On disjoint cycles

Hans L. Bodlaender

RUU-CS-90-29
August 1990

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

On disjoint cycles

Hans L. Bodlaender

Technical Report RUU-CS-90-29
August 1990

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN:0924-3275

ON DISJOINT CYCLES*

Hans L. Bodlaender
Department of Computer Science, University of Utrecht
P.O.Box 80.089, 3508 TB Utrecht, the Netherlands

Abstract

It is shown, that for each constant $k \geq 1$, the following problems can be solved in $\mathcal{O}(n)$ time: given a graph G, determine whether G has k vertex disjoint cycles, determine whether G has k edge disjoint cycles, determine whether G has a feedback vertex set of size $\leq k$. Also, every class $\mathcal{G}$, that is closed under minor taking, or that is closed under immersion taking, and that does not contain the graph formed by taking the disjoint union of k copies of K_3, has an $\mathcal{O}(n)$ membership test algorithm.

1 Introduction.

In this paper we consider the following problem: given a graph $G = (V, E)$, does G contain at least k vertex disjoint cycles. If k is part of the instance of the problem, then the problem is NP-complete, because then it contains PARTITION INTO TRIANGLES as a special case (take $k = |V|/3$). (See [12].) In this paper we consider the problem for fixed k.

The problem can be seen as a special case of the MINOR CONTAINMENT problem. A graph G is a minor of a graph H, if G can be obtained from H by a series of vertex deletions, edge deletions and edge contractions (an edge contraction is the operation to replace two adjacent vertices v, w by one vertex that is adjacent to all vertices that were adjacent to v or w). Robertson and Seymour [16] showed that for every fixed H there exists an $\mathcal{O}(n^3)$ algorithm to test whether a given graph G contains H as a minor. If H is planar, then they give an $\mathcal{O}(n^2)$ algorithm. Using recent results of Lagergren [13] or of Arnborg et. al. [2] it is possible to improve on the $\mathcal{O}(n^2)$ bound. In the former case, using an approximation for treewidth, one arrives at an $\mathcal{O}(n \log^2 n)$ algorithm. In the latter case, using graph rewriting, one gets an $\mathcal{O}(n \log n)$ algorithm, or an algorithm that uses $\mathcal{O}(n)$ time in the uniform cost measure and polynomial (not linear) space. In [6] (extending results in [11]) a

*This work has been partially supported by the ESPRIT II Basic Research Actions Program of the EC, under contract No. 3075 (project ALCOM).

class of graphs is given for which minor tests can be done in $\mathcal{O}(n)$ time with the help of depth first search. This class does not include the graph consisting of k disjoint copies of K_3, (here denoted as $k \cdot K_3$.) As a graph G contains $k \cdot K_3$ as a minor, if and only if G contains k vertex disjoint cycles, it follows from this paper that testing whether $k \cdot K_3$ is a minor of a given graph G can be done in $\mathcal{O}(n)$ time for fixed k.

We also consider the problem of finding k edge disjoint cycles in a graph G. This problem is related to the problem of finding immersions: a graph G has $k \cdot K_3$ as an immersion, if and only if G contains k edge disjoint cycles. (A graph G is an immersion of H, if G can be obtained from H by a series of vertex deletions, edge deletions and edge lifts. An edge lift is the operation of replacing two edges $(v,w),(w,x)$ by an edge (v,x).)

The problem in this paper is related to the (much more difficult) problem of finding vertex or edge disjoint paths between a number of fixed pairs of vertices. For an overview of recent results on this problem, see e.g. [17].

We also consider the problem of finding a minimum size feedback vertex set is a graph. A feedback vertex set in graph $G = (V,E)$, is a set $W \subseteq V$ such that the graph $(V - W, E - \{(v,w) | v \in W \vee w \in W\})$ is cycle-free. The problem of finding a minimum feedback vertex set is NP-complete [12]. Here we show that for fixed k, the problem of finding a feedback vertex set of size $\leq k$, if it exists, is solvable in $\mathcal{O}(n)$ time.

It must be pointed out that, although we give $\mathcal{O}(n)$ algorithms, these algorithms may be far from practical, due to the large constant factor, hidden in the "$\mathcal{O}$" notation. However, we believe that with further optimizations, practical algorithms can be derived, that have an $\mathcal{O}(n)$ worst case time bound. The purpose of this paper is to show that there exist linear time algorithms for the considered problems, not to derive the best possible algorithm.

We end this introduction with some definitions and notations, used in this paper.

The subgraph of $G = (V,E)$, induced by $W \subseteq V$ is denoted by $G[W] = (W, \{(v,w) \in E | v,w \in W\})$.

A tree-decompostion of a graph $G = (V,E)$ is a pair $(\{X_i | i \in I\}, T = (I,F))$ with $\{X_i | i \in I\}$ a family of subsets of V, and T a tree, such that $\cup_{i \in I} X_i = V$; $\forall (v,w) \in E : \exists i \in I : v,w \in X_i$; for all $v \in V : \{i \in I | v \in X_i\}$ forms a connected subtree of T.

The treewidth of a tree-decompositon $(\{X_i | i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G = (V,E)$ is the minimum treewidth over all possible tree-decompositions of G.

All graphs considered in this paper are assumed to be finite, undirected and simple. n denotes the number of vertices of graph $G = (V,E)$.

2 Finding two vertex or edge disjoint cycles.

In this section we give a linear time algorithm to test whether a given graph contains two vertex or edge disjoint cycles. We need the following lemmas.

Lemma 2.1
Let $G = (V, E)$ be an undirected graph. Suppose G contains a cycle c and four vertex disjoint paths, that have their endpoints on c, and are edge disjoint from c. Then G contains two vertex disjoint cycles.

Proof.
Let $v_{i_1}, v_{i_2} (1 \leq i \leq 4)$ be the endpoints of the four paths. For $i = 1, \ldots, 4$, look at a path formed by going from v_{i_1} to v_{i_2} over the cycle c. If one of these paths is entirely contained in another, or two of these paths are disjoint, then G contains two vertex disjoint cycles, as shown in Figure 1.

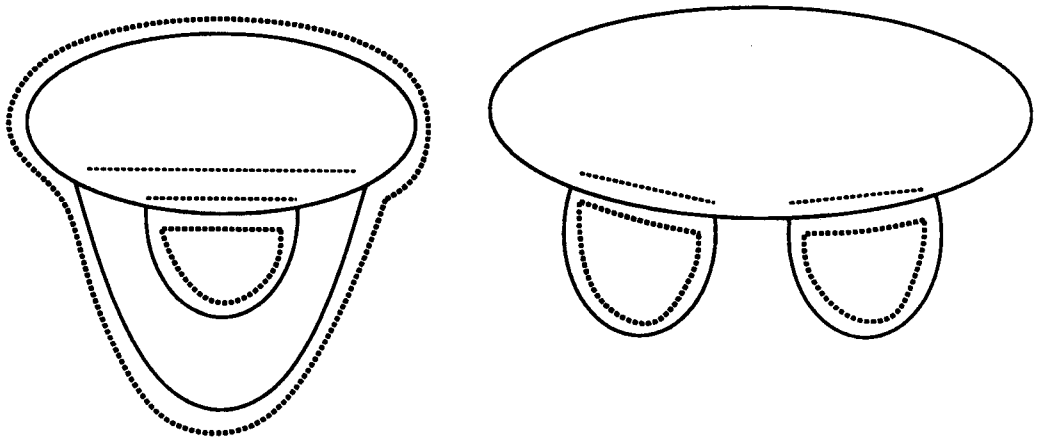

Figure 1: See Lemma 2.1

If this case does not appear for any pair $i_1 \neq i_2$, then we are in a situation as shown in Figure 2. Again G contains two vertex disjoint cycles. □

Lemma 2.2
Let $G = (V, E)$ be an undirected graph. Suppose G contains two cycles c_1, c_2 that share exactly one vertex and suppose G contains a path p, that is vertex disjoint from c_1, has both endpoints in c_2, and is edge disjoint from c_2. Then G contains two vertex disjoint cycles.

Proof.
The result is obvious from Figure 3. □

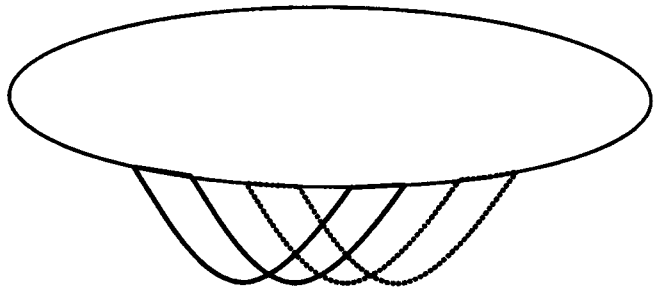

Figure 2: See Lemma 2.1

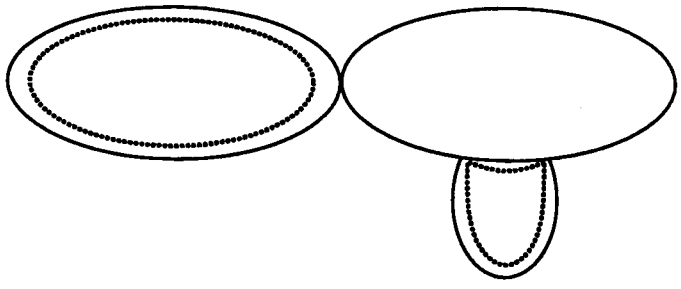

Figure 3: See Lemma 2.2

Theorem 2.3
There exists an algorithm, that uses $\mathcal{O}(n)$ time, and that given a graph $G = (V, E)$, either finds two vertex disjoint cycles in G or outputs a feedback vertex set X with $|X| \leq 9$.

Proof.
First test whether G is cycle-free. If so, take $X = \emptyset$, and we are done. Otherwise find a cycle c in G. Let W be the set of vertices on this cycle. Consider $G[V - W]$. If $G[V - W]$ contains a cycle we have two vertex disjoint cycles and we are done. So suppose $G[V - W]$ is a forest.

The algorithm now proceeds by trying to find the four paths, mentioned in Lemma 2.1. A cycle which shares exactly one vertex with W is treated as a path with both endpoints the same vertex in W, and is not handeled separately. In the remainder of this proof, an i-path is a path, that has i endpoints in W and no other vertex in $W (i = 1, 2)$.

The algorithm uses a counter α, that denotes the number of vertex disjoint 2-paths, found so far. We also use a set $X \subseteq V$. Initially $X \subseteq \emptyset$. An i-path is X-free, if it does not contain a vertex in X.

Repeat the following procedure, until it stops.

1. If G is a cycle with possibly a number of trees attached to it (each tree sharing exactly one vertex with the cycle), then take an arbitrary vertex $v \in W$, and let $X = \{v\}$. Now stop. Clearly $G[V - X]$ is cycle-free.

2. If $\alpha = 4$, then stop. By Lemma 3.1 or Lemma 3.2 we can find two vertex disjoint cycles.

3. If $G[V - X]$ is a forest, then stop. Now $\alpha \leq 3$.

4. Otherwise, an X-free 2-path exists. We will find such an X-free 2-path p, and then put at most 3 vertices from this path in X, such that all further X-free 2-paths are disjoint from p.

Let a *junction* be a vertex $v \in V - (W \cup X)$, that is endpoint of three edge-disjoint X-free 1-paths. (It follows that these paths are vertex-disjoint, except for the endpoints.) See Figure 4 for an example.

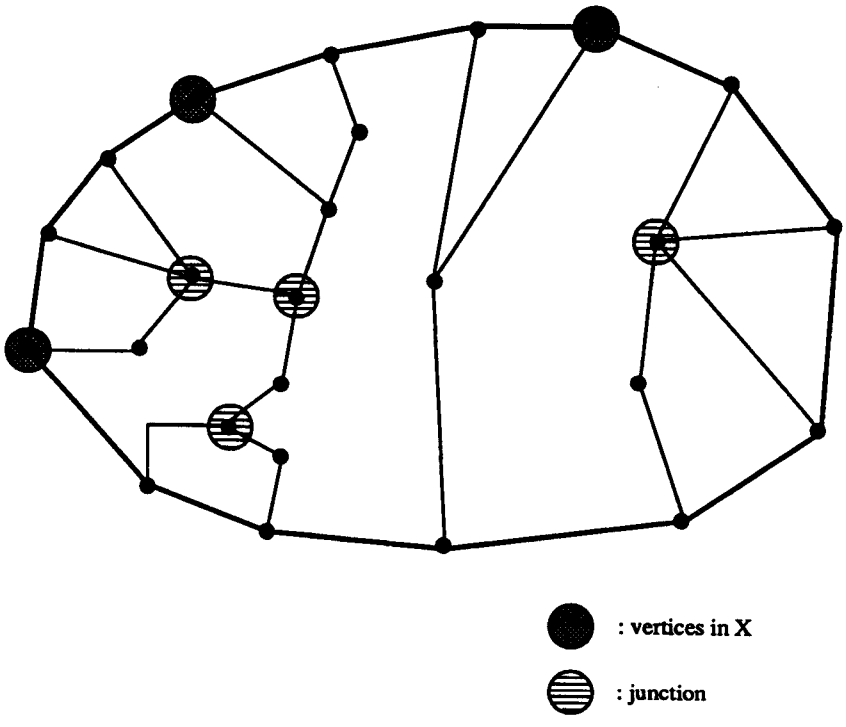

Figure 4: Example of junctions

Now one easily determines in linear time what vertices are a junction. If no junction exists, then take an X-free 2-path p, and put its endpoints in X. Increase α by 1, and continue with step 2. Note that all 2-paths, that are now X-free are vertex disjoint from p, otherwise p would contain a junction.

In case there exists at least one junction, we search for a junction, such that at least 2 of its X-free 1-paths do not contain other junctions. Such a junction can be

found in the following way. Start in an arbitrary junction v_o. Consider two of its X-free 1-paths. If neither contains a junction, we are done. Otherwise, let $v_1 \neq v_o$ be a junction on one of these paths.
Repeat the process with v_1, instead of v_o, and consider the two X-free 1-paths of v_1, that do not contain v_o. As $G[V - W]$ is cycle-free, this process ends after a finite number of steps. It is not necessary to consult each edge more than a constant number of times, so the procedure has a linear time implementation.

Let w_1, w_2 be the other endpoints of the two found paths from junction v_j. Combining these two paths we have an X-free 2-path p from w_1 to w_2. Now add w_1, w_2 and v_j to X, and add 1 to α.

We claim that all 2-paths, that are X-free (for the new set X), are disjoint from p. Because if not, the first vertex v^*, that is shared by p and an X-free 2-path p', would have been a junction before the addition of w_1, w_2 and w_j to X (see Figure 5). But then $v_j \rightarrow w_1$ or $v_j \rightarrow w_2$ would have contained a junction, and hence would not have been chosen by the procedure.

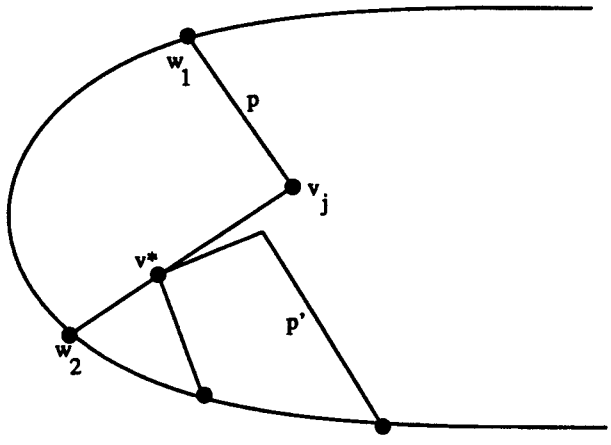

Figure 5: See Theorem 2.3

The procedure terminates, either because two vertex independent cycles are found, or because $G[V - X]$ is cycle-free. In the latter case $\alpha \leq 3$, and hence $|X| \leq 9$.

The procedure can be implemented to use time, linear in the number of vertices, even if there are more than a linear number of edges: testing acyclicity can be done in $\mathcal{O}(n)$ time, $G[V - W]$ has, when acyclic, $\mathcal{O}(n)$ edges, and one never needs to inspect more than $|X| + 3 = \mathcal{O}(1)$ edges (v, w) with $v \in V - W, w \in W$ per vertex $v \in V - W$. □

Theorem 2.4
There exists an algorithm, that uses $\mathcal{O}(n)$ time, and that given a graph $G = (V, E)$,

either finds two vertex disjoint cycles in G, or outputs a tree-decomposition of G with treewidth ≤ 10.

Proof.
First use the procedure of Theorem 2.3. In case we have a feedback vertex set X with $|X| \leq 9$, then make a tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ of $G[V - X]$ with treewidth 1 (this can easily be done, see e.g. [4]).

Now $(\{X_i \cup X | i \in I\}, T = (I, F))$ is a tree-decomposition of G with treewidth ≤ 10. □

Many graph problems, including many NP-complete problems, become linear time solvable for graphs, given together with a tree-decomposition with constant bounded treewidth (see e.g. [1, 3, 7, 18]). As the question, whether the input graph contains $\geq k$ vertex (or edge) disjoint cycles can be formulated in monadic second order form (see [3]), or in the calculus of [7], the next result follows from these papers and Theorem 2.4.

Corollary 2.5
There exists an algorithm, that uses $\mathcal{O}(n)$ time, that given a graph $G = (V, E)$, decides whether G contains two vertex (or edge) disjoint cycles, and if so, outputs these.

We remark here, that the given method is not (yet) practical, due to the high constant factor. However, we believe that the procedure of Theorem 2.3 with some optimizations, followed by an extensive case analysis, can yield a practical algorithm for the problem, that has a good worst-case running time. Probably a good average case running time is obtained by a straightforward backtracking procedure. However, this approach may in some cases give exponential time, e.g., with inputs of the form $K_{3,m}$.

Also, further optimizations are possible in the case of edge disjoint cycles.

3 Finding more than two disjoint cycles.

In this section we consider the problem of finding $k \geq 3$ vertex (or edge) disjoint cycles. We first need a lemma, similar to Lemma 2.1.

Lemma 3.1
Let $G = (V, E)$ be an undirected graph. Suppose G contains two vertex disjoint cycles c_1, c_2, and 9 vertex disjoint paths, that each have one endpoint in c_1, and one endpoint in c_2. Then G contains at least three vertex disjoint cycles.

Proof.
The proof of this lemma relies on a detailed and not very interesting case analysis, which is omitted from this paper.

Instead, we give here a much shorter proof for the weaker result, where " 9 " is replaced by "26".

Without loss of generality, we may suppose that the 26 vertex disjoint paths between c_1 and c_2 do not share with c_1 or c_2 other vertices than their endpoints. Number the vertices on c_1 $v_1 \cdots v_r$ in order of a traversal of c_1, and likewise number the vertices on c_2 $w_1 \cdots w_s$. Order the paths between c_1 and c_2 with respect to their endpoints on c_1. Now consider the sequence of the 26 endpoints of the paths on c_2, in this order. This is a sequence of 26 different numbers. By a theorem of Erdös and Szekeres [9], this sequence has a subsequence of six numbers, corresponding to vertices $v_{i_1} \cdots v_{i_6}$ with $i_1 < i_2 \cdots < i_6$ or $i_1 > i_2 > \cdots > i_6$. This corresponds to a situation, shown in Figure 6. Clearly, G has three vertex disjoint cycles. □

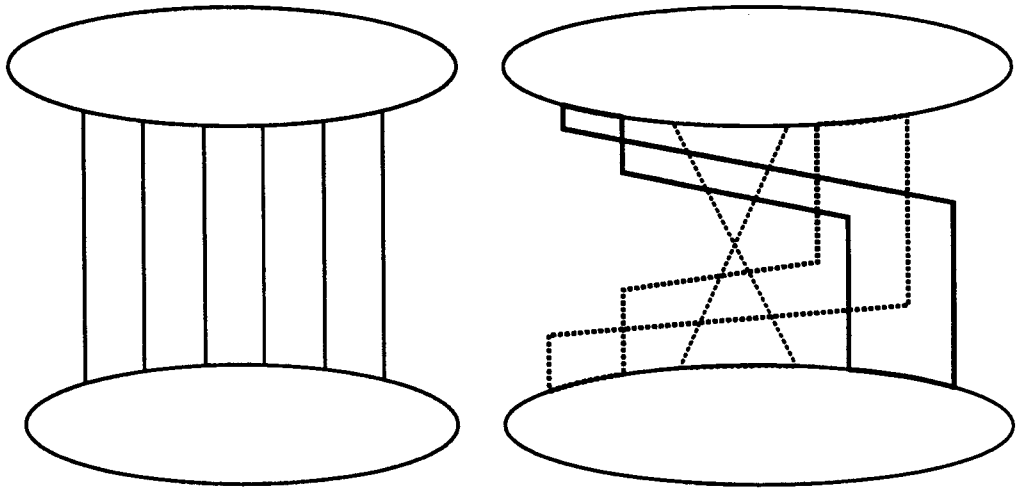

Figure 6: See Lemma 3.1

Lemma 3.2
Suppose $G = (V, E)$ contains $k-1$ vertex disjoint cycles $c_1 \cdots c_{k-1}$, and $3(k-1) + 4(k-1)(k-2) + 1$ disjoint paths, such that each path has both its endpoints on the cycles $c_1 \cdots c_{k-1}$, and does not share another vertex with the cycles. Then G contains k vertex disjoint cycles.

Proof.
By a pidgeonhole argument, either there are at least four paths with all endpoints on the same cycle c_i, or there are two cycles c_i, c_j with at least nine paths with one

endpoint on c_i, and one endpoint on c_j. In the former case, apply Lemma 2.1. In the latter case, apply Lemma 3.1. In both cases, we get one extra, disjoint cycle. □

Theorem 3.3
For every $k \geq 2$, there exists an algorithm, that uses $\mathcal{O}(n)$ time, and that given a graph $G = (V, E)$, either finds k vertex disjoint cycles, or finds a feedback vertex set $X \subseteq V$ with $|X| \leq 12k^2 - 27k + 15$.

Proof.
If $k = 2$, then use Theorem 2.3. Otherwise, first recursively apply the algorithm for $k - 1$. Either a suitable set X is found, or we find $k - 1$ vertex disjoint cycles. In the latter case, apply a procedure, similar to the one in the proof of Theorem 3.3, but instead let W be the set of all vertices on the $k - 1$ vertex disjoint cycles, and stop when $\alpha = 4k^2 - 9k + 6 (= 3(k-1) + 4(k-1)(k-2) + 1)$. If we stop with $\alpha = 4k^2 - 9k + 6$, then we have enough paths to apply Lemma 4.2. Otherwise $|X| \leq 3\alpha \leq 3(4k^2 - 9k + 5)$, and $G[V - X]$ is cycle free. □

In the same way as in section 2, we can derive the following results:

Theorem 3.4
For every $k \geq 2$, there exists an algorithm, that uses $\mathcal{O}(n)$ time, and that given a graph $G = (V, E)$, either finds two vertex disjoint cycles in G, or outputs a tree-decomposition of G with treewidth $\leq 12k^2 - 27k + 16$.

Corollary 3.5
For every $k \geq 2$, there exists an algorithm, that uses $\mathcal{O}(n)$ time, that given a graph $G = (V, E)$, decides whether G contains k vertex (or edge) disjoint cycles, and if so, outputs these.

More efficient algorithms can be found with help of a more detailed case analysis. The purpose here was only to show that linear time is achievable, not to give the most efficient algorithm.

4 Some consequences.

Robertson and Seymour [15, 14] proved that for every class of graphs G, that is closed under taking of minors (or immersions), there exists a finite set of graphs, called the obstruction set of G, such that a graph H belongs to G, if and only if no graph G in the obstruction set of G is a minor (immersion) of H. Combining this result with Theorem 4.4, the fact that minor tests can be done in $\mathcal{O}(n)$ time with a constant width tree-decomposition, and the observation that a graph G contains $k \cdot K_3$ as a minor (immersion), if and only if G contains k vertex (edge) disjoint cycles, we have the following result:

Corollary 4.1
Let G be a class of graphs, closed under minor (immersion) taking, that does not contain all graphs $k \cdot K_3$. Then there exists an $\mathcal{O}(n)$ algorithm to test membership in G.

Note that the algorithm uses also linear space, standard cost measure (in constrast with [2]), is non-constructive, and has a very large constant factor. Another consequence of our results is for the FEEDBACK VERTEX SET problem.

Corollary 4.2
For every constant k, there exists an algorithm, that uses $\mathcal{O}(n)$ time, that determines whether a given graph $G = (V, E)$ contains a feedback vertex set of size $\leq k$, and if so, outputs one.

Proof.
Use Theorem 4.4. If G contains $\geq k+1$ vertex disjoint cycles, then every feedback vertex set contains at least $k+1$ vertices. Otherwise, use the tree-decomposition, and a dynamic programming algorithm, e.g. as in [3, 5, 7] to find the optimal feedback vertex set. □

This result was also obtained, independently, by Fellows [10].

For a class of graphs $\mathcal{G}$, let *within k vertices of $\mathcal{G}$* denote the class of graphs $\{G = (V, E) \mid$ there exists a subset $W \subseteq V$ of at most k vertices, such that $G[V - W] \in \mathcal{G}\}$, i.e., a graph G is within k vertices of $\mathcal{G}$, if we can remove $\leq k$ vertices from G and the resulting graph belongs to $\mathcal{G}$. This type of problem was considered in [8]. We have the following easy result:

Lemma 4.3
Let $\mathcal{G}$ be a class of graphs with $G \in \mathcal{G} \Rightarrow$ treewidth $(G) \leq k$. Then $G \in$ within l vertices of $\mathcal{G} \Rightarrow$ treewidth $(G) \leq k + l$.

Proof.
Use the same technique as in Theorem 2.4. □

A corollary of this result is that a graph $G = (V, E)$ with a 'partial feedback vertex set' $W \subseteq V$ such that $G[V - W]$ does not contain a cycle with length $\geq K$ has treewidth $\leq |W| + K - 2$. (Use that a graph that contains no cycle with length $\geq K$ has treewidth $\leq K - 2$ [11].) Clearly, the same results hold if we consider classes *within k edges of $\mathcal{G}$*.

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - A survey. *BIT*, 25:2–23, 1985.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. Technical Report 90-02, Laboratoire Bordelais de Recherche en Informatique, Bordeaux, 1990.

[3] S. Arnborg, J. Lagergren, and D. Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Proc. 15 th ICALP*, pages 38–51. Springer Verlag, Lect. Notes in Comp. Sc. 317, 1988. To appear in J. of Algorithms.

[4] H. L. Bodlaender. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Dept. of Computer Science, Utrecht University, Utrecht, 1986.

[5] H. L. Bodlaender. NC-algorithms for graphs with small treewidth. In J. van Leeuwen, editor, *Proc. Workshop on Graph-Theoretic Concepts in Computer Science WG'88*, pages 1–10. Springer Verlag, LNCS 344, 1988.

[6] H. L. Bodlaender. On linear time minor tests and depth first search. In *Proceedings Workshop on Algorithms and Datastructures WADS'89*, pages 577–590. Springer Verlag, Lecture Notes in Computer Science vol. 382, 1989.

[7] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursive constructed graph families. Manuscript, 1988.

[8] D. J. Brown, M. R. Fellows, and M. A. Langston. Nonconstructive polynomial-time decidability and self-reducibility. *Int. J. Computer Math.*, 31:1–9, 1989.

[9] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compos. Math.*, 2:464–470, 1935.

[10] M. R. Fellows, 1989. Personal communication.

[11] M. R. Fellows and M. A. Langston. On seach, decision and the efficiency of polynomial-time algorithms. In *Proceedings of the 21th Annual Symposium on Theory of Computing*, pages 501–512, 1989.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[13] J. Lagergren. Efficient parallel algorithms for tree-decomposition and related problems. To appear in Ann. Symp. on Foundations of Computing Science'90, 1990.

[14] N. Robertson and P. D. Seymour. Graph minors. XVI. Wagner's conjecture. To appear.

[15] N. Robertson and P. D. Seymour. Graph minors — a survey. *Surveys in Combinatorics*, pages 153–171, 1985.

[16] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. Manuscript, 1986.

[17] J. van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science, A: Algorithms and Complexity Theory*, Amsterdam, 1990. North Holland Publ. Comp.

[18] T. V. Wimer. *Linear algorithms on k-terminal graphs.* PhD thesis, Dept. of Computer Science, Clemson University, 1987.