The Valve Location Problem in Simple Network Topologies

Hans L. Bodlaender[1], Alexander Grigoriev[2], Nadejda V. Grigorieva[3], and Albert Hendriks[1]

[1] Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
hansb@cs.uu.nl, alberthendriks@gmail.com
[2] Department of Quantitative Economics, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands
a.grigoriev@ke.unimaas.nl
[3] Institute of Power Resources Transport (IPTER), 144/3, pr. Octyabrya, Ufa-450055, Russia
n_grigorieva@yahoo.com

Abstract. To control possible spills in liquid or gas transporting pipe systems, the systems are usually equipped with shutoff valves. In case of an accidental leak these valves separate the system into a number of pieces limiting the spill effect. In this paper, we consider the problem, for a given edge-weighted network representing a pipe system and for a given number of valves, to place the valves in the network in such a way that the maximum possible spill, i.e. the maximum total weight of a piece, is minimized. We show that the problem is NP-hard even if restricted to any of the following settings: (i) for series-parallel graphs and hence for graphs of treewidth two; (ii) if all edge weights equal one. If the network is a simple path, a cycle, or a tree, the problem can be solved in polynomial time. We also give a pseudo-polynomial time algorithm and a fully polynomial approximation scheme for networks of bounded treewidth.

Keywords: Valve location problem; computational complexity; bounded treewidth; dynamic programming; binary search.

1 Introduction

In this paper, we consider a combinatorial problem that arose from a number of applications connected to operations and maintenance of a broad variety of transportation systems; for applications related to the long oil and gas pipelines see e.g. [10]; for applications in water supply engineering see [15]; for applications in electrical grid maintenance see [7]. Let us briefly discuss the related problem arising in oil and gas transportation. A pipeline is the most efficient and environmentally friendly way to transport hazardous liquids and gases, e.g. crude oil or natural gas, over land. In normal daily operations, pipelines do not produce any pollution. However, due to external factors or pipe corrosion, accidents on

H. Broersma et al. (Eds.): WG 2008, LNCS 5344, pp. 55–65, 2008.

pipelines sometimes happen and the accidental damage can be substantial. To control possible spills, every pipe system is usually equipped with special shutoff valves. Whenever the pipe system is depressurized, the valves automatically and instantly separate the pipe system into *pieces*. Therefore, the quantity of hazardous liquid or gas potentially leaving the system equals the total length of the pipes in the damaged piece of the system separated by shutoff valves. In the application at hand, there is a given edge-weighted network representing a pipe system and a given number of valves that can be placed in the vertices of the network. We want to solve the following problem: find a valve location in the network that minimizes the maximum total weight of a piece separated by shutoff valves.

This paper is organized as follows. In Section 2, we give a precise graph theoretic formulation of the problem. In Section 3, we show that using dynamic programming the problem can be solved in polynomial time on simple network topologies: paths, cycles and trees. In Section 4, we consider a more general case, namely the graphs of bounded treewidth. For these graphs, we give a pseudo-polynomial time dynamic programming algorithm, and then we turn this algorithm into a fully polynomial approximation scheme (FPTAS). Finally, in Section 5, we discuss the complexity of the problem. Here, we show that the problem is NP-hard even for series-parallel graphs and hence for graphs of treewidth at most two. We also show that the unweighted version of the problem, i.e. the problem where all edge weights equal one, is also NP-hard.

2 Graph Theoretic Formulation

The problem can be formulated in graph theoretic terms in a natural way. Let $G = (V, E)$ be an undirected graph representing a pipe network. Edges of the graph represent pipes. Let $\omega_e \in \mathbb{Z}^+$ denote the length of pipe $e \in E$. Vertices of the graph represent connection points between the pipes. Let k be a number of valves to be installed. We assume that a valve can be located in any vertex $v \in V$.

Consider a set of vertices $V' \subseteq V$. If we use V' as valve locations, we use $|V'|$ valves, and partition G into *pieces* as follows. The set of edges E is partitioned into sets with two edges in the same set of the partition if and only if they are on a path in G that does not contain a valve. Thus, E is partitioned into subsets $E_1, E_2, \ldots, E_S$ where edges in E_s, $1 \leq s \leq S$, form a connected component in G called a piece, and for any two subsets, E_s and E_t, the set of endpoints in E_s intersects the set of endpoints in E_t only in elements of V'. The *cost* of V', denoted $W_{\max}(V')$, is

$$W_{\max}(V') = \max_{1 \leq s \leq S} \sum_{e \in E_s} \omega_e,$$

i.e., the maximum total length of a piece or the maximum *spill*.

The VALVE LOCATION problem then is to find a subset V' of vertices in G such that $|V'| \leq k$ and the cost $W_{\max}(V')$ is minimized. In other words, we have

to find a k-elementary separator in G such that the maximum length connected component is minimized. We consider also the unweighted version of the problem where $\omega_e = 1$ for all $e \in E$.

Throughout the paper, n denotes the number of vertices in G, $\omega_{\max}$ the maximum length of an edge:

$$\omega_{\max} = \max_{e \in E} \omega_e,$$

and ω_Σ the total length of all edges:

$$\omega_\Sigma = \sum_{e \in E} \omega_e.$$

Clearly, the maximum spill is yet another network vulnerability measure. This concept is very close to many other known vulnerability measures, e.g. *vertex integrity* of a graph defined as $I(G) = \min\{|S| + m(G - S) : \ S \subset V\}$, where $m(H)$ denotes the maximum order of a component of H, see [2,3]; *minimum balanced separator* defined as a minimum order separator S such that the maximum component in $G - S$ contains at most βn vertices for a given $0 < \beta < 1$, see [1,8,14]; and some other, see e.g. [13]. The key difference between the maximum spill and the known vulnerability measures is that the maximum spill measures vulnerability of a graph in terms of the total edge weight (or length) of a component when all other measures are related to the maximum order (number of vertices) in a component. Of course, practical suitability of a certain measure depends heavily on applications.

Throughout the paper, we measure run time of the algorithms using the widely accepted convention that we can do an addition or multiplication of two integers in $O(1)$ time. If we want to count bit operations, we must multiply run times by a factor $\log \omega_\Sigma$.

3 Simple Networks: Paths, Cycles and Trees

In this section, we give dynamic programming algorithms to solve the problem in simple network topologies: paths, cycles and trees.

3.1 The Valve Location Problem on a Path

We first consider the VALVE LOCATION problem on a path. This simple case appears frequently in the practical settings of the long oil pipelines, and thus is of practical relevance; see [10]. We have two different exact algorithms. One uses 'text book' dynamic programming.

Proposition 1. *The* VALVE LOCATION *problem on a path can be solved by dynamic programming in $O(kn^2)$ time.*

The other algorithm is obtained by using a binary search for the optimal spill value and by checking feasibility of each spill value with a greedy algorithm.

Proposition 2. *Given a path and a value L, we can decide in $O(n)$ time if there is a solution to the* VALVE LOCATION *problem with k valves with cost at most L.*

The same idea also gives the minimum number of valves needed to guarantee a cost that is at most L. Using binary search for the optimal value in the range of integers between 0 and ω_Σ, we directly obtain the following result.

Corollary 1. *For a given path, we can solve the* VALVE LOCATION *problem in $O(n \log \omega_\Sigma)$ time.*

It is also possible to construct a fast 2-approximation algorithm using a greedy strategy for paths.

Proposition 3. *The* VALVE LOCATION *problem on a path admits a 2-approximation algorithm that uses $O(n)$ time.*

Finally, we can sharpen Proposition 3 when $\omega_{\max} \geq 3A_p$.

Proposition 4. *Consider the* VALVE LOCATION *problem on a path. Let k be the number of valves. If $\omega_{\max} \geq 3\omega_\Sigma/(k+1)$, then the optimal solution has cost $\omega_{\max}$. An optimal solution can be found in this case in $O(n)$ time.*

3.2 Cycles

If G is a cycle, then we can obtain exact and approximate solutions for the VALVE LOCATION by using variants to the algorithms for paths.

Proposition 5. *The* VALVE LOCATION *problem on a cycle admits a 2-approximation algorithm that uses $O(n)$ time.*

Proposition 6. *Consider the* VALVE LOCATION *problem on a cycle. Let k be the number of valves. If $\omega_{\max} \geq 3\omega_\Sigma/k$, then the optimal spill equals $\omega_{\max}$. An optimal valve location can be found in this case in $O(n)$ time.*

Theorem 1. *The* VALVE LOCATION *problem on a cycle can be solved by solving $O(n/k)$* VALVE LOCATION *problems on paths of length at most n.*

Corollary 2. *The* VALVE LOCATION *problem on a cycle can be solved in $O(n \min \{\log \omega_\Sigma, n/k\})$ time.*

3.3 Trees

Very recently, see [7], we became aware of the fact that the VALVE LOCATION problem on trees is an important modern research topic in electrical engineering. Whenever a regional power supply network (a tree) should be maintained, the engineers are shutting down some small subtree and they are interested in an optimal location of switchers. This application brings us to the VALVE LOCATION in trees. Surprisingly enough, already this special case of the problem is quite complicated: according to [7], a typical modern approach to the problem is a

genetic algorithm. In this section, we present a nontrivial algorithm that solves the problem on trees in polynomial time. More specifically, we show:

Theorem 2. *The* VALVE LOCATION *problem on a tree can be solved in* $O(nk^2 \log(n\omega_{\max}))$ *time.*

The global structure of the algorithm is a binary search on the optimal value in the range of integers between 0 and $n\omega_{\max}$. Thus, we directly obtain Theorem 2 as a corollary of the next result.

Proposition 7. *Given a tree, and an integer* L*, we can decide in* $O(nk^2)$ *time if we can place* k *valves with maximum piece size at most* L*.*

Proof. We choose an arbitrary vertex v_r as root of the tree. For rooted subtrees T', and integers i, $0 \leq i \leq k$, we define

$A_{T',L}(i)$ = the minimum over all possible ways to put at most i valves in T' such that no piece in T' has a total length of more than L, of the total length of the piece that contains the root node of T'.
$A_{T',L}(i) = 0$, if there is a way to put at most i valves in T' such that no piece in T' has a total length of more than L, such that there is a valve in the root node of T'.
$A_{T',L}(i) = \infty$, if there is no possible way to put at most i valves in T' such that no piece in T' has a total length of more than L.
$P_{T',L}(i) =$ **true** if and only if $A_{T',L}(i) = 0$, i.e. if we can put at most i valves in T' such that no piece in T' has a total length of more than L, such that there is a valve in the root node of T'.

We will compute tables $A_{T',L}$ and $P_{T',L}$ for several subtrees of T:

- For each vertex v in T except v_r, we compute a table for the subtree, consisting of the parent of v in T, v, and all the descendants of v. The root of this subtree is the parent of v. Call this subtree T_v^+.
- For each vertex v in T: if v has i children $w_1, w_2, \ldots, w_i$, then for each j, $0 \leq j \leq i$, we compute a table for the subtree, consisting of v, $w_1, \ldots, w_j$, and all descendants of $w_1, w_2, \ldots, w_j$. Vertex v is the root of this subtree. Call this subtree $T_{v,j}$. For the case $j = i$, write $T_v = T_{v,i}$; this is the tree consisting of v and all its descendants.

The following two lemmas give recursive formulations that show how to compute these tables.

Lemma 1. *Let* T *be obtained by taking the union of trees* T' *and* T'' *such that the root* r *of* T' *and* T'' *is the only vertex that belongs to both trees. Let* $0 \leq i \leq k$*.*

1. $P_{T,L}(i)$*, if and only if there are* i'*,* i'' *with* $i' + i'' = i - 1$*,* $0 \leq i' \leq k$*,* $0 \leq i'' \leq k$*, such that* $P_{T',L}(i')$ *and* $P_{T'',L}(i'')$*.*
2. *If* $P_{T,L}(i)$*, then* $A_{T,L}(i) = 0$*.*

3. *If not* $P_{T,L}(i)$*, then*

$$A_{T,L}(i) = \min_{i',i'',i'+i''=i,0\leq i,i'} A_{T',L}(i') + A_{T'',L}(i'')$$

if this term is at most L*, otherwise* $A_{T,L}(i) = \infty$*.*

Lemma 2. *Let* T *be a tree with root* r*, and let* T^+ *be obtained by adding an edge* $\{r, r'\}$ *to a new vertex* r' *with length* ℓ*. Let* r' *be the root of* T^+*. Let* $0 \leq i \leq k$*, and* L *be an integer.*

1. $P_{T^+,L}(i)$ *holds if and only if* $i > 0$ *and* $A_{T,L}(i-1) + \ell \leq L$*.*
2. *If* $P_{T^+,L}(i)$*, then* $A_{T^+,L}(i) = 0$*.*
3. *If not* $P_{T^+,L}(i)$*, then* $A_{T^+,L}(i) = A_{T^+,L}(i) + \ell$*, if this term is at most* L*, and* $A_{T^+,L}(i) = \infty$ *otherwise.*

Using Lemmas 1 and 2, we can compute all desired tables. Recall that L is fixed during the computation. Now, for all vertices in the tree, in postorder, we compute all values $A_{T_v,L}(i)$, and $P_{T_v,L}(i)$, for all i, $0 \leq i \leq k$. This is done in the following way. If v is a leaf of T, then computing these values is trivial. Otherwise, suppose v has s children, say $w_1, w_2, \ldots, w_s$. For all j, $1 \leq j \leq s$, we compute all values $A_{T_{v,j},L}(i)$, and $P_{T_{v,j},L}(i)$ for all i, $0 \leq i \leq k$. In case $j = 1$, we note that $T_{v,1}$ is the same subtree as $T^+_{w_1}$. Thus, using Lemma 2, we can compute the values $A_{T_{v,1},L}(i)$, and $P_{T_{v,1},L}(i)$ from the already earlier computed tables $A_{T_{w_1},L}$ and $P_{T_{w_1},L}$. For $2 \leq j \leq s$, we note that $T_{v,j}$ is the union of $T_{v,j-1}$ and $T^+_{w_j}$. Thus, we first compute the tables $A_{T^+_{w_j}}, L$ and $P_{T^+_{w_j}}, L$ given the tables $A_{T_{w_j},L}$ and $P_{T_{w_j},L}$ using Lemma 2. Then, we compute the tables $A_{T_{v,j},L}$ and $P_{T_{v,j},L}$ from the tables $A_{T^+_{w_j},L}$, $P_{T^+_{w_j},L}$, $A_{T_{v,j-1},L}$ and $P_{T_{v,j-1},L}$, using Lemma 1. Finally, note that $T_{v,s} = T_v$. When we have the tables $A_{T_{v_r},L}$ and $P_{T_{v_r},L}$, we can easily decide whether we can place k valves in T with maximum piece size at most L, using the following simple observation.

Proposition 8. *Let* v_r *be the root of* T*. There is a solution to the* VALVE LOCATION *problem with* k *valves and cost at most* L *if and only if* $A_{T_{v_r},L}(k) < \infty$*.*

If T has n vertices, then we compute $O(n)$ tables: $O(1)$ per edge in T. Each table can be computed in $O(k^2)$ time. This can be easily observed from Lemmas 1 and 2. Simply, iterate over all possible values of k and k', and compute the necessary value of k''. Each step involves $O(1)$ computations. Actually, the step that uses Lemma 2 needs only $O(k)$ time. This finishes the proof of Proposition 7. □

Notice that for trees, a result similar to Propositions 4 and 6 holds. However, in this case, this does not lead to an approximation algorithm with constant performance guarantee.

Proposition 9. *Consider the* VALVE LOCATION *problem on a tree. Let* k *be the number of valves. If* $\omega_{\max} \geq 3\omega_\Sigma/k$*, then the optimal spill equals* $\omega_{\max}$*.*

4 Algorithms for Graphs of Bounded Treewidth

In practice, most of the transportation systems are more complicated than trees. This makes the problem more difficult from algorithmic perspective. Fortunately, real-life networks in majority of applications (e.g., for oil and gas pipeline transportation) are outerplanar or, taking this more generally, the corresponding graphs have bounded treewidth; see e.g. [5,6,11]. For this type of networks we have the following results.

Theorem 3. *The* VALVE LOCATION *problem on graphs of treewidth q admits a dynamic programming algorithm running in time $(n\omega_{\max})^{O(q)}$.*

This dynamic programming algorithm follows the lines of several algorithms for other problems on graphs of bounded treewidth. For easier description, we use a *nice* tree decomposition of width at most q; for definition see below.

As a first step, we must find a tree decomposition of width at most q. This can be done in $O(n)$ time for fixed q; see [4]. At this point, we would like to make a remark concerning practical implementations. The algorithm in [4] has such a large hidden constant, that it is not of use in a practical setting. Fortunately, there are several heuristics that often give good bounds. Also, there are fast algorithms that construct tree decompositions of optimal width for graphs of treewidth at most three (including outerplanar graphs), see e.g. [6] for a discussion.

Given a tree decomposition, in $O(n)$ time one can transform it to a *nice* tree decomposition [12] with the same width. We now give the definition of a nice tree decomposition.

A *nice tree decomposition* of a graph $G = (V, E)$ is a rooted binary tree $T = (I, F)$, where each node $i \in I$ is a subset $X_i \subseteq V$, called *bag*, such that

1. $\bigcup_{i \in I} X_i = V$.
2. For all $\{v, w\} \in E$, there exists an $i \in I$, with $v, w \in X_i$.
3. For all $v \in V$, the set $\{i \in I \mid v \in X_i\}$ forms a subtree of T.
4. If $i \in I$ has two children j_1, j_2, then $X_i = X_{j_1} = X_{j_2}$ (JOIN NODE).
5. If $i \in I$ has one child j, then either there is a $v \in X_i$ with $X_j \cup \{v\} = X_i$ (INTRODUCE NODE) or there is a $v \in X_j$ with $X_i \cup \{v\} = X_j$ (FORGET NODE).
6. If $i \in I$ is a leaf in T, then $|X_i| = 1$ (LEAF NODE).

The *width* of a nice tree decomposition is $\max_{i \in I} |X_i| - 1$.

In our dynamic programming algorithm, we compute in postorder for each node of T a table. Associate to node $i \in I$ the subgraph $G_i = G[V_i]$, induced by the set of vertices in X_i or a bag X_j with j a descendant of i: $V_i = \bigcup X_j$, with the union taken over all j in the subtree of T rooted at j.

A placement of valves on the vertices of G_i has a *characteristic*, which is a 5-tuple $(j, Z, L, f, \sim)$, consisting of

- The number j of used valves in G_i.
- The subset $Z \subset X_i$ of the vertices in X_i that contain a valve.
- The maximum length of a piece in G_i.
- A function $f : X_i \to \mathbf{N}$, giving for each vertex $v \in V_i$ the total length of the piece that contains v; if there is a valve on v, then $f(v) = 0$.
- An equivalence relation $\sim$ on X_i, with for all $v, w \in X_i$, $v \sim w$, if and only if there is a path from v to w in G_i that does not contain a vertex with a valve.

In the table of i, we store all possible characteristics of all placements of valves in G_i. Note that in this way, tables have a size that is bounded by $(n\omega_{\max})^{O(q)}$.

A somewhat tedious case analysis, typical for dynamic programming algorithms on graphs of bounded treewidth, shows that we can compute for each of the four types of nodes the table of all characteristics for a node, given such a table for each of the children of the node, in time polynomial in the table size.

Then, computing these tables for all nodes in postorder gives an algorithm computing the table for the root node, and as G_r for the root node r equals G, we obtain the optimal valve location from this table.

We remark that the described dynamic programming is only a pseudo-polynomial time algorithm for the weighted version of the VALVE LOCATION problem on graphs of bounded treewidth. Using standard scaling arguments, we derive the following corollary.

Corollary 3. *The* VALVE LOCATION *problem on graphs of bounded treewidth admits a fully polynomial approximation scheme.*

5 Complexity Results

In this section, we show that two restricted versions of the VALVE LOCATION problem are NP-hard. For general networks it is strongly NP-hard as even the unweighted version of the problem is NP-hard, while for series-parallel graphs (a special case of graphs of treewidth at most two) the problem is weakly NP-hard. Note that this complements the result that the problem is solvable in pseudo-polynomial time on graphs of bounded treewidth.

Theorem 4. *The* VALVE LOCATION *problem is NP-hard even if* $\omega_e = 1$ *for all* $e \in E$.

The proof of Theorem 4 is quite straightforward and it is based on a reduction from the strongly NP-hard problem 3-PARTITION.

The second complexity result is less trivial. For this result let us remind a definition of a series-parallel graph. A *series-parallel graph* is a graph $G = (V, E)$ with two special vertices, called its terminals, often denoted s and t, that can be formed with the following operations:

- A graph consisting of a single edge $\{s, t\}$ between its terminals is a series-parallel graph.

- If G and H are terminal graphs, with terminals s_G, t_G, and s_H and t_H, then the *series composition* of G and H is a series-parallel graph. In the series composition, we take the disjoint union, then identify t_G and s_H, and take s_G and t_H as terminals of the resulting graph.
- If G and H are terminal graphs, with terminals s_G, t_G, and s_H and t_H, then the *parallel composition* of G and H is a series-parallel graph. In the parallel composition, we take the disjoint union, then identify s_G and s_H and identify t_G and t_H. The two vertices obtained by identification are the terminals of the resulting graph.

Theorem 5. *The* VALVE LOCATION *problem is weakly NP-hard for series-parallel graphs.*

Proof. We show that the VALVE LOCATION problem is weakly NP-hard for the following graphs: we have two vertices s and t, and a number of internally disjoint paths from s to t of length exactly five.

We use a reduction from PARTITION; see e.g. [9]. Suppose we are given positive integers $a_1, a_2, \ldots, a_n$. The PARTITION problem asks if these integers can be partitioned into two sets with equal sum, i.e. we look for two sets, each of sum $B = \sum_{i=1}^{n} a_i/2$. We may assume B is integer, as if $\sum_{i=1}^{n} a_i$ is odd, the PARTITION problem trivially has no solution.

As the corresponding instance for the VALVE LOCATION problem, we take n disjoint paths from s to t. Each of these paths has length five, i.e., four intermediate vertices, which we call $v_{i,1}, v_{i,2}, \ldots, v_{i,4}$. The successive lengths of the edges on the ith path are 1, a_i, $B - a_i + n$, a_i, 1. Call the resulting graph G, see Figure 1.

Proposition 10. *Set $A = \{a_1, a_2, \ldots, a_n\}$ can be partitioned into two sets, both of sum B, if and only if we can place at most $2n$ valves in G such that each part has total length at most $B + n$.*

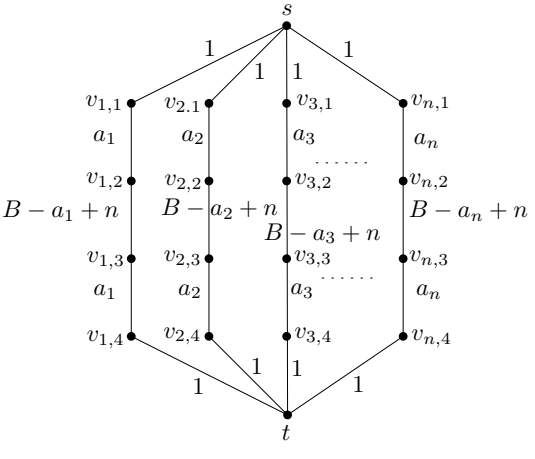

Fig. 1. The series-parallel graph constructed in Theorem 5

The NP-hardness of the VALVE LOCATION problem on series-parallel graphs now follows, by noting that G is series-parallel: a path can be constructed by a sequence of series compositions, and by parallel compositions, we can identify the endpoints of the paths. □

As series-parallel graphs have treewidth two, the results of the previous section show that (unless P=NP), the problem on series-parallel graphs cannot be strongly NP-hard. Moreover, Theorem 5 excludes the possibility for fixed-parameter tractability of the problem with respect to the parameter "graph treewidth".

6 Conclusions

In this paper we presented fast algorithms for several practically relevant classes of instances of the VALVE LOCATION problem. Moreover, applying literally the same techniques to the vertex integrity problem, we can tackle this later problem as well.

Acknowledgments

We thank Alexandr Kostochka for pointing on several very useful references on graph integrity.

References

1. Alon, N., Seymour, P., Thomas, R.: A separator theorem for graphs with an excluded minor and its applications. In: Proc. of the 22nd Symposium on Theory of Computing, STOC 1980, pp. 293–299. ACM Press, New York (1980)
2. Barefoot, C.A., Entringer, R., Swart, H.C.: Vulnerability in graphs: a comparative survey. J. Comb. Math. Comb. Comput. 1, 12–22 (1987)
3. Barefoot, C.A., Entringer, R., Swart, H.C.: Integrity of trees and the diameter of a graphs. Congressus Numerantium 58, 103–114 (1987)
4. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. 25, 1305–1317 (1996)
5. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comp. Sc. 209, 1–45 (1998)
6. Bodlaender, H.L.: Treewidth: Characterizations, applications, and computations. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 1–14. Springer, Heidelberg (2006)
7. Bouwman, S.: A survey of OR models and techniques for electrical grid companies. In: Proceedings of the 33rd Conference on the Mathematics of Operations Research, Lunteren 2008, Landelijk Netwerk Mathematische Besliskunde, The Netherlands (2008)
8. Feige, U., Mahdian, M.: Finding small balanced separators. In: Proceedings of the 37th Annual Symposium on Theory of Computing, STOC 2006, pp. 375–384. ACM Press, New York (2006)

9. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman, San Francisco (1979)
10. Grigorieva, N.V., Grigoriev, A.: Optimal valve location in long oil pipelines. Research Memorandum RM/07/007, Maastricht Research School of Economics of Technology and Organizations (METEOR), Maastricht University, Maastricht, The Netherlands (2007), http://ideas.repec.org/p/dgr/umamet/2007007.html
11. Hicks, I.V., Koster, A.M.C.A., Kolotoglu, E.: Branch and tree decomposition techniques for discrete optimization. In: Smith, J.C. (ed.) TutORials 2005, INFORMS Tutorials in Operations Research Series, ch. 1, pp. 1–29. INFORMS Annual Meeting (2005)
12. Kloks, T.: Treewidth. Computations and Approximations. LNCS, vol. 842. Springer, Berlin (1994)
13. Kratsch, D., Kloks, T., Müller, H.: Measuring the vulnerability for classes of intersection graphs. Discrete Applied Mathematics 77(3), 259–270 (1997)
14. Marx, D.: Parameterized Graph Separation Problems. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 71–82. Springer, Heidelberg (2004)
15. Ozger, S., Mays, L.W.: Optimal location of isolation valves: A reliability approach. In: Water Supply Systems Security, Digital Engineering Library. McGraw-Hill, New York (2004), http://dx.doi.org/10.1036/0071455663.CH13