplus S)$. Let H be an l-terminal graph, and let $S \in D_{[\,]}(H)$ such that (G_1, S_1) and (H, S) are $\oplus$-compatible.

Then, since $h(G_1, S_1) = h(G_2, S_2)$,

$$\begin{aligned} h(G_1 \oplus H, S_1 \oplus S) &= f_{\oplus}(h(G_1, S_1), h(H, S)) \\ &= f_{\oplus}(h(G_2, S_2), h(H, S)) \\ &= h(G_2 \oplus H, S_2 \oplus S). \end{aligned}$$

Hence $Q(G_1 \oplus H, S_1 \oplus S) = Q(G_2 \oplus H, S_2 \oplus S)$. This shows that condition 1 of Theorem 3.1 holds.

Condition 2 of Theorem 3.1 is also satisfied, since each $S \in D(G)$ has at most $O(t|V(G)| + t|E(G)|)$ elements (vertices and edges).

Now consider condition 3. We use a data structure for storing tuples $S = (S_1, \ldots, S_t) \in D_{[]}(G)$, which consists of an array of t data structures for each S_i. If S_i is a set of vertices or edges, then these vertices or edges are put in a list. If S_i is a vertex or edge, or ϵ, then this vertex or edge or ϵ is stored. Furthermore, we keep a pointer from each vertex and edge to each place in the data structure where this vertex or edge occurs. There are at most t of these pointers for each vertex and each edge.

For each two fixed l-terminal graphs H and H', each l-terminal graph G, if we have $S \in D(G \oplus H)$ stored in this way, then we can compute $S[H]$ as follows. Make a new data structure for $S[H]$ with each $S_i[H]$ empty for each i. For each vertex v in H, follow the pointers from v to the places in which it occurs in S, and check in which part S_i of S it occurs. Then add v to $S_i[H]$. Do the same for all edges. Then for each i, check if D_i is a set of vertices or edges, but there is no vertex or edge in the data structure at the location of $S_i[H]$, and if so, add ϵ to $S_i[H]$. This can all be done in constant time, since H has constant size, and each vertex or edge occurs at most once in each S_i, so at most t times in S.

Let $S' = (S'_1, \ldots, S'_t) \in D_{[]}(H')$ such that $(H, S) \sim_{rQ,l} (H', S')$. $S' \oplus S[G]$ can be computed as follows. For each vertex v of H which is not a terminal, follow the pointers from v to all places in S where it occurs, and delete it there. Do the same for all edges in H for which at least one end point is not a terminal.

For each vertex v of H' which is a terminal, follow the pointers from v to all pointers in S' where it occurs, and delete v at that place. Do the same for all edges in H' of which both end points are terminals.

Next, for each i, $1 \leq i \leq t$, append the list S'_i to the list S_i. The resulting data structure represents $S' \oplus S[G]$. Hence Algorithm ConstructSolution can be used.

The following theorem follows.

Theorem 3.2. *Let $P(G) = \exists_{S \in D(G)}\ Q(G, S)$ and let $k \geq 1$. If Q is MS-definable, and $D(G) = D_1(G) \times \cdots \times D_t(G)$, for some $t \geq 1$, such that for each i, $D_i(G)$ is either $\mathcal{P}(V(G))$, $\mathcal{P}(E(G))$, $V(G)$ or $E(G)$, then there is a finite set of safe, complete and terminating reduction rules and an implementation of Algorithm* ConstructSolution *which can be used to construct in linear time an $S \in D(G)$ such that $Q(G, S)$ holds, if $P_k(G)$ holds.*

If in addition, we have a definition of Q in monadic second order logic, then such a set of reduction rules and implementation of Algorithm ConstructSolution *can be constructed.*

As a corollary, we also have the following.

Corollary 3.1. *Let P be a graph property. Suppose that P can be written in the form*

$$P(G) = \exists_{S \in D_1(G) \times \cdots \times D_t(G)} Q(G, S),$$

in such a way that for each G and i, $D_i(G)$ is equal to $V(G)$, $E(G)$, $\mathcal{P}(V(G))$ or $\mathcal{P}(E(G))$, and furthermore Q is decidable, a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ is decidable, and $|C_{rQ,l}|$ is finite. Then for each $k \geq 1$, there is a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules for P_k and an implementation of Algorithm ConstructSolution *which can be used to compute for each graph G, in linear time, an $S \in D(G)$ such that $Q(G, S)$ holds, if $P_k(G)$ holds.*

If, in addition, Q and $\sim_{rQ,l}$ are effectively decidable, then $\mathcal{R}$ and the implementation of Algorithm ConstructSolution *can be constructed.*

4 Optimization Problems

In this section we show how the idea of reduction algorithms can be extended to optimization problems.

Let R be a function, mapping the set of graphs to $\mathbb{Z} \cup \{\text{false}\}$. Typically, R will be an optimization problem, like independent set, vertex cover, etc. We will call R a graph optimization problem. The value false is used to denote that a certain condition does not hold. Denote $\mathcal{Z} = \mathbb{Z} \cup \{\text{false}\}$. Define addition on $\mathcal{Z}$ as follows: if $i, j \in \mathbb{Z}$, then we take for $i + j$ the usual sum, and for all $i \in \mathcal{Z}$, $i + \text{false} = \text{false} + i = \text{false}$.

Instead of reduction rules, we use *reduction-counter* rules for graph optimization problems.

Definition 4.1. *A* reduction-counter rule *is a pair $((H, H'), i)$, where (H, H') is a reduction rule, and $i \in \mathbb{Z}$. An application of reduction-counter rule $((H, H'), i)$ is the operation, that takes a counter $cnt \in \mathbb{Z}$ and a graph G of the form $G_1 \oplus G_3$, with G_1 isomorphic to H_1, and replaces cnt by $cnt + i$ and G by the graph $G_2 \oplus G_3$, with G_2 isomorphic to H_2. We write $G \xrightarrow{r} G_2 \oplus G_3$.*

For two graphs G and G', and a set of reduction-counter rules $\mathcal{R}$, we write $G \xrightarrow{\mathcal{R}} G'$, if there exists an $r = ((H, H'), i) \in \mathcal{R}$ with $G \xrightarrow{r} G'$.

To be able to use a reduction algorithm with reduction-counter rules for graph optimization problems, we need a notion of finiteness, safeness, completeness, termination and decrease for a set of reduction-counter rules.

Definition 4.2. *Let R be a graph optimization problem. Let $\mathcal{R}$ be a set of reduction-counter rules.*

- *$\mathcal{R}$ is* safe *for R if, whenever $G \xrightarrow{r} G'$ for some $r = ((H, H'), i) \in \mathcal{R}$, then $R(G) = R(G') + i$.*
- *$\mathcal{R}$ is* complete *for R if the set of graphs $\{G \mid R(G) \neq \text{false} \wedge \neg\exists G' : G \xrightarrow{\mathcal{R}} G'\}$ is finite.*
- *$\mathcal{R}$ is* terminating *if there does not exist an infinite sequence $G_1 \xrightarrow{\mathcal{R}} G_2 \xrightarrow{\mathcal{R}} G_3 \xrightarrow{\mathcal{R}} \cdots$.*
- *$\mathcal{R}$ is* decreasing *if whenever $G \xrightarrow{\mathcal{R}} G'$, then G' contains fewer vertices than G.*

To solve a graph optimization problem R on a graph G with a linear time reduction algorithm, we can now use a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction-counter rules as follows. Apply the reduction algorithm as usual, using R, but maintain during the reduction an integer counter. Initially, this counter is equal to zero, and after applying a reduction rule $((H, H'), i)$, the counter is increased by i. Let G_j denote the graph after the jth reduction is applied, and let cnt_j denote the value of the counter at this moment. It is important to note that the sum of $R(G_j)$ and the counter is equal for all j. Thus, at each moment in the reduction algorithm, $R(G) = R(G_j)+cnt_j$. Hence, when G has been rewritten to a small graph G_t, and G_t is in the finite set $F = \{G \mid R(G) \neq \text{false} \wedge \neg\exists_{G'}\ G \xrightarrow{\mathcal{R}} G'\}$, then $R(G) = R(G_t) + cnt_t$, which can be computed easily, since G_t is small. However, if G_t is not in F, then $R(G) = \text{false}$.

In analogy with $\sim_{P,l}$ for graph properties P, we define $\sim_{R,l}$ for graph optimization problems R.

Definition 4.3. *For a graph optimization problem R the equivalence relation $\sim_{R,l}$ on l-terminal graphs is defined as follows.*

$$G_1 \sim_{R,l} G_2 \Leftrightarrow \exists_{i\in\mathbb{Z}}\ \forall_{l\text{-terminals graphs } H}\ R(G_1 \oplus H) = R(G_2 \oplus H) + i.$$

Optimization problem R is of finite integer index *if the number of equivalence classes of $\sim_{R,l}$ is finite, for each fixed l.*

Note that a if reduction-counter rule $((H, H'), i)$ is safe for a graph optimization problem R, then $H \sim_{R,l} H'$. Furthermore, if $H \sim_{R,l} H'$ for two l-terminal graphs H and H', then there is a reduction-counter rule $((H, H'), i)$ for some $i \in \mathbb{Z}$.

For given R, let $C_{R,l}$ be the set of equivalence classes of $\sim_{R,l}$ and for each l-terminal graph G, let $ec_{R,l}(G) = c$ if $c \in C_{R,l}$ and G belongs to equivalence class c.

For a graph optimization problem R and an integer $k \geq 1$, the graph optimization problem R_k is defined as

$$R_k(G) = \begin{cases} \text{false} & \text{if } \mathrm{tw}(G) > k \\ R(G) & \text{otherwise} \end{cases}$$

Lemma 4.1. *If R is of finite integer index, then for each $k \geq 1$, R_k is of finite integer index.*

Proof. For each $l \geq 0$, let $\sim_l$ be the equivalence relation on l-terminal graphs which is defined as follows.

$$G_1 \sim_l G_2 \Leftrightarrow G_1 \sim_{R,l} G_2 \wedge G_1 \sim_{TW_k,l} G_2$$

If $G_1 \sim_l G_2$, then $G_1 \sim_{R,l} G_2$ and $G_1 \sim_{TW_k,l} G_2$, and hence there is an $i \in \mathbb{Z}$, such that for all l-terminal graphs H, $R_k(G_1 \oplus H) = R_k(G_2 \oplus H) + i$. Hence $\sim_l$ is a refinement of $\sim_{R_k,l}$. Furthermore, for each $l \geq 0$, $\sim_l$ has a finite number of equivalence classes, hence so has $\sim_{R_k,l}$. □

The following theorem is the analogy of Theorem 2.1 for finite integer index problems.

Theorem 4.1. *Let k be a constant, suppose R is a graph optimization problem which is of finite integer index. Then there exists a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction-counter rules for R_k. Moreover, for each reduction-counter rule $((H, H'), i) \in \mathcal{R}$, H and H' are open, and if H has one or more terminals, then H is connected.*

If, in addition, there is an equivalence relation $\sim_l$ for each $l \geq 0$, which is a refinement of $\sim_{R,l}$, which is effectively decidable, has a finite number of equivalence classes, and for each pair H, H', if $H \sim_l H'$, then we can effectively compute an $i \in \mathbb{Z}$ such that for each G, $R(H \oplus G) = R(H' \oplus G) + i$, then such a set $\mathcal{R}$ of reduction-counter rules can be constructed, and for each $((H, H'), i) \in \mathcal{R}$, $H \sim_l H'$.

Proof. Let P be the graph property, defined as follows. For each graph G, $P(G) = (R(G) \neq \mathrm{false})$. For each $l \geq 0$, $\sim_{R,l}$ is a refinement of $\sim_{P,l}$, and $\sim_{R_k,l}$ is a refinement of $\sim_{P_k,l}$. Hence, with Corollary 2.1, there is a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules for P_k, such that for each $(H, H') \in \mathcal{R}$, $H \sim_{R_k,l} H'$. For each reduction rule (H, H'), make a reduction-counter rule $((H, H'), i)$, where $i = 0$ if for all G, $R(H \oplus G) = \mathrm{false}$ (and hence $R(H' \oplus G) = \mathrm{false}$), $i = R(H \oplus G) \Leftrightarrow R(H' \oplus G)$ for some G such that $R(H \oplus G) \in \mathbb{Z}$ otherwise. Let $\mathcal{R}'$ denote the set of all these reduction-counter rules. Then $\mathcal{R}'$ is a finite, safe, complete and decreasing set of reduction-counter rules for R.

If we have a refinement $\sim_l$ of $\sim_{R,l}$, for each $l \geq 0$, then we can construct a finite, safe, complete and decreasing set $\mathcal{R}$ of reduction rules for P_k, such that for each rule $(H, H') \in \mathcal{R}$, $H \sim_l H'$. If we can compute an $i \in \mathbb{Z}$ for which $R(H \oplus G) = R(H' \oplus G) + i$ for each G, then we can again turn each of these rules (H, H') in a reduction-counter rule $((H, H'), i)$ with i defined as above. □

Note that the algorithm of Arnborg et al. [2] can easily be adapted to solve a graph optimization problem with a finite, safe, complete and decreasing set of reduction-counter rules. In Section 6 we show that the efficient parallel algorithm of [6] can be adapted to solve graph optimization problems with a finite, safe, complete and decreasing set of reduction-counter rules.

In the remainder of this section, we therefore give a method which makes it easier to prove that a graph optimization problem is of finite integer index if it has the following form.

$$R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\},$$

where D is a solution domain, for each $S \in D(G)$, $z(S) \in \mathbb{Z}$, and either $\mathrm{opt} = \max$ or $\mathrm{opt} = \min$. (If there is no $S \in D(G)$ for which $Q(G, S)$ holds, then we define $R(G)$ to be false.)

Also, we use this method for a number of problems to prove that they are of finite integer index, and we show for a number of other problems that they are not of finite integer index.

Given R, z, D, as above, and a fixed refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$, we define for each l-terminal graph G and $c \in C_{rQ,l}$ $\mathrm{opt}(G, c) \in \mathcal{Z}$ as follows.

$$\mathrm{opt}(G, c) = \mathrm{opt}\{z(S) \mid S \in D_{[]}(G) \wedge ec_{rQ,l}(G, S) = c\}$$

If there is an $S \in D_{[]}(G)$ such that $ec_{rQ,l}(G, S) = c$, then let $\mathrm{optS}(G, c) \in D_{[]}(G)$ be such that $ec_{rQ,l}(G, \mathrm{optS}(G, c)) = c$ and $z(\mathrm{optS}(G, c)) = \mathrm{opt}(G, c)$. $\mathrm{opt}(G, c)$ represents

'the value of the best partial solution on G in equivalence class c', and $\mathrm{optS}(G, c)$ gives such a partial solution (if existing).

Let $\sim_{Q,l}$ be as defined in Section 3.

Theorem 4.2. *Let $R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\}$. Suppose D is inducible for $[\,]$ and there is a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ for which the following conditions hold.*

1. *For each $l \geq 0$, $|C_{rQ,l}|$ is finite ($C_{rQ,l}$ is the set of equivalence classes of $\sim_{rQ,l}$).*
2. *Function z can be extended to the domain of partial solutions for terminal graphs (i.e. $z : D_{[]}(G) \to \mathbb{Z}$ for each terminal graph G) such that the following holds.*
   (a) *For each $l \geq 0$, each $c, c' \in C_{rQ,l}$, if c and c' are $\oplus$-compatible, then there is a constant $d_l(c, c') \in \mathbb{Z}$ such that for all l-terminal graphs G and H, all $S' \in D(G)$, $S'' \in D(H)$, if $(G, S) \in c$ and $(H, S') \in c'$, then $z(S' \oplus S'') = z(S') + z(S'') \Leftrightarrow d_l(c, c')$.*
   (b) *For each $l \geq 0$, there is a constant $K_l \in \mathbb{N}$, and for each l-terminal graph G there is an integer $i_G \in \mathbb{Z}$, such that for each partial solution $S \in D_{[]}(G)$, if $|z(S) \Leftrightarrow i_G| > K_l$, then S can not lead to an optimal solution, i.e. for each l-terminal graph H, for each $S' \in D(G \oplus H)$, if $Q(G \oplus H, S')$ holds and $S'[G] = S$, then $z(S') \neq R(G \oplus H)$.*

Then there is an equivalence relation $\sim_l$ which is a refinement of $\sim_{R,l}$ and has a finite number of equivalence classes for each $l \geq 0$, and hence R is of finite integer index.

If, in addition, $\sim_{rQ,l}$ is effectively decidable, there is an effectively computable function s, which assigns to each graph G a positive integer, such that for each $S \in D_{[]}(G)$, the number of bits to store S is at most $s(G)$, and $z(S)$ is effectively computable for each terminal graph G and each $S \in D_{[]}(G)$, and for each terminal graph G, i_G is effectively computable, then $\sim_l$ is effectively decidable, and for each two l-terminal graphs H and H', if $H \sim_l H'$, then we can effectively compute an $i \in \mathbb{Z}$, such that for each l-terminal graph G, $R(H \oplus G) = R(H' \oplus G) + i$.

Proof. Suppose conditions 1 and 2 hold for R. Let z on partial solution domains be defined as in condition 2. For each $l \geq 0$, let d_l be as in condition 2(a), let $K_l \in \mathbb{N}$ be as in condition 2(b), and for all l-terminal graphs G, let $i_G \in \mathbb{Z}$ be as in condition 2(b).

We construct an equivalence relation $\sim_l$ on l-terminal graphs, such that $\sim_l$ is a refinement of $\sim_{R,l}$, and we show that $\sim_l$ has a finite number of equivalence classes for each l.

For each $l \geq 0$, each l-terminal graph G, let f_G be a function mapping each $c \in C_{rQ,l}$ to the set $\{\Leftrightarrow K_l, \ldots, K_l\} \cup \{\mathrm{false}\}$, and, for each $c \in C_l$, let

$$f_G(c) = \begin{cases} \max(G, c) \Leftrightarrow i_G & \text{if } \Leftrightarrow K_l \leq \max(G, c_G) \Leftrightarrow i_G \leq K_l \\ \mathrm{false} & \text{otherwise.} \end{cases}$$

For each $l \geq 0$, let $G_1 \sim_l G_2 \Leftrightarrow f_{G_1} = f_{G_2}$, and let C_l denote the set of equivalence classes of $\sim_l$.

$|\{\Leftrightarrow K_l, \ldots, K_l\} \cup \{\text{false}\}| = 2K_l + 2$, which depends only on l. Furthermore, for each $l \geq 0$, $|C_{rQ,l}|$ is constant, which means that there are at most a constant number of different functions f_G, so $|C_l|$ is finite.

We now have to prove that $\sim_l$ is a refinement of $\sim_{R,l}$, i.e. we have to prove that for all l-terminal graphs G_1 and G_2, if $G_1 \sim_l G_2$, then there is an $i \in \mathbb{Z}$, such that for all l-terminal graphs H, $R(G_1 \oplus H) = R(G_2 \oplus H) + i$. We only show this for the case that $\text{opt} = \max$. If $\text{opt} = \min$, the proof is similar.

Suppose $f_{G_1} = f_{G_2} = f$. We show that $R(G_1 \oplus H) = R(G_2 \oplus H) + i_{G_1} \Leftrightarrow i_{G_2}$ (where i_{G_1} and i_{G_2} are the integers as defined in condition 2(b) of the theorem).

First consider the case that $R(G_1 \oplus H) = \text{false}$. Then $\{S \in D(G_1 \oplus H) \mid Q(G_1 \oplus H, S)\} = \emptyset$, since $z(S) \in \mathbb{Z}$ for each $S \in D(G_1 \oplus H)$. This means that for each $c \in C_l$, if $f(c) \neq \text{false}$ (hence there is an $S_1 \in D_{[]}(G_1)$ with $ec_l(G_1, S_1) = c$ and S_1 can lead to an optimal solution), then for all $c' \in C_l$, if c and c' are $\oplus$-compatible, $f_H(c') = \text{false}$ (i.e. there is no $S_H \in D_{[]}(H)$ such that (G_1, S_1) and (H, S_H) are $\oplus$-compatible and S_H may lead to an optimal solution). This also means that for all $S_2 \in D_{[]}(G_2)$, if S_2 can lead to an optimal solution, then there is no $S_H \in D_{[]}(H)$ such that (G_2, S_2) and (H, S_H) are $\oplus$-compatible and S' may lead to an optimal solution. Hence $R(G_1 \oplus H) \Leftrightarrow i_{G_1} = \text{false} = R(G_2 \oplus H) \Leftrightarrow i_{G_2}$.

Suppose $R(G_1 \oplus H) \in \mathbb{Z}$. Let $S \in D(G_1 \oplus H)$ be such that $Q(G_1 \oplus H, S)$ holds and $z(S) = R(G_1 \oplus H)$. Let $S_H = S[H]$, $S_1 = S[G_1]$, $c = ec_{rQ,l}(G_1, S_1)$ and $c' = ec_{rQ,l}(H, S_H)$. We first show that $z(S_1) = \max(G_1, c)$. Suppose not. Then there is an $S_1' \in D_{[]}(G_1)$ such that $ec_{rQ,l}(G_1, S_1') = c$ and $z(S_1') = \max(G_1, c)$. But then $Q(G_1 \oplus H, S_1' \oplus S_H) = Q(G_1 \oplus H, S_1 \oplus S_H) = \text{true}$, and $z(S_1' \oplus S_H) = z(S_1') + z(S_H) \Leftrightarrow d_l(c, c') > z(S_1) + z(S_H) \Leftrightarrow d_l(c, c') = z(S_1 \oplus S_H) = R(G_1 \oplus H)$, which is a contradiction.

Since S is optimal and $z(S_1) = \max(G_1, c)$, it must hold that $z(S_1) = i_{G_1} + f(c)$. Since $f(c) \in \mathbb{Z}$, $\max(G_2, c) \in \mathbb{Z}$, and hence there is an $S_2 \in D_{[]}(G_2)$ such that $ec_{rQ,l}(G_2, S_2) = c$ and $z(S_2) = \max(G_2, c)$. Furthermore, $Q(G_2 \oplus H, S_2 \oplus S_H)$ holds, and

$$
\begin{aligned}
R(G_2 \oplus H) &\geq z(S_2 \oplus S_H) \\
&= z(S_2) + z(S_H) \Leftrightarrow d_l(c, c') \\
&= \{z(S_2) = \max(G_2, H) = f(c) + i_{G_2}\} \\
&\quad f(c) + i_{G_2} + z(S_H) \Leftrightarrow d_l(c, c') \\
&= f(c) + i_{G_1} \Leftrightarrow i_{G_1} + i_{G_2} + z(S_H) \Leftrightarrow d_l(c, c') \\
&= \{z(S_1) = f(c) + i_{G_1}\} \\
&\quad z(S_1) + z(S_H) \Leftrightarrow d_l(c, c') \Leftrightarrow i_{G_1} + i_{G_2} \\
&= z(S) \Leftrightarrow i_{G_1} + i_{G_2} \\
&= R(G_1 \oplus H) \Leftrightarrow i_{G_1} + i_{G_2}.
\end{aligned}
$$

Hence $R(G_1 \oplus H) \Leftrightarrow i_1 \leq R(G_2 \oplus H) \Leftrightarrow i_2$. By symmetry, $R(G_2 \oplus H) \Leftrightarrow i_2 \leq R(G_1 \oplus H) \Leftrightarrow i_1$, which means that $R(G_1 \oplus H) \Leftrightarrow i_1 = R(G_2 \oplus H) \Leftrightarrow i_2$.

We have now shown that $\sim_l$ is a refinement of $\sim_{R,l}$, for each $l \geq 0$. Since $|C_l|$ is finite for each l, this means that R is of finite integer index.

If $\sim_{rQ,l}$ is effectively decidable, we have an effectively computable function s, i_G is effectively computable for each G, and $z(S)$ is effectively computable for each S, then $\sim_l$ is effec-

tively decidable, and we can effectively compute for each pair of l-terminal graphs H, H', an $i \in \mathbb{Z}$ such that for each l-terminal graph G, $R(H \oplus G) = R(H' \oplus G) + i$. □

While the theorem may seem complex to use, it is in most cases not hard to find an equivalence relation $\sim_{rQ,l}$ which satisfies conditions 1 and 2(a). Only condition 2(b) is often not easy to prove. Therefore, we give two other theorems, which are weaker than Theorem 4.2, but easier to use for showing problems to be of finite integer index, as will be demonstrated later in this section.

Theorem 4.3. *Let $R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G,S)\}$. Suppose D is inducible for $[\,]$ and there is a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ for which condition 1 of Theorem 4.2 holds, and z can be extended to the domain of partial solutions such that condition 2(a) of Theorem 4.2 holds, and*

2(c) for each $l \geq 0$, there is a constant $K_l \in \mathbb{N}$, such that for each l-terminal graph G and for each $c, c' \in C_l$, if $\mathrm{opt}(G,c) \neq \mathrm{false}$ and $\mathrm{opt}(G,c') \neq \mathrm{false}$ and partial solutions in class c or c' may lead to a solution, then $|\mathrm{opt}(G,c) \Leftrightarrow \mathrm{opt}(G,c')| \leq K_l$.

Then condition 2(b) of Theorem 4.2 also holds.

Proof. Suppose conditions 1 and 2(a) of Theorem 4.2 hold, and condition 2(c) holds for R. Let z be as in condition 2 and for each $l \geq 0$, let d_l be as in condition 2(a), let $K_l \in \mathbb{N}$ be as in condition 2(c). For all l-terminal graphs G, let $i_G = 0$ if there is no $S \in D_{[\,]}(G)$ which can lead to a solution, otherwise, let $i_G = \mathrm{opt}(G,c)$ for some $c \in C_l$ which may lead to a solution.

We now show that with these definitions of K_l and i_G, condition 2(b) of Theorem 4.2 holds. We only consider the case that $\mathrm{opt} = \max$. The case that $\mathrm{opt} = \min$ can be proved similarly.

Let G be an l-terminal graph, let $S \in D_{[\,]}(G)$, let $c = ec_l(G,S)$, and suppose $|z(S) \Leftrightarrow i_G| > K_l$. If $z(S) < \mathrm{opt}(G,c)$, then S can not lead to an optimal solution, since for each l-terminal graph H, each $S' \in D(G \oplus H)$, if $S'[G] = S$, then $z(S') < z(\mathrm{optS}(G,c) \oplus S'[H])$, and $Q(G \oplus H, S')$ holds if and only if $Q(G \oplus H, \mathrm{optS}(G,c) \oplus S[H])$ holds.

If $z(S) = \mathrm{opt}(G,c)$, then by condition 2(c), $|z(S) \Leftrightarrow i_G| \leq K_l$. □

Theorem 4.4. *Let $R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G,S)\}$. Suppose D is inducible for $[\,]$ and there is a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ for which condition 1 holds, and z can be extended to the domain of partial solutions such that condition 2(a) holds, and*

2(d) for each $l \geq 0$, there is a constant $K'_l \in \mathbb{N}$, and with each l-terminal graph G, we can associate an equivalence class $c_G \in C_{rQ,l}$, such that the following holds.

(i) For all l-terminal graphs G and H, and $S \in D_{[\,]}(G)$, $S' \in D_{[\,]}(H)$, if $ec_{rQ,l}(G,S) = c_G$ and $ec_{rQ,l}(H,S') = c_H$, then (G,S) and (H,S') are $\oplus$-compatible, and $Q(G \oplus H, S \oplus S')$ holds.

(ii) If $\mathrm{opt} = \max$, then for all l-terminals graphs G, all $S \in D_{[\,]}(G)$, if S can lead to a solution (i.e. there is an (H,S') such that $Q(G \oplus H, S \oplus S')$ holds), then $z(S) \Leftrightarrow \mathrm{opt}(G, c_G) \leq K_l$.

(iii) If $\mathrm{opt} = \min$*, then for all* l*-terminals graphs* G*, all* $S \in D_{[]}(G)$*, if* S *can lead to a solution, then* $\mathrm{opt}(G, c_G) \Leftrightarrow z(S) \leq K_l$*.*

Then condition 2(b) of Theorem 4.2 also holds.

Proof. Suppose conditions 1 and 2(a) of Theorem 4.2 hold, and condition 2(d) holds for R. Let z be as in condition 2, and for each $l \geq 0$, let d_l be as in condition 2(a), let $K'_l \in \mathbb{N}$ be as in condition 2(d), and for all l-terminal graphs G, let $c_G \in C_{rQ,l}$ be as in condition 2(d).

For each $l \geq 0$, let

$$K_l = K'_l + 2\max\{|d_l(c, c')| \mid c, c' \in C_{rQ,l} \wedge c \text{ and } c' \text{ are } \oplus\text{-compatible}\},$$

and for each l-terminal graph G, let

$$i_G = \begin{cases} \mathrm{opt}(G, c_G) & \text{if } \mathrm{opt}(G, c_G) \neq \mathrm{false} \\ 0 & \text{otherwise.} \end{cases}$$

We now show that with these definitions of K_l and i_G, condition 2(b) of Theorem 4.2 holds. We only consider the case that $\mathrm{opt} = \max$. The case that $\mathrm{opt} = \min$ can be proved similarly.

Let G be an l-terminal graph, let $S \in D_{[]}(G)$, and suppose $|z(S) \Leftrightarrow i_G| > K_l$. We have to show that S can not lead to an optimal solution. Suppose S leads to a solution. Let H be an l-terminal graph and suppose there is an $S' \in D(G \oplus H)$ such that $Q(G \oplus H, S')$ holds and $S'[G] = S$. We show that $z(S') < R(G \oplus H)$. Let $S_H = S'[H]$, let $c = ec_{rQ,l}(G, S)$ and let $c' = ec_{rQ,l}(H, S_H)$.

Because of condition 2(d)(ii), $z(S) \Leftrightarrow \max(G, c_G) \leq K'_l$, which means that $z(S) \neq \mathrm{false}$ and $\max(G, c_G) \neq \mathrm{false}$, hence $i_G = \max(G, c_G)$. This means that $z(S) \Leftrightarrow i_G \leq K_l$, and hence $i_G \Leftrightarrow z(S) > K_l$, so $z(S) < \max(G, c_G) \Leftrightarrow K_l$. Furthermore, because of condition 2(d)(ii), $z(S_H) \leq \max(H, c_H) + K'_l$. This means that

$$\begin{aligned}
z(S') &= z(S) + z(S_H) \Leftrightarrow d_l(c, c') \\
&< \max(G, c_G) \Leftrightarrow K_l + \max(H, c_H) + K'_l \Leftrightarrow d_l(c, c') \\
&= \{ \text{ by definition of } K_l \} \\
&\quad \max(G, c_G) + \max(H, c_H) \Leftrightarrow d_l(c, c') \Leftrightarrow 2\max\{d_l(c, c') \mid c, c' \in C_{rQ,l}\} \\
&\leq z(\mathrm{maxS}(G, c_G)) + z(\mathrm{maxS}(H, c_H)) \Leftrightarrow \max\{d_l(c, c') \mid c, c' \in C_{rQ,l}\} \\
&= \{ \text{ condition 2(d)(i)} \} \\
&\quad z(\mathrm{maxS}(G, c_G) \oplus \mathrm{maxS}(H, c_H)) \\
&\qquad + d_l(c_G, c_H) \Leftrightarrow \max\{d_l(c, c') \mid c, c' \in C_{rQ,l}\} \\
&\leq z(\mathrm{maxS}(G, c_G) \oplus \mathrm{maxS}(H, c_H)) \\
&\leq R(G \oplus H).
\end{aligned}$$

Hence $z(S') < R(G \oplus H)$. This completes the proof. □

We now give a number of graph optimization problems for each of which we can either prove that it is of finite integer index by using the method of Theorem 4.2, Theorem 4.3, or Theorem 4.4, or we can prove that it is not of finite integer index.

Definition 4.4 (INDUCED BOUNDED DEGREE SUBGRAPH)
Given: A fixed integer constant $p \geq 0$, a graph $G = (V, E)$.
Find: The maximum value of $|S|$, where $S \subseteq V$ and all vertices in $G[S]$ have degree at most p.

For $p = 0$, this is the INDEPENDENT SET problem.

Definition 4.5 (p-DOMINATING SET)
Given: A fixed integer constant $p \geq 1$, a graph $G = (V, E)$.
Find: The minimum value of $|S|$, where $S \subseteq V$ and all vertices in $V \Leftrightarrow S$ have at least p neighbors in S.

For $p = 1$, this is the DOMINATING SET problem.

Definition 4.6 (PARTITION INTO CLIQUES)
Given: A graph $G = (V, E)$.
Find: The minimum value of s, such that there is a partition $\{V_1, \ldots, V_s\}$ of V in which for each i, $1 \leq i \leq s$, $G[V_i]$ is a complete graph.

Definition 4.7 (COVERING BY CLIQUES)
Given: A graph $G = (V, E)$.
Find: The minimum value of s, such that there is a set $\{V_1, \ldots, V_s\}$, in which for each i, $1 \leq i \leq s$, $V_i \subseteq V$, $G[V_i]$ is a complete graph, and for each edge $e \in E$, there is an i, $1 \leq i \leq s$, such that $e \in E(G[V_i])$.

Definition 4.8 (HAMILTONIAN PATH COMPLETION NUMBER)
Given: A graph $G = (V, E)$.
Find: The minimum value of $|S|$, where $S \subseteq \{\{u, v\} \mid u, v \in V\}$, such that $G' = (V, E \cup S)$ contains a Hamiltonian path.

Definition 4.9 (MAXIMUM CUT)
Given: A graph $G = (V, E)$.
Find: The maximum value of $z((V_1, V_2))$, where (V_1, V_2) partitions V, and $z(V_1, V_2) = |\{\{v, w\} \in E \mid v \in V_1 \wedge w \in V_2\}|$.

Definition 4.10 (MAXIMUM LEAF SPANNING TREE)
Given: A graph $G = (V, E)$.
Find: The maximum value of $z(T)$, where T is a spanning tree of G, and $z(T)$ denotes the number of vertices of degree one of T.

Definition 4.11 (LONGEST PATH)
Given: A graph $G = (V, E)$.
Find: The maximum value of s, such that there is a path $(v_1, v_2, \ldots, v_s)$ in G.

Definition 4.12 (LONGEST CYCLE)
Given: A graph $G = (V, E)$.
Find: The maximum value of s, such that there is a path $(v_1, \ldots, v_s)$ in G, and $\{v_s, v_1\} \in E$.

(The problem HAMILTONIAN CIRCUIT COMPLETION NUMBER can be solved using results for HAMILTONIAN PATH COMPLETION NUMBER and is not further discussed here.)

Theorem 4.5. *The following graph optimization problems are of finite integer index:*

1. INDUCED BOUNDED DEGREE SUBGRAPH *for all* $p \geq 0$,
2. p-DOMINATING SET *for all* $p \geq 1$,
3. MAXIMUM CUT *on graphs with bounded degree,*
4. PARTITION INTO CLIQUES,
5. HAMILTONIAN PATH COMPLETION NUMBER, *and*
6. MAXIMUM LEAF SPANNING TREE.

For each of these problems on graphs with bounded treewidth, a finite, safe, complete and decreasing set of reduction-counter rules can be constructed.

Proof.

1 INDUCED BOUNDED DEGREE SUBGRAPH. Let $p \geq 0$ be fixed. $D(G) = \mathcal{P}(V)$, and for a given graph and solution $S \in D(G)$,

$$Q(G, S) = \forall_{v \in S} \, |N_{G,S}(v)| \leq p,$$

where

$$N_{G,S}(v) = \{w \in S \mid \{v, w\} \in E(G)\},$$

$z(S) = |S|$, and $\mathrm{opt} = \max$. For two l-terminal graphs G and H, and $S \in D(G \oplus H)$, let $S[G] = S \cap V(G)$, and let $z(S[G]) = |S[G]|$. Hence $D_{[]}(G) = D(G)$, and two solutions $S \in D_{[]}(G)$ and $S' \in D_{[]}(H)$ are compatible and $\oplus$-compatible if they contain the same terminals.

For each $l \geq 0$, let $I_l = \{1, \ldots, l\}$, let $F_l = \{\{i, j\} \mid 1 \leq i < j \leq l\}$, and for each l-terminal graph $G = (V, E, \langle x_1, \ldots, x_l \rangle)$, let $F(G) = \{\{i, j\} \mid \{x_i, x_j\} \in E\}$.

For each $l \geq 0$, let

$$\begin{aligned} C_{rQ,l} \;=\; & \{(I, \mathrm{false}) \mid I \subseteq I_l\} \cup \{(F, I, N) \mid F \subseteq F_l \\ & \wedge I \subseteq I_l \wedge N \subseteq \{(i, n) \mid i \in I_l \wedge n \in \{1, \ldots, p\}\}. \end{aligned}$$

$|C_{rQ,l}|$ is bounded, because p is fixed. For each l-terminal graph $G = (V, E, \langle x_1, \ldots, x_l \rangle)$, each $S \in D_{[]}(G)$, let $ec_{rQ,l}(G, S) \in C_{rQ,l}$ be defined as follows. If there is a $v \in S$ such that $|N_{G,S}(v)| > p$, then $ec_{rQ,l}(G, S) = (I, \mathrm{false})$, where $I = \{i \in I_l \mid x_i \in S\}$, otherwise, $ec_{rQ,l}(G, S) = (F, I, N)$, where

$$\begin{aligned} F &= F(G), \\ I &= \{i \in I_l \mid x_i \in S\}, \\ N &= \{(i, |N_{G,S}(x_i)|) \mid i \in I\}. \end{aligned}$$

Let $G_1 = (V_1, E_1, \langle x_1, \ldots, x_l \rangle)$ and $G_2 = (V_2, E_2, \langle y_1, \ldots, y_l \rangle)$ be two l-terminal graphs, let $S_1 \in D_{[]}(G_1)$ and $S_2 \in D_{[]}(G_2)$. $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$ if and only if $ec_{rQ,l}(G_1, S_1) = ec_{rQ,l}(G_2, S_2)$.

We first show that $\sim_{rQ,l}$ is a refinement of $\sim_{Q,l}$ for all l. Suppose $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$. Clearly, (G_1, S_1) and (G_2, S_2) are compatible. We have to show that for each l-terminal graph $H = (V_H, E_H, \langle z_1, \ldots, z_l \rangle)$, each $S_H \in D_{[]}(H)$ such that (G_1, S_1) and (H, S_H) are $\oplus$-compatible, $Q(G_1 \oplus H, S_1 \oplus S_H)$ holds if and only if $Q(G_2 \oplus H, S_2 \oplus H)$ holds. Let H be an l-terminal graph, let $S_H \in D_{[]}(H)$ such that (G_1, S_1) and (H, S_H) are $\oplus$-compatible. If $ec_{rQ,l}(G_1, S_1) = ec_{rQ,l}(G_2, S_2) = (I, \text{false})$ for some $I \subseteq I_l$, then $Q(G_1 \oplus H, S_1 \oplus S_H) = \text{false} = Q(G_2 \oplus H, S_2 \oplus S_H)$. Suppose $ec_{rQ,l}(G_1, S_1) = ec_{rQ,l}(G_2, S_2) = (F, I, N)$, where $N = \{(i, n_i) \mid i \in I\}$.

$$
\begin{aligned}
& Q(G_1 \oplus H, S_1 \oplus S_H) \\
&= \forall_{v \in S_1 \oplus S_H} |N_{G_1 \oplus H, S_1 \oplus S_H}(v)| \leq p \\
&= \forall_{i \in I} |N_{H,S_H}(z_i)| + |N_{G_1,S_1}(x_i)| \Leftrightarrow |\{j \in I \mid x_j \in N_{G_1,S_1}(x_i) \wedge z_j \in N_{H,S_H}(z_i)\}| \leq p \\
&\quad \wedge \forall_{v \in S_1 - X} |N_{G_1 S_1}(v)| \leq p \\
&\quad \wedge \forall_{v \in S_H - Z} |N_{H,S_H}(v)| \leq p \\
&= \forall_{i \in I} |N_{H,S_H}(z_i)| + |n_i| \Leftrightarrow |\{j \in I \mid \{i, j\} \in F \wedge \{z_i, z_j\} \in E(H)\}| \leq p \\
&\quad \wedge \forall_{v \in S_1 - X} |N_{G_1,S_1}(v)| \leq p \\
&\quad \wedge \forall_{v \in S_H - Z} |N_{H,S_H}(v)| \leq p \\
&= \forall_{i \in I} |N_{H,S_H}(z_i)| + |N_{G,S_2}(y_i)| \Leftrightarrow |\{j \in I \mid y_j \in N_{G_2,S_2}(y_i) \wedge z_j \in N_{H,S_H}(z_i)\}| \leq p \\
&\quad \wedge \forall_{v \in S_2 - Y} |N_{G_2,S_2}(v)| \leq p \\
&\quad \wedge \forall_{v \in S_H - Z} |N_{H,S_H}(v)| \leq p \\
&= Q(G_2 \oplus H, S_2 \oplus S_H)
\end{aligned}
$$

Hence $\sim_{rQ,l}$ is a refinement of $\sim_{Q,l}$.

We now show that d_l is well defined for all $l \geq 0$. Let $c, c' \in C_{rQ,l}$, such that c and c' are compatible. Let $I \subseteq I_l$ such that $c = (I, \text{false})$ or $c = (F, I, N)$ for some F and N, and $c' = (I, \text{false})$ or $c' = (F', I, N')$ for some F' and N'. Let G and H be l-terminal graphs, let $S \in D_{[]}(G)$ and $S' \in D_{[]}(H)$ such that $ec_{rQ,l}(G, S) = c$ and $ec_{rQ,l}(H, S') = c'$. Then $|S \oplus S'| = |S \cup S'| = |S| + |S'| \Leftrightarrow |I|$, hence $d_l(c, c') = |I|$.

We now define K_l and c_G for all $l \geq 0$ and all l-terminal graphs G, as in condition 2(d) of Theorem 4.4. For each $l \geq 0$, let $K_l = l$, and for each l-terminal graph G, let $c_G = (F(G), \emptyset, \emptyset)$. Clearly, for each l-terminal graphs G and H, each $S \in D_{[]}(G)$ and $S' \in D_{[]}(H)$, if $ec_{rQ,l}(G, S) = c_G$ and $ec_{rQ,l}(H, S') = c_H$, then (G, S) and (H, S') are $\oplus$-compatible, and $Q(G \oplus H, S \oplus S')$ holds. Furthermore, for each l-terminal graph $G = (V, E, X)$, and each $S \in D_{[]}(G)$ that can lead to a solution (i.e. $ec_{rQ,l}(G, S) \neq (F(G), \text{false})$), $ec_{rQ,l}(G, S \Leftrightarrow X) = c_G$ and $|S| \Leftrightarrow \max(G, c_G) \leq |S| \Leftrightarrow |S \Leftrightarrow X| \leq l = K_l$.

This proves that MAXIMUM DEGREE BOUNDED SUBGRAPH is of finite integer index for all fixed $p \geq 0$.

2 p-DOMINATING SET. Let $p \geq 1$ be fixed. This proof is similar to the previous one. $D(G) = \mathcal{P}(V)$, for all l-terminal graphs G, $S \in D(G)$,

$$Q(G, S) = \forall_{v \in V-S} |N_{G,S}(v)| \geq p,$$

$z(S) = |S|$, and $\mathrm{opt} = \min$. $[\,]$ is defined in the same way as for INDUCED BOUNDED DEGREE SUBGRAPH, and hence so are $\oplus$ and ($\oplus$-)compatibility. For each $l \geq 0$, let

$$\begin{aligned} C_{rQ,l} \;=\; & \{(I, \mathrm{false}) \mid I \subseteq I_l\} \cup \{(F, I, N) \mid F \subseteq F_l \\ & \wedge\, I \subseteq I_l \wedge N \subseteq \{(i, n) \mid i \in I_l \Leftrightarrow I \wedge n \in \{1, \ldots, p\}\}. \end{aligned}$$

For each l-terminal graph $G = (V, E, X = \langle x_1, \ldots, x_l \rangle)$, each $S \in D_{[\,]}(G)$, let $ec_{rQ,l}(G, S) \in C_{rQ,l}$ be defined as follows. If there is a $v \in V \Leftrightarrow X$ such that $|N_{G,S}(v)| < p$, then $ec_{rQ,l}(G, S) = (F(G), \mathrm{false})$. Otherwise, $ec_{rQ,l}(G, S) = (F, I, N)$, where

$$\begin{aligned} F &= F(G), \\ I &= \{i \in I_l \mid x_i \in S\}, \\ N &= \{(i, |N_{G,S}(x_i)|) \mid i \in I \Leftrightarrow I_l\}. \end{aligned}$$

In the same way as for INDUCED BOUNDED DEGREE SUBGRAPH it can be shown that $\sim_{rQ,l}$ is a refinement of $\sim_{Q,l}$.

For each $l \geq 0$, d_l is defined in the same way as for INDUCED BOUNDED DEGREE SUBGRAPH, since d_l only depends on D.

We show again that condition 2(d) of Theorem 4.4 holds. For each $l \geq 0$, let $K_l = l$, and for each l-terminal graph G, let $c_G = (F(G), I_l, \emptyset)$. Clearly, for all l-terminal graphs G and H and $S \in D_{[\,]}(G)$, $S' \in D_{[\,]}(H)$ such that $ec_{rQ,l}(G, S) = c_G$ and $ec_{rQ,l}(H, S') = c_H$, $Q(G \oplus H, S \oplus S')$ holds. Furthermore, for each l-terminal graph $G = (V, E, X)$, each $S \in D_{[\,]}(G)$, if $ec_{rQ,l}(G, S) \neq (F(G), \mathrm{false})$, then $ec_{rQ,l}(G, S \cup X) = c_G$, and $\min(G, c_G) \Leftrightarrow |S| \leq |S \cup X| \Leftrightarrow |S| \leq l = K_l$.

3 MAXIMUM CUT on graphs with bounded degree. Let $d \geq 0$ be the maximum degree of the graphs. For each graph G, let $D(G)$ be the set of pairs (V_1, V_2), such that V_1 and V_2 partition V. For each two l-terminal graphs G and H, and $S = (V_1, V_2) \in D(G \oplus H)$, let $S[G] = (V_1 \cap V(G), V_2 \cap V(G))$. Note that $D_{[\,]}(G) = D(G)$, and that D is inducible for $[\,]$. Two pairs (G, S) and (H, S) are ($\oplus$-)compatible if S and S' partition the terminals of G and H in the same way.

For each graph G, each $S = (V_1, V_2) \in D(G)$, let $Q(G, S) = \mathrm{true}$, let

$$z(S) = |\{\{u, v\} \in E(G) \mid u \in V_1 \wedge v \in V_2\}|,$$

and let $\mathrm{opt} = \max$. Let z on the domain of partial solutions be defined in the same way as on the domain of solutions.

For each $l \geq 0$, let

$$C_{rQ,l} = \{(F, (I, I_l \Leftrightarrow I)) \mid F \subseteq F_l \wedge I \subseteq I_l\},$$

and for each l-terminal graph $G = (V, E, X = \langle x_1, \ldots, x_l \rangle)$, $S = (V_1, V_2) \in D_{[]}(G)$, let $ec_{rQ,l}(G, S) = (F, (I, I_l \Leftrightarrow I)) \in C_{rQ,l}$, where

$$\begin{aligned} F &= F(G), \\ I &= \{i \mid x_i \in V_1\}. \end{aligned}$$

Let G_1 and G_2 be l-terminal graphs, $S_1 \in D_{[]}(G_1)$, $S_2 \in D_{[]}(G_2)$. $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$ if and only if $ec_{rQ,l}(G_1, S_1) = ec_{rQ,l}(G_2, S_2)$. If $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$, then (G_1, S_1) and (G_2, S_2) are compatible, and hence $\sim_{rQ,l}$ is a refinement of $\sim_{Q,l}$.

We now define d_l and K_l for each $l \geq 0$, and c_G for all graphs G.

Let G and H be l-terminal graphs, let $S = (V_1, V_2) \in D_{[]}(G)$ and $S' = (W_1, W_2) \in D_{[]}(H)$, such that (G, S) and (H, S') are $\oplus$-compatible. Let $ec_{rQ,l}(G, S) = (F, (I, J))$, and let $ec_{rQ,l}(H, S) = (F', (I, J))$. Then

$$\begin{aligned} z(S \oplus S') &= |\{\{u, v\} \in E(G \oplus H) \mid u \in V_1 \cup W_1 \wedge v \in V_2 \cup W_2\}| \\ &= |\{\{u, v\} \in E(G) \mid u \in V_1 \wedge v \in V_2\}| \\ &\quad + |\{\{u, v\} \in E(H) \mid u \in W_1 \wedge v \in W_2\}| \\ &\quad \Leftrightarrow |\{\{u, v\} \in E(G) \cap E(H) \mid u \in W_1 \wedge v \in W_2\}| \\ &= z(S) + z(S') \Leftrightarrow |\{\{i, j\} \in F \cap F' \mid i \in I \wedge j \in J\}|. \end{aligned}$$

Hence $d_l((F, (I, J)), (F', (I, J))) = |\{\{i, j\} \in F \cap F' \mid i \in I \wedge j \in J\}|$.

We now show that condition 2(c) of Theorem 4.3 holds. For each $l \geq 0$, let $K_l = 2 \cdot l \cdot d$. Let $G = (V, E, X)$ be an l-terminal graph, let $c = (F(G), (I, J))$ and let $c' = (F(G), (I', J'))$. We have to show that $|\max(G, c) \Leftrightarrow \max(G, c')| \leq K_l$. Let $S = (V_1, V_2) = \mathrm{maxS}(G, c)$. Let $S' = (W_1, W_2)$, where

$$\begin{aligned} W_1 &= (V_1 \Leftrightarrow X) \cup \{x_i \mid i \in I'\} \\ W_2 &= (V_2 \Leftrightarrow X) \cup \{x_i \mid i \in J'\}. \end{aligned}$$

Then $ec_{rQ,l}(G, S') = c'$ and furthermore,

$$\begin{aligned} \max(G, c) \Leftrightarrow \max(G, c') &\leq z(S) \Leftrightarrow z(S') \\ &= z(S) \Leftrightarrow z(S) \\ &\quad \Leftrightarrow |\{\{u, v\} \in E(G) \mid u \in V_1 \cap W_1 \wedge v \in V_2 \cap W_1\}| \\ &\quad \Leftrightarrow |\{\{u, v\} \in E(G) \mid u \in V_1 \cap W_2 \wedge v \in V_2 \cap W_2\}| \\ &\leq 2 \cdot l \cdot d = K_l. \end{aligned}$$

Because of symmetry, this means that $|\max(G, c) \Leftrightarrow \max(G, c')| \leq K_l$. Hence MAXIMUM CUT is of finite integer index on graphs with bounded degree.

4 PARTITION INTO CLIQUES. For each graph G, let $D(G)$ be the set of all partitions $S = \{V_1, \ldots, V_s\}$ of $V(G)$ for which each $V_i \in S$ induces a connected subgraph of G. For each $S \in D(G)$, let

$$Q(G, S) = \forall_{V' \in S}\ G[V'] \text{ is a complete graph,}$$

let $z(S) = |S|$, and let $\mathrm{opt} = \min$.

For each l-terminal graphs $G = (V, E, X)$ and $H = (V', E', Y)$, each $S \in D(G \oplus H)$, let

$$S[G] = \{V'' \cap V(G) \mid V'' \cap V(G) \neq \emptyset\},$$

and let $z(S[G]) = |S[G]|$. Hence $D_{[]}(G)$ is the set of all partitions S in G in which for each $V' \in S$, all connected components of $G[V']$ have at least one vertex in X.

Note that D is inducible for $[\,]$, since, for an $S \in D(G \oplus H)$, it is not possible that there is a $V'' \in S$ such that both $V'' \cap V(G) \Leftrightarrow X \neq \emptyset$ and $V'' \cap V(H) \Leftrightarrow Y \neq \emptyset$, while $S \cap X = \emptyset$. Two pairs (G, S) and (H, S') are ($\oplus$-)compatible if the terminals of G and H are partitioned in the same way in S and S'.

For each $l \geq 0$, let

$$\begin{aligned} C_{rQ,l} \;=\; & \{(F, \mathrm{false}) \mid F \subseteq F_l\} \\ & \cup \{(F, \mathcal{J}) \mid F \subseteq F_l \wedge \\ & \mathcal{J} = \{(J_1, b_1), \ldots, (J_t, b_t)\}) \mid t \geq 1 \wedge \\ & \{J_1, \ldots, J_t\} \text{ partitions } I_l \wedge \forall_i\, J_i \neq \emptyset \wedge b_i \in \{\mathrm{true}, \mathrm{false}\}\}\} \end{aligned}$$

For each l-terminal graph $G = (V, E, \langle x_1, \ldots, x_l\rangle)$, each $S \in D_{[]}(G)$, let $ec_{rQ,l}(G, S) \in C_{rQ,l}$ be defined as follows. If there are $V' \in S$ and $v, w \in V'$, $v \neq w$, such that $v \notin X$ and $\{v, w\} \notin E$, then $ec_{rQ,l}(G, S) = (F(G), \mathrm{false})$. Otherwise, $ec_{rQ,l}(G, S) = (F, \mathcal{J})$, where

$$\begin{aligned} F \;&=\; F(G), \\ \mathcal{J} \;&=\; \{(J, b) \mid (\exists_{V' \in S}\, J = \{i \in I_l \mid x_i \in V'\} \wedge J \neq \emptyset \wedge b \Leftrightarrow (V' \subseteq X))\}. \end{aligned}$$

(b is a Boolean variable in the definition above.) For each $l \geq 0$, let $\sim_{rQ,l}$ be defined as usually. It is fairly easy to check that if $ec_{rQ,l}(G_1, S_1) = ec_{rQ,l}(G_2, S_2)$, then $(G_1, S_1) \sim_{Q,l} (G_2, S_2)$.

We now define d_l for each $l \geq 0$. Let G and H be l-terminal graphs, let $S \in D_{[]}(G)$, $S' \in D_{[]}(H)$, such that (G, S) and (H, S') are $\oplus$-compatible. Let $ec_{rQ,l}(G, S) = (F, \mathcal{J})$, and let $ec_{rQ,l}(H, S) = (F', \mathcal{J}')$. Then

$$\begin{aligned} |S| + |S'| \Leftrightarrow |S \oplus S'| \;&=\; |\{V' \in S \oplus S' \mid V' \cap X \neq \emptyset\}| \\ &=\; |\mathcal{J}|. \end{aligned}$$

Hence $d_l((F, \mathcal{J}), (F', \mathcal{J}')) = |\mathcal{J}|$.

We now show that condition 2(d) of Theorem 4.4 holds. For each $l \geq 0$, let $K_l = l$, and for each l-terminal graph G, let

$$c_G = (F(G), \{(i, \mathrm{true}) \mid 1 \leq i \leq l\}).$$

Let G be an l-terminal graph, let $S \in D_{[]}(G)$, such that $ec_{rQ,l}(G, S) \neq \mathrm{false}$. Furthermore, let

$$S' = \{\{v\} \mid v \in X\} \cup \{V' \Leftrightarrow X \mid V' \in S \wedge V' \not\subseteq X\}.$$

Then $S' \in D_{[]}(G)$ and $ec_{rQ,l}(G, S') = c_G$, and hence $\min(G, c_G) \Leftrightarrow |S| \leq |S'| \Leftrightarrow |S| \leq l = K_l$. Hence PARTITION INTO CLIQUES is of finite integer index.

5 HAMILTONIAN PATH COMPLETION NUMBER. A path P in a graph G is a subgraph of G, denoted as a sequence $(v_1, \ldots, v_s)$, such that for each i, $1 \leq i < s$, $\{v_i, v_{i+1}\} \in E(G)$. A path $P = (v_1, \ldots, v_s)$ is non-empty if $s \geq 1$. The vertices of a path P are denoted by $V(P)$.

For each graph G, let each element S in $D(G)$ be a set of non-empty paths in G, such that the set $\{V(P) \mid P \in S\}$ partitions G. For each $S \in D(G)$, let $Q(G, S) = \mathrm{true}$, and let $z(S) = |S| \Leftrightarrow 1$. Furthermore, let $\mathrm{opt} = \min$. Note that this describes the problem HAMILTONIAN PATH COMPLETION NUMBER.

Let G and H be l-terminal graphs, $S \in D(G \oplus H)$, $S = \{P_1, \ldots, P_m\}$. Let

$$S[G] = \bigcup_{i=1}^{m} \{P_i' \mid P_i' \text{ is a component of } P_i \cap G\},$$

and let $z(S[G]) = |S[G]| \Leftrightarrow 1$, i.e. $S[G]$ is the set of paths in G which is obtained from S by deleting all vertices and edges which are not in G, and deleting empty paths, and $z(S[G])$ is the number of these paths minus 1. Note that D is inducible for this definition of $[\,]$.

Let $G = (V, E, X = \langle x_1, \ldots, x_l \rangle)$ be an l-terminal graph G, let $P = (v_1, \ldots, v_s)$ be a path in G. Suppose $V(P) \cap X = \{x_{i_1}, \ldots, x_{i_q}\}$, $q \geq 1$, and for each $1 \leq j < m \leq q$, x_{i_j} occurs on the left side of x_{i_m} in P (i.e. by walking from v_1 to v_s in P, we meet x_{i_j} earlier than x_{i_m}). Then $\mathrm{Ind}(P)$ is defined as follows.

$$\mathrm{Ind}(P) = \begin{cases} (i_1, i_2, \ldots, i_q) & \text{if } x_{i_1} = v_1 \wedge x_{i_q} = v_s \\ (\mathrm{d}, i_1, i_2, \ldots, i_q) & \text{if } x_{i_1} \neq v_1 \wedge x_{i_q} = v_s \\ (i_1, i_2, \ldots, i_q, \mathrm{d}) & \text{if } x_{i_1} = v_1 \wedge x_{i_q} \neq v_s \\ (\mathrm{d}, i_1, i_2, \ldots, i_q, \mathrm{d}) & \text{if } x_{i_1} \neq v_1 \wedge x_{i_q} \neq v_s. \end{cases}$$

(d denotes the 'dummy' vertex.)

For each l-terminal graph G, each $S \in D_{[\,]}(G)$, let $ec_{rQ,l}(G, S)$ be defined as follows.

$$ec_{rQ,l}(G, S) = \{\mathrm{Ind}(P) \mid P \in S \wedge V(P) \cap X \neq \emptyset\}$$

For each $l \geq 0$, let $C_{rQ,l}$ and $\sim_{rQ,l}$ be defined as usual. Note that if $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$, then (G_1, S_1) and (G_2, S_2) are compatible, and hence $(G_1, S_1) \sim_{Q,l} (G_2, S_2)$.

Let $G = (V, E, X)$ and $H = (V', E', Y)$ be l-terminal graphs, $S \in D_{[\,]}(G)$ and $S' \in D_{[\,]}(H)$. If (G, S) and (H, S') are $\oplus$-compatible, then

$$\begin{aligned} z(S) + z(S') \Leftrightarrow z(S \oplus S') \;=\; & |\{P \in S \mid V(P) \cap X \neq \emptyset\}| \\ & + |\{P \in S' \mid V(P) \cap Y \neq \emptyset\}| \\ & \Leftrightarrow |\{P \in S \oplus S' \mid V(P) \cap X \neq \emptyset\}|. \end{aligned}$$

This value can be computed from $ec_{rQ,l}(G, S)$ and $ec_{rQ,l}(H, S)$, hence d_l is well defined.

We now define K_l for each $l \geq 0$ and c_G for each terminal graph G, and show that they satisfy condition 2(d) of Theorem 4.4. For each $l \geq 0$, let $K_l = 2l$, and for each l-terminal graph G, let

$$c_G = \{(i) \mid 1 \leq i \leq l\}.$$

Clearly, if $ec_{rQ,l}(G,S) = c_G$ and $ec_{rQ,l}(H,S') = c_H$, then (G,S) and (H,S') are $\oplus$-compatible.

Let G be an l-terminal graph, $S \in D_{[]}(G)$. Let

$$S' = \{(v) \mid v \in X\} \cup S[V(G) \Leftrightarrow X].$$

Then $S' \in D_{[]}(G)$, and $ec_{rQ,l}(G,S') = c_G$. Hence $\min(G, c_G) \Leftrightarrow z(S) \leq |S'| \Leftrightarrow |S| \leq l + |S| + l \Leftrightarrow |S| = 2l = K_l$.

This completes the proof that HAMILTONIAN PATH COMPLETION NUMBER is of finite integer index.

6 MAXIMUM LEAF SPANNING TREE. For each graph G, let $D(G)$ be the set of all spanning trees of G. For each $S \in D(G)$, let $Q(G,S) = \text{true}$, let $z(S) =$ the number of vertices of degree one in S, and let $\text{opt} = \max$.

For each l-terminal graphs G and H, each $S \in D(G \oplus H)$, let $S[G]$ be the set of trees in G obtained by deleting all vertices and edges from S which are not in G. Hence $D_{[]}(G)$ is the set of all spanning forests F of G for which each connected component of F contains at least one terminal of the graph. Note that D is inducible for $[\,]$. Let z on the domain of partial solutions be defined in the same way as on the domain of solutions.

For each $l \geq 0$, let

$$\begin{aligned} C_{rQ,l} &= \{(F, \text{false}) \mid F \subseteq F_l\} \cup \\ &\quad \{(F, \mathcal{I}, \{(i,s_i) \mid 1 \leq i \leq l \wedge s_i \in \{0,1,2\}) \mid F \subseteq F_l \wedge \mathcal{I} \text{ partitions } I_l\} \end{aligned}$$

For each l-terminal graph $G = (V, E, \langle x_1, \ldots, x_l \rangle)$, each $S \in D_{[]}(G)$, if S contains more than one connected component, and one of these components does not contain a terminal, then let $ec_{rQ,l} = (F, \text{false})$ (there is no H such that $G \oplus H$ contains a spanning tree), otherwise, let $ec_{rQ,l}(G,S) = (F, \mathcal{I}, A)$, where

$$\begin{aligned} F &= F(G), \\ \mathcal{I} &= \{J \mid \exists_{V' \subseteq V}\ V' \text{ is a connected component of } S \wedge J = \{i \in I_l \mid x_i \in V'\} \wedge J \neq \emptyset\} \\ A &= \{(i, s_i) \mid 1 \leq i \leq l \wedge s_i = |N_{S,V}(x_i)| \text{ if } |N_{S,V}(x_i)| \leq 2, \text{, otherwise } s_i = 2\}. \end{aligned}$$

For each $l \geq 0$, let $\sim_{rQ,l}$ be defined as usually. It is fairly easy to check that if $(G_1, S_1) \sim_{rQ,l} (G_2, S_2)$, then $(G_1, S_1) \sim_{Q,l} (G_2, S_2)$.

We now show that d_l is well defined. Let G and H be l-terminal graphs, let $S \in D_{[]}(G)$, $S' \in D_{[]}(H)$, such that (G,S) and (H,S') are $\oplus$-compatible. Let $ec_{rQ,l}(G,S) = (F, \mathcal{I}, A)$, and let $ec_{rQ,l}(H,S) = (F', \mathcal{I}', A')$, where $A = \{(i, s_i) \mid 1 \leq i \leq l\}$ and $A' = \{(i, s'_i) \mid 1 \leq i \leq l\}$. Then

$$\begin{aligned} z(S) + z(S') \Leftrightarrow z(S \oplus S') = \\ |\{i \in I_l \mid |N_{S \oplus S', V(G) \cup V(H)}(x_i)| \geq 2 \wedge s_i = 1\}| \\ + |\{i \in I_l \mid |N_{S \oplus S', V(G) \oplus V(H)}(x_i)| \geq 2 \wedge s'_i = 1\}|. \end{aligned}$$

This value depends only on $(F, \mathcal{I}, A)$ and $(F', \mathcal{I}', A')$, hence d_l is well defined.

We now show that condition 2(b) of Theorem 4.2 holds. For each $l \geq 0$, let $K_l = 2l$, and for each terminal graph G, let $i_G = z(F)$, where F is a maximal spanning forest of G (i.e. the connected components of F are the connected components of G), and $z(F)$ is maximum. Let G be an l-terminal graph, such that each connected component of G contains a terminal. Let $S \in D_{[]}(G)$ and suppose that $|z(S) \Leftrightarrow i_G| > K_l$. First suppose that $z(S) > i_G + K_l$. Let S' be a maximal spanning forest of G such that S is a subgraph of S'. S' can be obtained from S by adding at most $l \Leftrightarrow 1$ edges, and hence $z(S') \geq z(S) \Leftrightarrow 2l$. But $i_G \geq z(S')$, hence $z(S) \leq i_G + K_l$, which gives a contradiction.

Suppose $z(S) < i_G \Leftrightarrow K_l$. Let H be an l-terminal graph, let $S' \in D_{[]}(H)$ such that (G, S) and (H, S') are $\oplus$-compatible. We show that $z(S \oplus S') < R(G \oplus H)$. Let F be a maximal spanning forest of G such that $z(F) = i_G$. Let G' be the subgraph of $G \oplus H$ with $V(G') = V(G \oplus H)$, and $E(G') = E(F) \cup E(S')$. The number of vertices of degree one in G' is at least $z(F) + z(S') \Leftrightarrow l$. G' can be modified into a spanning tree T of $G \oplus H$ by deleting a number of edges in G'. This does not decrease the number of vertices of degree one, since if a vertex has one incident edge, then this edge is not removed. Hence $z(T) \geq z(G') \geq z(F) + z(S') \Leftrightarrow l = i_G + z(S') \Leftrightarrow l > z(S) + z(S') + K_l \Leftrightarrow l \geq z(S) + z(S') \geq z(S \oplus S')$. This means that $z(S \oplus S') < R(G \oplus H)$.

Hence MAXIMUM LEAF SPANNING TREE is of finite integer index.

For each problem, we have given an explicit definition of $\sim_{rQ,l}$, hence $\sim_{rQ,l}$ is effectively decidable. Furthermore, z, s and i_G are effectively computable. Hence we can construct a finite, safe, complete and decreasing set of reduction-counter rules of R_k ($k \geq 1$), where R is one of the optimization problems of this theorem. □

Theorem 4.6. *The following problems are not of finite integer index:*

1. MAXIMUM CUT,
2. COVERING BY CLIQUES,
3. LONGEST PATH, *and*
4. LONGEST CYCLE.

Proof. Below, in each of the parts of the proof, R denotes the respective optimization problem.

1 MAXIMUM CUT. We give an infinite set $\mathcal{G}$ of two-terminal graphs such that for each G and G' in this set, if $G \neq G'$ then $G \not\sim_{R,2} G'$. For each $n \geq 2$, let G_n be a two-terminal graph which is defined as follows (see also Figure 3).

$$V(G_n) = X \cup A \cup B_n \cup C_n,$$

where all sets are disjoint, $X = \langle x_1, x_2 \rangle$ is the set of terminals, $A = \{a_1, a_2\}$, and B_n and C_n each contain n vertices.

$$\begin{aligned} E(G_n) \;=\; & \{\{x_1, a_1\}, \{x_2, a_2\}\} \\ & \cup \{\{a_i, v\} \mid 1 \leq i \leq 2 \wedge v \in B_n \cup C_n\} \\ & \cup \{\{x_2, b\} \mid b \in B_n\} \\ & \cup \{\{x_1, c\} \mid c \in C_n\}. \end{aligned}$$

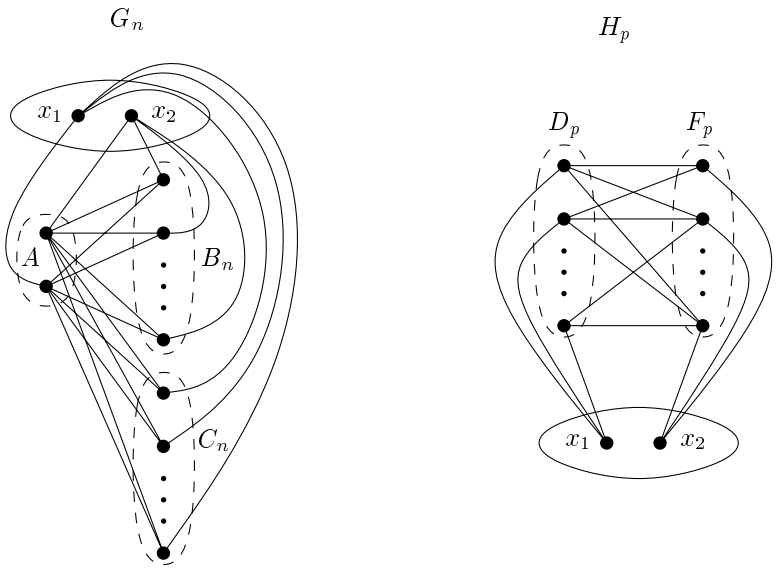

Figure 3: The graphs G_n ($n \geq 2$) and H_p ($p \geq 0$).

Let $\mathcal{G} = \{G_n \mid n \geq 2 \wedge n \text{ even}\}$.

Claim 4.1. Let $n \geq 1$, let H be a two-terminal graph. There is a partition (V_1, V_2) of $V = V(G_n \oplus H)$ such that $A \subseteq V_1$, $B_n \cup C_n \subseteq V_2$, and $z((V_1, V_2))$ is maximum.

Proof. Suppose (W_1, W_2) is a maximum cut for $G_n \oplus H$, let $M^* = z((W_1, W_2))$. Let $A_1 = W_1 \cap A$, $A_2 = W_2 \cap A$, $BC_1 = (B_n \cup C_n) \cap W_1$, and $BC_2 = (B_n \cup C_n) \cap W_2$. Let

$$\begin{aligned}(V_1, V_2) &= (W_1 \Leftrightarrow BC_1 \cup A_2, W_2 \Leftrightarrow A_2 \cup BC_1), \text{ and}\\ (V_1', V_2') &= (W_2 \Leftrightarrow BC_2 \cup A_1, W_1 \Leftrightarrow A_1 \cup BC_2).\end{aligned}$$

Then (V_1, V_2) and (V_1', V_2') are both cuts of $G_n \oplus H$, and $A \subseteq V_1$, $A \subseteq V_1'$, $B_n \cup C_n \subseteq V_2$ and $B_n \cup C_n \subseteq V_2'$. We now show that either (V_1, V_2) or (V_1', V_2') is a maximum cut. Let $M = z((V_1, V_2))$ and let $M' = z((V_1', V_2'))$. We consider two cases, namely

1. $|A_2| = 0 \vee |BC_1| = 0$, and
2. $0 < |A_2| \leq |A|$ and $0 < |BC_1| \leq |BC_1 \cup BC_2|$.

In case 1,

$$\begin{aligned}M &\geq M^* + |A_1| \cdot |BC_1| + |A_2| \cdot |BC_2| \Leftrightarrow |A_2| \Leftrightarrow |BC_1|\\ &= M^* + |A_2|(|BC_2| \Leftrightarrow 1) + |BC_1|(|A_1| \Leftrightarrow 1)\\ &\geq M^*.\end{aligned}$$

In case 2,

$$\begin{aligned} M' &\geq M^* + |A_1| \cdot |BC_1| + |A_2| \cdot |BC_2| \Leftrightarrow |A_1| \Leftrightarrow |BC_2| \\ &= M^* + |A_1|(|BC_1| \Leftrightarrow 1) + |BC_2|(|A_2| \Leftrightarrow 1) \\ &\geq M^*. \end{aligned}$$

This proves the claim. □

For each $p \geq 0$, let H_p be the graph defined as follows (see also Figure 3).

$$V(H_p) = Y \cup D_p \cup F_p,$$

where all sets are disjoint $Y = \langle y_1, y_2 \rangle$ is the set of terminals, and D_p and F_p each contain p vertices, and

$$\begin{aligned} E(H_p) &= \{\{d, f\} \mid d \in D_p \wedge f \in F_p\} \\ &\quad \cup \{\{y_1, d\} \mid d \in D_p\} \\ &\quad \cup \{\{y_2, f\} \mid f \in F_p\}. \end{aligned}$$

Claim 4.2. Let $p \geq 0$, let H be a two-terminal graph. There is a partition (V_1, V_2) of $V = V(H_p \oplus H)$ such that $D_p \subseteq V_1$, $F_p \subseteq V_2$, and $z((V_1, V_2))$ is maximum.

Proof. Similar to proof of Claim 4.1. □

We now show that for each $G_n, G_m \in \mathcal{G}$, if $n \neq m$, then $G_n \not\sim_{R,2} G_m$.

For $i \geq 2$, each $p \geq 0$, consider the graph $G_i \oplus H_p$: let $V_1^{(i,p)} = A \cup D_p$, let $V_2^{(i,p)} = B_i \cup C_i \cup F_p$. There is a maximum cut (W_1, W_2) of $G_i \oplus H_p$ such that $V_1^{(i,p)} \subseteq W_1$ and $V_2^{(i,p)} \subseteq W_2$, because of Claim 4.1, Claim 4.2, and the fact that H_p is symmetrical in D_p and F_p. In the following table, all cuts that are candidates for maximum cuts in $G_i \oplus H_p$ are given, together with there values.

nr.	cut	value
1	$(V_1^{(i,p)} \cup X, V_2^{(i,p)})$	$4i + p^2 + 2i + p$
2	$(V_1^{(i,p)} \cup \{x_1\}, V_2^{(i,p)} \cup \{x_2\})$	$4i + p^2 + i + 1$
3	$(V_1^{(i,p)} \cup \{x_2\}, V_2^{(i,p)} \cup \{x_1\})$	$4i + p^2 + i + 1 + 2p$
4	$(V_1^{(i,p)}, V_2^{(i,p)} \cup X)$	$4i + p^2 + 2 + p$

Note that either cut 1 or cut 3 is maximum, since $i \geq 2$, and $p \geq 0$.

Let $n > m > 1$, n, m even. If $p = 0$, then 1 is a maximum cut for both $G_n \oplus H_0$ and $G_m \oplus H_0$, since $i \geq 2$, which means that $2i > i + 1$. Hence $R(G_n \oplus H_0) = 6n$ and $R(G_m \oplus H_0) = 6m$, so $R(G_n \oplus H_0) \Leftrightarrow R(G_m \oplus H_0) = 6(n \Leftrightarrow m)$.

Let $p = \frac{1}{2}(n+m) \Leftrightarrow 1$. Then

$$\begin{aligned} R(G_n \oplus H_p) &= 4n + p^2 + \max\{2n + \frac{1}{2}(n+m) \Leftrightarrow 1, n+1+(n+m) \Leftrightarrow 2\} \\ &= 4n + p^2 + \max\{2\frac{1}{2}n + \frac{1}{2}m \Leftrightarrow 1, 2n+m \Leftrightarrow 1\} \\ &= 4n + p^2 + 2\frac{1}{2}n + \frac{1}{2}m \Leftrightarrow 1 \\ &= 6\frac{1}{2}n + \frac{1}{2}m + p^2 \Leftrightarrow 1, \end{aligned}$$

and

$$\begin{aligned} R(G_m \oplus H_p) &= 4m + p^2 + \max\{2m + \frac{1}{2}(n+m) \Leftrightarrow 1, m+1+(n+m) \Leftrightarrow 2\} \\ &= 4m + p^2 + \max\{2\frac{1}{2}m + \frac{1}{2}n \Leftrightarrow 1, 2m+n \Leftrightarrow 1\} \\ &= 4m + p^2 + 2m + n \Leftrightarrow 1 \\ &= 6m + n + p^2 \Leftrightarrow 1 \end{aligned}$$

Hence

$$\begin{aligned} R(G_n \oplus H_p) \Leftrightarrow R(G_m \oplus H_p) &= (6\frac{1}{2}n + \frac{1}{2}m + p^2 \Leftrightarrow 1) \Leftrightarrow (6m + n + p^2 \Leftrightarrow 1) \\ &= 5\frac{1}{2}(n \Leftrightarrow m) \end{aligned}$$

However, $5\frac{1}{2}(n \Leftrightarrow m) \neq 6(n \Leftrightarrow m) = R(G_n \oplus H_0) \Leftrightarrow R(G_m \oplus H_0)$, since $n \neq m$. So $G_n \not\sim_{R,2} G_m$. As each G_n, $n > 1$, n even, belongs to a different equivalence class of $\sim_{R,2}$, the MAXIMUM CUT problem is not of finite integer index.

2 COVERING BY CLIQUES. For each $n \geq 1$, let G_n be the two-terminal graph with vertex set

$$V(G_n) = \{x_1, x_2\} \cup \{a_1, \ldots, a_n\},$$

(x_1 and x_2 are the first and the second terminal, respectively), and edge set

$$E(G_n) = \{\{x_i, a_j\} \mid 1 \leq i \leq 2 \wedge 1 \leq j \leq n\}.$$

Let $\mathcal{G} = \{G_n \mid n \geq 1\}$. We show that for each $G_n, G_m \in \mathcal{G}$, if $n \neq m$, then $G_n \not\sim_{R,2} G_m$.

Let H be the two-terminal graph consisting of terminals y_1 and y_2 and no edges, and let H' be the two-terminal graph consisting of terminals y_1 and y_2 and edge $\{y_1, y_2\}$.

For each i, $i \geq 1$, $R(G_i \oplus H) = |\{e \mid e \in E(G_i)\}| = 2i$, since $G_i \oplus H$ contains no cliques of more than two vertices. Furthermore, $R(G_i \oplus H') = |\{\{x_1, x_2, a_j\} \mid 1 \leq j \leq n\}| = i$. This means that for all n and m, $n \neq m$,

$$R(G_n \oplus H) \Leftrightarrow R(G_m \oplus H) = 2n \Leftrightarrow 2m \neq n \Leftrightarrow m = R(G_n \oplus H') \Leftrightarrow R(G_m \oplus H'),$$

and hence $G_n \not\sim_{R,l} G_m$.

3 LONGEST PATH. For each $n \geq 1$, let G_n be the two-terminal graph with vertex set

$$V(G_n) = \{x_1, x_2\} \cup \{a_1, \ldots, a_n\},$$

(x_1 and x_2 are the first and the second terminal, respectively), and edge set

$$E(G_n) = \{\{x_1, a_1\}\} \cup \{\{a_i, a_{i+1}\} \mid 1 \leq i < n\}.$$

Let $\mathcal{G} = \{G_n \mid n \geq 1 \wedge n \text{ even}\}$. Furthermore, for each $p \geq 1$, let H_p be the two-terminal graph with vertex set

$$V(H_p) = \{y_1, y_2\} \cup \{b_1, \ldots, b_p\},$$

(y_1 and y_2 are the first and the second terminal, respectively), and edge set

$$E(G_n) = \{\{y_2, b_1\}\} \cup \{\{b_i, b_{i+1}\} \mid 1 \leq i < p\}.$$

For each $i \geq 1$, $j \geq 1$, $R(G_i \oplus H_j) = \max\{i, j\} + 1$.

Let $1 \leq n < m$, such that n and m are even. Then $R(G_n \oplus H_{n+1}) \Leftrightarrow R(G_m \oplus H_{n+1}) = n + 1 + 1 \Leftrightarrow (m + 1) = n \Leftrightarrow m + 1 < 0$. Furthermore, $R(G_n \oplus H_m) \Leftrightarrow R(G_m \oplus H_m) = m + 1 \Leftrightarrow (m + 1) = 0$. Hence $G_n \not\sim_{R,l} G_m$.

4 LONGEST CYCLE. For each $n \geq 1$, let G_n be the two-terminal graph with vertex set

$$V(G_n) = \{x_1, x_2\} \cup \{a_1, \ldots, a_n\},$$

(x_1 and x_2 are the first and the second terminal, respectively), and edge set

$$E(G_n) = \{\{x_1, a_1\}\} \cup \{\{x_1, a_n\}\} \cup \{\{a_i, a_{i+1}\} \mid 1 \leq i < n\}.$$

Let $\mathcal{G} = \{G_n \mid n \geq 1 \wedge n \text{ even}\}$. Furthermore, for each $p \geq 1$, let H_p be the two-terminal graph with vertex set

$$V(H_p) = \{y_1, y_2\} \cup \{b_1, \ldots, b_p\},$$

(y_1 and y_2 are the first and the second terminal, respectively), and edge set

$$E(G_n) = \{\{y_2, b_1\}\} \cup \{\{y_2, b_p\}\} \cup \{\{b_i, b_{i+1}\} \mid 1 \leq i < p\}.$$

The rest of the proof is similar to the proof that LONGEST PATH is not of finite integer index. □

5 Constructing Optimal Solutions

Let R be a graph optimization problem. If R can be written in the form

$$R(G) = \text{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\},$$

then we are often not only interested in the value of $R(G)$ for a given graph G, but also a solution $S \in D(G)$ for which $z(S) = R(G)$. In this section we show that we can combine the

results of Sections 3 and 4 to get a method with which we can make an efficient reduction algorithm for R, in which both $R(G)$ and an $S \in D(G)$ for which $Q(G, S)$ holds and $R(G) = z(S)$ are computed. We can then use a small modification of Algorithm ConstructSolution to do this: instead of using a finite, safe, complete and decreasing set of reduction rules, use a finite, safe, complete and decreasing set of reduction-counter rules. Then in lines 1 to 4 of Algorithm ConstructSolution, apply the reduction-counter rules as described in Section 4.

Theorem 5.1. *Let R be a graph optimization problem. Suppose R can be written in the form*

$$R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\},$$

where D is inducible for $[\,]$, Q is decidable, there is a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ which is decidable, $|C_{rQ,l}|$ is finite for each $l \geq 0$, conditions 2 and 3 of Theorem 3.1 hold, and conditions 2 and 3 of Theorem 4.2 hold.

Then for each $k \geq 1$, there exists a finite, safe, complete and terminating set $\mathcal{R}$ of reduction-counter rules for R_k, and an implementation of the modification of Algorithm ConstructSolution *which can be used to compute for each graph G, in linear time, the value $R_k(G)$, and if $R_k(G) \in \mathbb{Z}$, an $S \in D(G)$ such that $Q(G, S)$ holds and $z(S) = R(G)$.*

If, in addition, Q and $\sim_{rQ,l}$ are effectively decidable, z is effectively computable, the function s from condition 2 of Theorem 3.1 is effectively computable, and in condition 3 of Theorem 3.1, $S[H]$ and $S' \oplus S[G]$ are effectively computable from S, S', H and H', then we can construct $\mathcal{R}$ and the implementation of the modification of Algorithm ConstructSolution.

Proof. For each l-terminal graph G, let f_G be the function as defined in the proof of Theorem 4.2. Let $\sim_l$ be the equivalence relation on l-terminal graphs defined as follows. $G_1 \sim_l G_2 \Leftrightarrow f_{G_1} = f_{G_2}$. Theorem 4.1 shows that there is a finite, safe, complete and decreasing set of reduction-counter rules $\mathcal{R}$ for R, such that for each rule $((H, H'), i) \in \mathcal{R}$, $H \sim_l H'$. This set can be constructed if $\sim_{rQ,l}$ is effectively decidable and functions z and s are effectively computable.

In the same way as is shown in the proof of Theorem 3.1, for each reduction-counter rule $((H_1, H_2), i)$ in $\mathcal{R}$, keep a table T for H_1, which contains for each possible equivalence class $c \in C_{rQk,l}$, a partial solution $S_1 \in D_{[]}(H_1)$ such that $ec_{rQk,l}(H_1, S_1) = c$ and $z(S_1) = \mathrm{opt}(G_1, c)$, if $f_{G_1}(c) \neq \mathrm{false}$, false otherwise. This table can be constructed if $\sim_{rQ,l}$ is decidable, and functions s and z are effectively computable.

Let G be a graph. The modification of Algorithm ConstructSolution can now be further refined as follows. In line 8, an optimal $S \in D(G)$ (G is the reduced graph here) for which $Q(G, S)$ holds can be constructed as follows (if Q is effectively decidable, and z and s are effectively computable). Each possible $S \in D(G)$ is tried, and if $Q(G, S)$ holds, and $z(S)$ is optimal, then this solution is taken. Note that this can be done in constant time, see also the proof of Theorem 3.1.

In line 14 of Algorithm ConstructSolution, the construction of S' is done in the same way as is shown in the proof of Theorem 3.1. First $S[H'_i]$ is computed. Then $c = ec_{rQk,l}(H'_i, S[H'_i])$ is computed. After that $S'' = T(c)$ is obtained, and $S' = S'' \oplus S[H]$ is computed. In the proof of Theorem 4.2 it is shown that $f_{H'_i}(c) \neq \mathrm{false}$, and hence S'' exists. The fact that S' is an optimal solution in G', if S is an optimal solution in G, follows from the last part of the proof of Theorem 4.2. □

Theorem 5.1 can be applied to all problems of Theorem 4.5.

Theorem 5.2. *Let $R = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G,S)\}$ be one of the following graph optimization problems, with* opt*, z, D and Q as defined in the proof of Theorem 4.5:*

1. INDUCED BOUNDED DEGREE SUBGRAPH *for all $p \geq 0$,*
2. p-DOMINATING SET *for all $p \geq 1$,*
3. MAXIMUM CUT *on graphs with bounded degree,*
4. PARTITION INTO CLIQUES*,*
5. HAMILTONIAN PATH COMPLETION NUMBER*, and*
6. MAXIMUM LEAF SPANNING TREE*.*

For each $k \geq 1$, there is, and we can construct, a finite, safe, complete and decreasing set of reduction-counter rules for R_k and an implementation of the modification of Algorithm ConstructSolution *with which we can compute in linear time the value $R_k(G)$, and an $S \in D(G)$ for which $z(S) = R_k(G)$, if $R_k(G) \in \mathbb{Z}$.*

Proof. We only have to show for each problem that conditions 2 and 3 of Theorem 3.1 hold (in the 'effective' way). This can be done straightforwardly. □

6 Parallel Reduction Algorithms

It is possible to combine the results of Sections 2 up to 5 with results from [6] to obtain fast parallel algorithms for several problems on graphs with bounded treewidth.

A set of applications of reduction(-counter) rules is said to be *concurrent*, if there is no inner vertex of any subgraph to be rewritten that also occurs in another subgraph to be rewritten.

The idea behind concurrent applications of rules is that in a parallel algorithm, all reduction steps from a concurrent set can be carried out simultaneously. This is very useful in order to obtain fast parallel algorithms, based on reduction.

We use a result from [6] to show that there exists a finite, safe, complete, and decreasing set $\mathcal{R}$ of reduction rules for finite index properties P, such that in any graph G of treewidth at most k with more than a constant number of vertices, a set of $\Omega(n)$ concurrent reductions can be found.

Definition 6.1. *Suppose $G = (V,E) = H_1 \oplus H_2$, $H_1 = (V_1, E_1, X)$ open. Let d be a constant positive integer. We say that H_1 is* strongly connected *in G with respect to a fixed adjacency list representation of G and bound d, if for all $v, w \in V_1$, there is a path $(v = x_0, x_1, \ldots, x_r = w)$ in H_1 from v to w, such that for all i, $1 \leq i < r$, the edges $\{x_i, x_{i-1}\}$ and $\{x_i, x_{i+1}\}$ have distance at most d in the adjacency list of x_i.*

The following lemma is a weaker version of Lemma 5 in [6].

Lemma 6.1. *[Bodlaender, Hagerup [6]] For all integers $k, n_{\min} \geq 1$, there are integers $d, n_{\max} \geq 1$ and a real $c > 0$, such that every connected graph G with $n > n_{\max}$ vertices and treewidth at most k, with an arbitrary adjacency-list representation, contains at least cn strongly connected subgraphs which each have at most $n_{\max}$ inner vertices and at most $2k+1$ terminals, and no inner vertex of any of these subgraphs occurs in another subgraph.*

We now can prove the following result, which is a generalization of Theorem 2.1, geared towards parallel algorithms.

Theorem 6.1. *Let k be a constant, P a graph property that is of finite index. There exist a set of reduction rules $\mathcal{R}$ for P_k, and a constant d, that fulfil the following properties.*

- *$\mathcal{R}$ is finite, safe, complete, and decreasing for P_k.*
- *For each reduction rule $(H, H') \in \mathcal{R}$, H and H' are open, and if H has one or more terminals, then H is connected.*
- *There exist constants $n_{\max}$, $c > 0$, such that for any connected graph $G = (V, E)$ for which $P_k(G)$ holds or which is a connected component of a graph H with $P_k(H)$, either $|V| \leq n_{\max}$, or for any adjacency-list representation of G, there exists a set of at least $c|V|$ concurrent applications of rules from $\mathcal{R}$ in G, such that for each application, replacing a terminal subgraph G_1 by another terminal subgraph, G_1 is strongly connected in G with respect to this adjacency-list representation and d.*

If there is also an equivalence relation $\sim_l$ for each $l \geq 0$, which is a refinement of $\sim_{P,l}$, is effectively decidable, and has a finite number of equivalence classes, then such a set of reduction rules can be constructed.

Proof. We use the same approach as in the proof of Theorem 2.1. For every $l \leq 2(k+1)$, and every equivalence class c of $\sim_{P_k,l}$, we take representing open terminal graphs H_c, as in the proof of Theorem 2.1. Again, r is the maximum number of vertices of any such representing graph, over all equivalence classes of $\sim_{P_k,l}$, all $l \leq 2(k+1)$. Let $n_{\max}$, d, be as given by Lemma 6.1, with $n_{\min} = r + 1$.

For all zero-terminal graphs H with at least $r+1$ and at most $n_{\max}$ vertices, if we have a representative for the class c which contains H, then add reduction rule (H, H_c) to $\mathcal{R}$. For all l, $1 \leq l \leq 2(k+1)$, and for all open connected l-terminal graphs H with at least $r+1 = n_{\min}$ and at most $n_{\max} + l$ vertices, if we have a representative for the equivalence class c in which H is contained, then add the reduction rule (H, H_c) to $\mathcal{R}$.

As in the proof of Theorem 2.1, we can see that $\mathcal{R}$ fulfils the first two stated properties. Construction of the set can also be done as in the proof of Theorem 2.1.

Finally, we note that any connected graph G with more than $n_{\max}$ vertices such that $P_k(G)$ holds contains a set of $c|V|$ concurrent applications of rules from $\mathcal{R}$, each involving a strongly connected terminal subgraph: Lemma 6.1 states that there are $c|V|$ strongly connected subgraphs in G which have at most $n_{\max}$ inner vertices and at most $2k+1$ terminals, and of which the sets of inner vertices are pairwise disjoint. Each of these strongly connected subgraphs can be taken as left-hand-side in a rule application (because of its size), and hence the rule applications are concurrent. □

Theorem 6.1 allows us to use the method from [6] to obtain fast parallel algorithms, based on graph reduction. The basic idea is the following: each vertex can have a processor find all $O(1)$ vertices to which it has a path of distance at most d, such that any two consecutive edges $\{x_{i-1}, x_i\}$ and $\{x_i, x_{i+1}\}$ have distance at most d in the adjacency list of x_i. Then, the processor looks for a possible occurrence of a left-hand-side of a rule application in the subgraph, just discovered. Each such occurrence gives a possible rule application. By building a *conflict graph*, where applications that are not concurrent correspond to adjacent vertices, and finding an independent set in the conflict graph, a concurrent set of applications is found. After $O(\log n)$ parallel reduction rounds, G is reduced to a collection of connected components, each of size at most $n_{\max}$. By repeatedly, in parallel, grouping these sets of size between $n_{\max}+1$ and $2n_{\max}$, and reducing each group to a graph of size at most $n_{\max}$, we end up with a reduced graph G of constant size, after $O(\log n)$ rounds. For more details, we refer to [6].

Moreover, the approach from Section 3 can be used to construct solutions. Reductions are then undone in parallel, in reverse order. By using proper bookkeeping, we can make sure that a reduction is undone by the same processor that carried out the reduction. Thus, we increase the time by not more than a constant factor, and use the same number of processors.

We denote the product of the number of processors, and the time used by a parallel algorithm, as *the number of operations* of the algorithm. The techniques from [6], combined with the results of this paper, give the following results.

Theorem 6.2. *Let k be a constant, P a finite index graph property.*
(i) The problem whether for a given graph G, $P_k(G) = P(G) \wedge \mathrm{tw}(G) \leq k$ holds can be solved on an EREW PRAM using $O(\log n \log^* n)$ time, $O(n)$ operations, and $O(n)$ space, and on a CRCW PRAM using $O(\log n)$ time, $O(n)$ operations, and $O(n)$ space. If a refinement $\sim_l$ of $\sim_{P,l}$ with a finite number of equivalence classes is effectively decidable, then the algorithms can be constructed.
(ii) Suppose that P can be written in the form

$$P(G) = \exists_{S \in D_1(G) \times \cdots \times D_t(G)} Q(G, S),$$

in such a way that for each G and i, $D_i(G)$ is equal to $V(G)$, $E(G)$, $\mathcal{P}(V(G))$ or $\mathcal{P}(E(G))$. Then, the problem, given a graph G, to construct an $S \in D(G)$ with $Q(G, S)$ if $P_k(G)$ holds, can be solved on an EREW PRAM using $O(\log n \log^* n)$ time, $O(n)$ operations, and $O(n)$ space, and on a CRCW PRAM using $O(\log n)$ time, $O(n)$ operations, and $O(n)$ space. If Q is effectively decidable, a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ is effectively decidable, and $|C_{rQ,l}|$ is finite, then the algorithms can be constructed. These last conditions hold when a definition of Q in monadic second order logic is given.

In particular, Theorem 6.2 shows that many well known graph problems, including k-COLORABILITY for fixed k, HAMILTONIAN CIRCUIT, etc., when restricted to graphs of bounded treewidth, can be solved constructively in the stated resource bounds.

A similar approach can be taken for problems that are of finite integer index. In addition to what is written above, each processor p owns an integer variable α_p, which is initially 0. For each reduction-counter rule $((H, H'), i)$ carried out, we let one processor p that is involved in carrying out the reduction, add i to its integer variable α_p. Suppose we have rewritten input

graph G to a graph G' of constant bounded size. If $R(G')$ is false, then $R(G)$ is false. Otherwise, we add $R(G')$ with the sum of all values α_p, over all processors p. This sum can easily be computed in $O(\log n)$ time with $O(n)$ operations and space on an EREW PRAM. The obtained value equals $R(G)$. In all other aspects, the method is the same as the one used for finite index problems.

Theorem 6.3. *Let k be a constant, R a graph optimization problem which is of finite integer index. The problem to compute for a given graph G the value $R_k(G)$ can be solved on an EREW PRAM using $O(\log n \log^* n)$ time, $O(n)$ operations, and $O(n)$ space, and on a CRCW PRAM using $O(\log n)$ time, $O(n)$ operations, and $O(n)$ space.*

If a refinement $\sim_l$ of $\sim_{R,l}$ with a finite number of equivalence classes is effectively decidable, and for each pair H, H', if $H \not\sim_l H'$, then we can effectively compute an $i \in \mathbb{Z}$ such that $((H, H'), i)$ is safe for R_k, then the algorithms can be constructed.

Theorem 6.4. *Let R be a graph optimization problem. Suppose R can be written in the form*

$$R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\},$$

where D is inducible for $[\,]$, Q is decidable, there is a refinement $\sim_{rQ,l}$ of $\sim_{Q,l}$ which is decidable, $|C_{rQ,l}|$ is finite for each $l \geq 0$, conditions 2 and 3 of Theorem 3.1 hold, and conditions 2 and 3 of Theorem 4.2 hold.

Then, for each $k \geq 1$, there exists algorithms, that compute for each G the value $R_k(G)$, and if $R_k(G) \in \mathbb{Z}$, an $S \in D(G)$ such that $Q(G, S)$ holds and $z(S) = R(G)$, using on an EREW PRAM $O(\log n \log^* n)$ time, $O(n)$ operations, and $O(n)$ space, and using on a CRCW PRAM $O(\log n)$ time, $O(n)$ operations, and $O(n)$ space.

If, in addition, Q and $\sim_{rQ,l}$ are effectively decidable, z is effectively computable, the function s from condition 2 of Theorem 3.1 is effectively computable, and in condition 3 of Theorem 3.1, $S[H]$ and $S' \oplus S[G]$ are effectively computable from S and H, then we can construct the algorithms.

This implies parallel algorithms with the stated resource bounds for (the constructive versions of) INDUCED BOUNDED DEGREE SUBGRAPH for all $p \geq 0$, p-DOMINATING SET for all $p \geq 1$, MAXIMUM CUT on graphs with bounded degree, PARTITION INTO CLIQUES, HAMILTONIAN PATH COMPLETION NUMBER, and MAXIMUM LEAF SPANNING TREE when restricted to graphs of bounded treewidth.

7 Conclusions and Further Research

In this paper, we have shown that reduction algorithms as introduced by Arnborg et al. [2] can not only be used to decide whether a graph property holds for a given graph with bounded treewidth, but in many cases, they can also be used to give a solution if one exists, to solve optimization problems on graphs with bounded treewidth, and to construct optimal solutions for graph optimization problems.

The reduction algorithms of Arnborg et al. use a linear amount of time, but a polynomial amount of space. We have shown that the techniques from [6] can be used to run the reduction algorithms in $O(\log n \log^* n)$ time on an EREW PRAM with $O(n)$ operations and $O(n)$

space, and in $O(\log n)$ time on a CRCW PRAM with $O(n)$ operations and space (n is the number of vertices of the graph). Sequential implementations of these algorithms give linear time algorithms, that use linear space (and the standard pointer-machine model).

We have shown that the reduction algorithms for deciding graph properties and constructing a solution can be applied to the class of graph properties that are definable in monadic second-order logic, and for which the solution space consists of tuples of vertices, edges, vertex-sets and edge-sets. This class includes problems like k-COLORABILITY (for fixed k) and HAMILTONIAN CIRCUIT. Unfortunately, it seems that this method can not be used to find tree or path decompositions of a graph with small treewidth, since we do not know whether the solution space for these problems can be represented in the right form. It is an interesting open problem whether our approach can be extended such that tree and path decompositions for graphs with small treewidth can be found.

For graph optimization problems, we have given a method to prove that for a given graph optimization problem a reduction algorithm can be used, and we have proved for a number of problems that this works, e.g. INDEPENDENT SET, PARTITION INTO CLIQUES. For all these problems, optimal solutions can be constructed. An interesting problem would be to find general characterizations of large classes of graph optimization problems for which our method can be used.

It is also possible to generalize the results in the paper to directed, mixed and/or labeled graphs. In case of labeled graphs, we can allow the graph to be given together with a labeling of the vertices and/or edges with labels from a set of size, bounded by a constant. These labels could also act as weights for finite integer index problems, e.g., we can deal with WEIGHTED INDEPENDENT SET, with each vertex a weight from $\{1, 2, \ldots, C\}$ for some fixed C, in the same way as we dealt with INDEPENDENT SET. The desired output of the algorithm can also be a direction given to each edge of the graph, such that the directed variant of the input graph fulfils a certain (MS-definable or otherwise finite state) property. Each of these generalizations can be handeled in a very similar way as the results, given in this paper.

The results of this paper can also be used to give algorithms that generate all solutions for a graph property P of the form $P(G) = \exists_{S \in D(G)} Q(G, S)$, or all optimal solutions for a graph optimization problem R of the form $R(G) = \mathrm{opt}\{z(S) \mid S \in D(G) \wedge Q(G, S)\}$. For these cases, algorithm ConstructSolution can be modified as follows. The table T that is kept for each left-hand-side H of a reduction(-counter) rule, as described in the proofs of Theorem 3.1 and Theorem 5.1, contains for each equivalence class c a list of all (optimal) $S \in D_{[]}(H)$ with $ec_{rQ,l}(H, S) = c$. In Line 8, all possible (optimal) $S \in D_{[]}(G)$ are constructed for which $Q(G, S)$ holds. In Line 14, for each (optimal) solution S of G, the following is done. First, $S' = S[H'_i]$ and $c = ec_{rQk,l}(H'_i, S[H'_i])$ are computed. Then the list of all partial solutions $S'' \in T(c)$ is obtained, suppose there are m such partial solutions. Then m copies are made of S, and each copy of S is changed into $S'' \oplus S[H]$ for some $S'' \in T(c)$. Note that this algorithm runs in $O(n + s)$ time, where n is the number of vertices of the input graph, and s is the amount of space needed to store all solutions for the input graph.

Graph reduction may possibly also be a useful tool which helps to solve hard problems on arbitrary (sparse) graphs. A possible approach to many such problems could be the following.

- Fix some safe and decreasing set of reduction or reduction-counter rules.

- Apply reduction rules on the input graph and reduced versions, until no rule application is possible.
- Use another method to solve the problem on the reduced graph.
- Possibly, construct a solution for the original problem from the solution for the reduced problem by using the method of Section 3.

The hope is, of course, that the graph after the reductions is substantially smaller than the original graph, and hence, that the running time of the third step is much smaller than the running time when this algorithm would have been applied directly to the input graph. Provided that the reductions can be carried out quick enough, this approach may be a nice tool to reduce the time needed to solve some graph problems on arbitrary sparse graphs.

References

[1] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In *Proceedings of the AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, 1993.

[3] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12:308–340, 1991.

[4] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[5] H. L. Bodlaender. On reduction algorithms for graphs with small treewidth. In *Proceedings 19th International Workshop on Graph-Theoretic Concepts in Computer Science WG'93*, pages 45–56, 1994.

[6] H. L. Bodlaender and T. Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. In Z. Fülöp and F. Gécseg, editors, *Proceedings 22nd International Colloquium on Automata, Languages and Programming*, pages 268–279, Berlin, 1995. Springer-Verlag, Lecture Notes in Computer Science 944.

[7] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7:555–581, 1992.

[8] B. Courcelle. Graph rewriting: an algebraic and logical approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, volume B*, pages 192–242, Amsterdam, 1990. North Holland Publ. Comp.

[9] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[10] J. Lagergren and S. Arnborg. Finding minimal forbidden minors using a finite congruence. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, pages 532–543. Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[11] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.