Fixed-Parameter Algorithms, IA166

Sebastian Ordyniak

Faculty of Informatics
Masaryk University Brno

Spring Semester 2013

Outline

The Party Problem

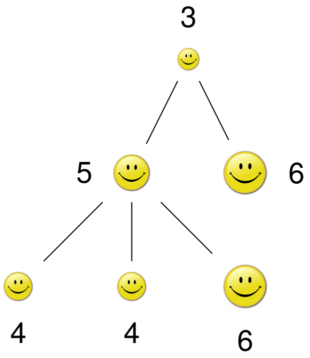
Party Problem

Problem: Invite some colleagues to a party.
Maximize: The total fun factor of the invited people.
Constraint: Everyone should be having fun.
Do not invite a colleague and his direct boss at the same time!

The Party Problem

PARTY PROBLEM

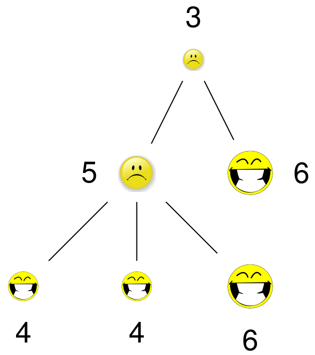
Input: A tree with weights on the vertices.
Question: Find an independent set of maximum weight.



The Party Problem

PARTY PROBLEM

Input: A tree with weights on the vertices.
Question: Find an independent set of maximum weight.



Dynamic Programming on trees (or tree-like structures)

- A dynamic programming algorithm on a tree (or a tree-like structure) usually computes a set of records for every node of the tree in a bottom-up manner, i.e., we first compute the records for the leaves of the tree and then work our way up the tree.
- Informally, a record is a compact representation of partial solutions, i.e., solutions obtained for the subtree below the current node.
- Ideally, the solution for the whole problem can be directly inferred from the set of records computed for the root of the tree.

Example: Solving the party problem

Here and in the sequel we use the following notation: Let T be a (rooted) tree and $t \in V(T)$, then:

- $T(t)$ is the subtree of T rooted at t;
- $\mathcal{R}(t)$ denotes the set of records for the tree node t.

Solving the party problem: The Records

For the PARTY PROBLEM a record is a pair (inc, w) where inc is a boolean value and w is a real value. The semantics of a record for a tree node $t \in V(T)$ is as follows:

- $(0, w) \in \mathcal{R}(t)$ iff w is the maximum weight of an independent set of $T(t)$ that does not contain v;
- $(1, w) \in \mathcal{R}(t)$ iff w is the maximum weight of an independent set of $T(t)$;

Clearly, the solution of the party problem can be easily obtained from $\mathcal{R}(r)$ as the weight w such that $(1, w) \in \mathcal{R}(r)$.

Solving the party problem: Computing the Records

We need to show that we can compute the records for the PARTY PROBLEM for every node of the tree in a bottom-up manner, i.e., we need to show that the set of all records can be computed:

(1) For the leave nodes of the tree.
(2) For every inner node of the tree (given the set of records of all its children).

Solving the party problem: Computing the Records

For the PARTY PROBLEM this can be done as follows (here T is the given tree with weight function w and $t \in V(T)$):

(1) If t is a leave node of T then $\mathcal{R}(t) := \{(0,0),(1,w(t))\}$.

(2) If t is an inner node of T with children $t_1,\dots,t_l$, then
$\mathcal{R}(t) := \{(0,w_o),(1,w_i)\}$ where
$w_o := \sum\{\, w : 1 \leq i \leq l \text{ and } (1,w) \in \mathcal{R}(t_i)\,\}$
and
$w_i := \max\{w_o, w(t) + \sum\{\, w : 1 \leq i \leq l \text{ and } (0,w) \in \mathcal{R}(t_i)\,\}\}$.

Solving the party problem: Computing the Records

For the PARTY PROBLEM this can be done as follows (here T is the given tree with weight function w and $t \in V(T)$):

(1) If t is a leave node of T then $\mathcal{R}(t) := \{(0,0),(1,w(t))\}$.

(2) If t is an inner node of T with children $t_1,\dots,t_l$, then
$\mathcal{R}(t) := \{(0,w_o),(1,w_i)\}$ where
$w_o := \sum\{\, w : 1 \leq i \leq l \text{ and } (1,w) \in \mathcal{R}(t_i)\,\}$
and
$w_i := \max\{w_o, w(t) + \sum\{\, w : 1 \leq i \leq l \text{ and } (0,w) \in \mathcal{R}(t_i)\,\}$.

This gives a polynomial time algorithm for the PARTY PROBLEM on trees!

Outline

Treewidth

Introduction

- Treewidth is a measure of how “tree-like” a graph is.
- Treewidth has become a very successful notion both in structural and algorithmic graph theory.
- Almost every natural problem on graphs becomes solvable in polynomial time on graphs of bounded treewidth, usually even fixed-parameter tractable when parameterized by treewidth.
- Algorithms on graphs of bounded treewidth usually follow the general dynamic programming approach that we presented for trees.
- Treewidth is usually defined in terms of a so called tree-decomposition (although many different alternative definitions exist).

Definition

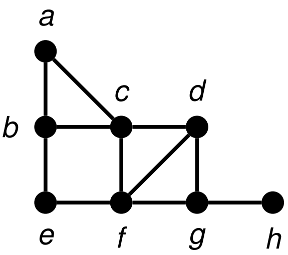
A tree decomposition of a graph G is a pair (T, X) where T is a tree and $X = \{ X(t) : t \in V(T) \}$ is set of subsets of $V(G)$ such that:

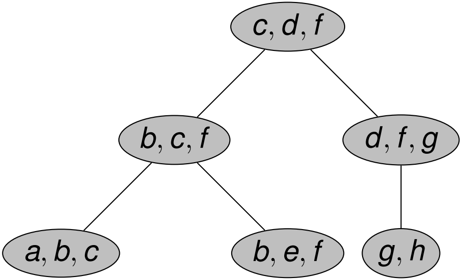
- **T1** For every $\{u, v\} \in E(G)$ there is a node $t \in V(T)$ such that $\{u, v\} \in X(t)$.
- **T2** For every $v \in V(G)$, the subgraph of T induced by $X^{-1}(v) := \{ t \in V(T) : v \in X(t) \}$ is non-empty and connected.

To distinguish between vertices of G and T, the vertices of T are called nodes. The sets $X(t)$ are also called the bags of the tree decompositon.

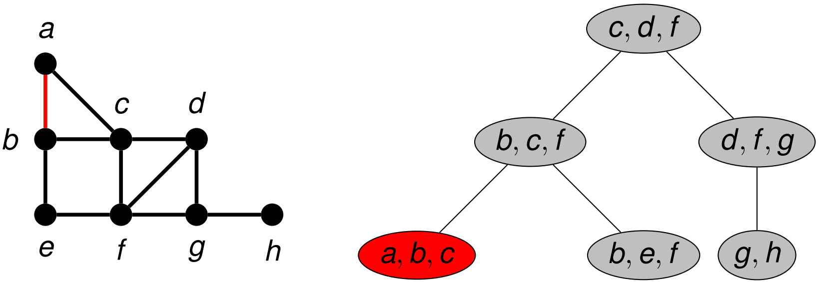
The width of a tree decomposition is $(\max_{t \in V(T)} |X(t)|) - 1$ and the treewidth of G is the smallest width of any tree decompositon of G.

Example

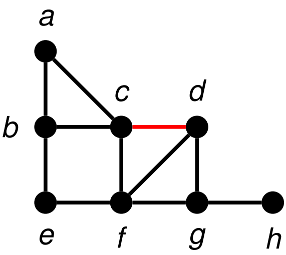


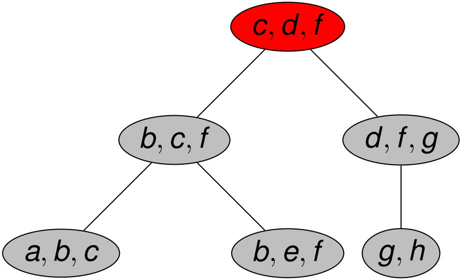


Example

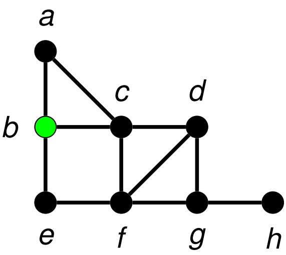


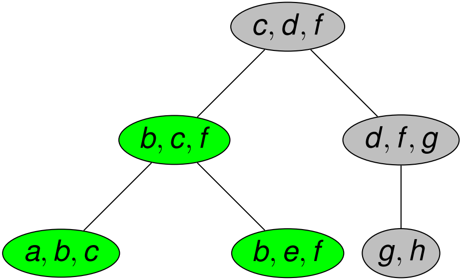
Example



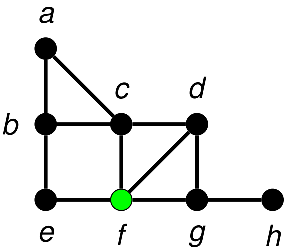


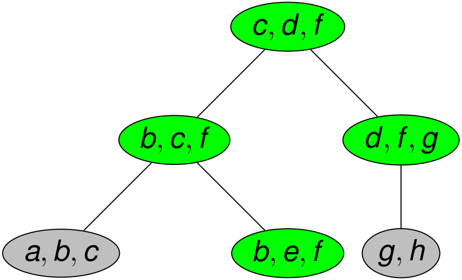
Example





Example





Basic Properties

A tree decomposition of a graph G is a pair (T, X) where T is a tree and $X = \{\, X(t) : t \in V(T) \,\}$ is set of subsets of $V(G)$ such that:

- T1 For every $\{u, v\} \in E(G)$ there is a node $t \in V(T)$ such that $\{u, v\} \in X(t)$.
- T2 For every $v \in V(G)$, the subgraph of T induced by $X^{-1}(v) := \{\, t \in V(T) : v \in X(t) \,\}$ is non-empty and connected.

Property T2 is often called the "connectedness condition" and can be equivalently formulated as:

- T2' For every $t, t', t'' \in V(T)$ such that t' lies on the unique path between t and t'' in T it holds that: $X(t) \cap X(t'') \subseteq X(t')$. Furthermore, every vertex of G is contained in some bag of T.

Basic Properties

Observation (-1)

Let G be a graph. Then $\mathrm{tw}(G) \leq |V(G)| - 1$.

Observation (0)

$\mathrm{tw}(G) = 0$ iff G contains no edges.

Observation (1)

Let H be a subgraph of a graph G. Then $\mathrm{tw}(H) \leq \mathrm{tw}(G)$.

Proof:

Let (T, X) be a tree decomposition of G. Then (T, X') such that $X(t)' := X(t) \cap V(H)$ for every $t \in V(T)$ is a tree decomposition of H whose width is at most as high as the width of (T, X). $\square$

Basic Properties

Observation (2)

Let A and B be 2 graphs and let G be the disjoint union of A and B. Then $\mathrm{tw}(G) = \max\{\mathrm{tw}(A), \mathrm{tw}(B)\}$.

Proof:

Let (T^A, X^A) and (T^B, X^B) be tree decompositions of A and B, respectively. Then (T, X) such that:

- T is the disjoint union of T^A and T^B plus an addional node r that is connected to one node of T^A and one node of T^B.
- $X(r) := \emptyset$, $X(t) := X(t)^A$ for every $t \in V(T^A)$, and $X(t) := X(t)^B$ for every $t \in V(T^B)$.

is a tree decomposition of G of width at most $\max\{\mathrm{tw}(A), \mathrm{tw}(B)\}$. □

Basic Properties

Observation (2)

Let A and B be 2 graphs and let G be the disjoint union of A and B. Then $\text{tw}(G) = \max\{\text{tw}(A), \text{tw}(B)\}$.

Corollary (1)

Let G be a graph. Then the treewidth of G is equal to the maximum treewidth of the connected components of G.

Basic Properties

Observation (2)

Let A and B be 2 graphs and let G be the disjoint union of A and B. Then $\mathrm{tw}(G) = \max\{\mathrm{tw}(A), \mathrm{tw}(B)\}$.

Corollary (1)

Let G be a graph. Then the treewidth of G is equal to the maximum treewidth of the connected components of G.

Basic Properties

Observation (3)

If G is a forest and contains at least one edge then $\text{tw}(G) = 1$.

Proof:

Because of Observation (0) it holds that $\text{tw}(G) \geq 1$. Furthermore, it follows from Corollary (1) that we only need to consider the treewidth of G's connected components, i.e., we need to show that every tree has a tree decomposition of width 1. Suppose that G is a tree. W.l.o.g. we can assume that G is rooted in some arbitrary vertex and that $p(t)$ denotes the parent of a vertex $t \in V(G)$. Then (G, X) such that $X(t) := \{t, p(t)\}$ is a tree decomposition of G of width at most 1. $\square$

Small Tree Decompostions

Definition

A tree decomposition (T, X) is small if $X(t) \not\subseteq X(t')$ for every distinct $t, t' \in V(T)$.

Proposition (1)

Given a tree decomposition of a graph G. Then in polynomial time we can construct a small tree decompositon of G (of the same width).

Proposition (2)

Let (X, T) be a small tree decomposition of G. Then $|V(T)| \leq |V(G)|$.

Small Tree Decompostions

Proposition (1)

Given a tree decomposition of a graph G. Then in polynomial time we can construct a small tree decompositon of G.

Proof:

Let (T, X) be a tree decomposition of G with $X(t) \subseteq X(t')$ for some distinct $t, t' \in V(T)$. By considering the unique path from t to t' in T we can find adjacent nodes with this property. Hence, w.l.o.g. we can assume that $\{t, t'\} \in E(T)$. Consequently, contracting the edge $\{t, t'\}$ into a new node t'' and setting $X(t'') := X(t')$ gives a smaller tree decomposition of G. Hence, we can continue this process until a small tree decomposition of G is obtained. $\square$

Small Tree Decompostions

Proposition (2)

Let (X, T) be a small tree decomposition of G. Then $|V(T)| \leq |V(G)|$.

Proof:

By induction over $n = |V(G)|$. If $n = 1$ then $|V(T)| = 1$, as required.

If $n > 1$ then consider a leaf l of T with neighbor l'. Deleting l from T yields a small tree decomposition (T', X') of $G' := G \setminus (X(l) \setminus X(l'))$.

Because $X(l) \setminus X(l') \neq \emptyset$ we obtain by induction:

$|V(T)| = |V(T')| + 1 \leq |V(G')| + 1 \leq |V(G)|$

, as required. $\square$

Minors

Observation (4)

Let H be obtained from G by contracting an edge $\{v, w\}$ into z. Then $\mathrm{tw}(H) \leq \mathrm{tw}(G)$.

Proof:

Let (T, X) be a tree decomposition of G. Then (T, X') such that $X(t)' := X(t) \cup \{z\}$ for every $t \in V(T)$ with $\{v, w\} \cap X(t) \neq \emptyset$ and $X(t)' := X(t)$, otherwise, is a tree decomposition of H whose width is at most the width of (T, X). $\square$

Minors

Definition

A graph H is a minor of a graph G if H can be obtained from a subgraph of G via edge contractions.

Because of Observation (1) and (4) we obtain:

Observation (5)

Let H be a minor of G. Then $\mathrm{tw}(H) \leq \mathrm{tw}(G)$.

Tree Decompositions and Cuts

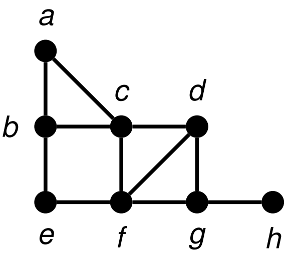
Definitions

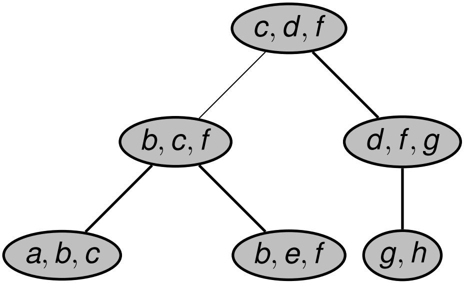
- Let (T, X) be a tree decomposition, $\{t, t'\} \in E(T)$, and $U \subseteq V(T)$. We denote by T_t and $T_{t'}$ the 2 components of $T - \{t, t'\}$ (such that T_t contains t and $T_{t'}$ contains t'). Furthermore, we denote by $X(U)$ the set of vertices $\bigcup_{t \in U} X(t)$.
- Let G be a connected graph and $S, T \subseteq V(G)$ be disjoint and non-empty vertex sets of G. A set $C \subseteq V(G)$ is a cut if $G \setminus C$ is disconnected. It is a k-cut if $|C| \leq k$. Furthermore, C is an (S, T)-cut or a cut separating S and T if $G \setminus C$ contains no paths with end vertices in both S and T.

Tree Decompositions and Cuts

Lemma

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.

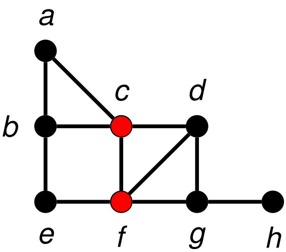


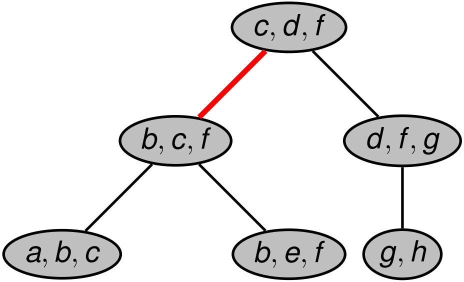


Tree Decompositions and Cuts

Lemma

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.

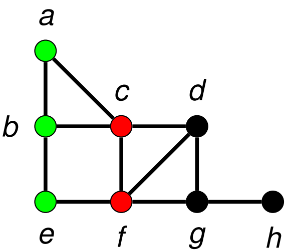


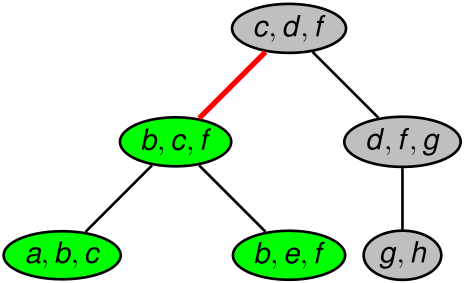


Tree Decompositions and Cuts

Lemma

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.

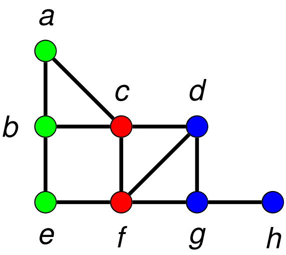


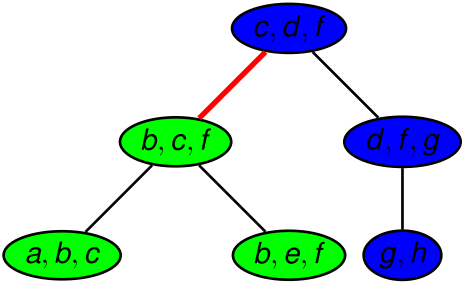


Tree Decompositions and Cuts

Lemma

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.





Tree Decompositions and Cuts

Lemma

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.

Proof:

Because of Property T2 of a tree decomposition we obtain
$C = X(t) \cap X(t') = X(T_t) \cap X(T_{t'})$.
Hence, $\{S_t, C, S_{t'}\}$ is a partition of $V(G)$. It hence suffices to show that $G \setminus C$ contains no edge $\{u, v\}$ with $u \in S_t$ and $v \in S_{t'}$.

Tree Decompositions and Cuts

Lemma (1)

Let (T, X) be a tree decomposition of a graph G and $\{t, t'\} \in E(T)$. Furthermore, let $C := X(t) \cap X(t')$, $S_t := X(T_t) \setminus X(T_{t'})$ and $S_{t'} := X(T_{t'}) \setminus X(T_t)$. Then C is an $(S_t, S_{t'})$-cut in G.

Proof, continued:

Let $\{u, v\} \in E(G)$. Because of Property T1 of a tree decomposition we know that there is a $t'' \in V(T)$ such that $\{u, v\} \subseteq X(t'')$.
If $t'' \in V(T_t)$ then $u, v \in X(T_t)$ and hence $u, v \notin X(T_{t'})$. If $t'' \in V(T_{t'})$ then $u, v \in X(T_{t'})$ and hence $u, v \notin X(T_t)$. $\square$

Tree Decompositions and Cuts

Lemma (2)

Let G be a connected graph with $\mathrm{tw}(G) \leq k$. Then $|V(G)| = k + 1$ or G has a k-cut.

Proof:

Consider a small tree decomposition (T, X) of G of width at most k. If $|V(G)| > k + 1$, then $|V(T)| \geq 2$, so we may consider any two adjacent nodes $t, t' \in V(T)$. Because (T, X) is small it holds that $X(t) \setminus X(t') \neq \emptyset$ and $X(t') \setminus X(t) \neq \emptyset$, and $|X(t) \cap X(t')| \leq k$. Then, by the previous lemma, $C = X(t) \cap X(t')$ is a k-cut in G. □

Tree Decompositions and Cuts

As an immediate consequence of Lemma (2) we obtain:

Corollary

If $\mathrm{tw}(G) = 1$, then G is a forest.

Corollary

Let K_n be the complete graph on n vertices. Then $\mathrm{tw}(K_n) = n - 1$.

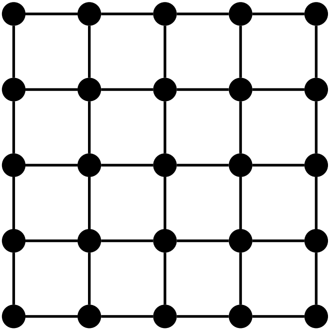
Tree Decompositions and Cuts

A $k \times l$-grid, denoted $G_{k \times l}$, is the graph with vertex set:

$$\{\,(i,j) : 1 \leq i \leq k \text{ and } 1 \leq j \leq l\,\}$$

and edge set:

$$\{\,\{(i,j),(i,j+1)\} : 1 \leq i \leq k, 1 \leq j < l\,\} \cup$$
$$\{\,\{(i,j),(i+1,j) : 1 \leq i < k, 1 \leq j \leq l\,\}$$



Tree Decompositions and Cuts

Proposition

$\mathrm{tw}(G_{k\times l}) \le \min\{k, l\}$.

As an immediate consequence of Lemma (2) we obtain:

Proposition

$\mathrm{tw}(G_{k\times l}) \ge \min\{k, l\}$.

Outline

Computing Treewidth

The following problem is NP-hard:

k-TREEWIDTH **Parameter:** k

Input: A graph G and a natural number k.
Question: Is $\mathrm{tw}(G) \leq k$ (and if so compute a tree decomposition of width at most k)

Theorem

k-TREEWIDTH is fixed-parameter tractable, i.e., there are 2 FPT-algorithms for k-TREEWIDTH: (1) $O(2^{O(k^3)}|V(G)|)$ and (2) $O(3^{3k}k(|V(G)|)^2)$.

Theorem

Treewidth can be approximated to within $k\sqrt{\log k}$.

Computing Treewidth

Remark

Because k-TREEWIDTH is fixed-parameter tractable we can always assume that we are given a tree decomposition of optimal width when designing fixed-parameter algorithms for problems parameterized by treewidth.