Bounded Treewidth Graphs – A Survey German Russian Winter School St. Petersburg, Russia

Andreas Krause – krausea@cs.tum.edu
Technical University of Munich

February 12, 2003

This survey gives an introduction to the class of *bounded treewidth* graphs, for which many $\mathcal{NP}$-complete problems can be solved efficiently. The concept of a tree decomposition is explained. Subclasses of the *bounded treewidth* graphs are identified. Results about finding tree decompositions are summarized. Some problems which are efficiently solvable on *bounded treewidth* graphs are listed and algorithms for two of them – finding maximum independent sets and the calculation of the Tutte polynomial are sketched.

1 Introduction

For computationally complex problems, e.g. $\mathcal{NP}$-complete problems, it is important to know, for which problem instances efficient algorithms are available. This survey shows, how important $\mathcal{NP}$-complete graph problems can be solved efficiently if the graph belongs to the class of bounded treewidth graphs.

Since in general trees are algorithmically comfortable to deal with, graph classes are sought which behave similarly to trees. The class of *bounded treewidth* graphs turns out to show similar properties.

1.1 Treeminology

In the following, the term graph means a pair $G = (V, E)$ of vertices V and edges E, $E \subseteq V \times V$ where E is symmetric, i.e. the graph is undirected. Graphs don't need to be simple, loops and parallel edges are allowed, i.e. E may be a multiset.

Definition 1. A tree decomposition of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$ where T is a tree with nodes X_i, $X_i \subseteq V$ for all $i \in I$ satisfying

- $\bigcup_{i \in I} X_i = V$
- For all edges $(v, w) \in E$ there exists an $i \in I$ with $v \in X_i, w \in X_i$
- For all $i, j, k \in I$ it holds that if j is on the path from i to k in T then $X_i \cap X_k \subseteq X_j$

The width of a tree decomposition $(\{X_i | i \in I\}, T = (I, F))$ is defined to be

$$\max_{i \in I} |X_i| - 1$$

and the treewidth of a graph G is the minimum width over all tree decompositions of G.

Figure 1 shows an example of graph with a tree decomposition.

1.2 Which graphs have bounded treewidth?

Several classes of graphes, which are important in practice, have been shown to have constant bounded treewidth, among them are

- Trees / Forests (treewidth 1)
- Series parallel networks (treewidth 2)
- Outerplanar graphs (treewidth 2)
- Halin graphs (treewidth 3)

It should be noted that, while the class of outerplanar graphs (graphs with an embedding in the plane such that all vertices can be placed on the outward face) has bounded treewidth, the class of *planar graphs* in general doesn't have bounded treewidth, neither has the class of *bipartite graphs*.

The *complete graph* K_n with n vertices has treewidth $n-1$, and the following result shows, that any graph containing a k-clique has at least treewidth $k-1$:

Lemma 2. *Let $G = (V, E)$ be a graph and H be a minor of G (a graph resulting from G by edge removal and edge contraction). Then*

$$\mathit{treewidth}(H) \leq \mathit{treewidth}(G)$$

1.3 Applications

According to [Bod93], graphs with *bounded treewidth* occur frequently in practical problems; among them are

- *Expert systems* – Graphs modelling certain types of expert systems have been observed to have small treewidth, which allows otherwise time-consuming statistical computations for reasoning with uncertainty.
- *Evolution theory* – In phylogeny, one wants to construct evolution trees for species, their characteristics, thereby extrapolating possible extinct ancestors. One variant of these clustering techniques is the *perfect phylogeny* problem. This can be stated as a graph problem with a vertex-colored graph. It can be shown that a necessary condition for the solution is that the treewidth of G is smaller than the number of colors.
- *Natural language processing* – It has been observed, that dependency graphs of the major syntactic relations among words typically have small tree-width, which can be exploited for language processing.

2 "Easy" problems for bounded treewidth graphs

In this section, a short summary is given on problems, for which in general no deterministic polynomial time algorithm is known, but which turn out to be efficiently solvable for *bounded treewidth* graphs. Many of the algorithms will even run in linear time.

2.1 Finding maximum independent sets

A well known $\mathcal{NP}$-complete problem is to find for a graph $G = (V, E)$ a maximum independent set, that is a subset $W \subseteq V$ of vertices such that for all $v, w \in W$ it holds that $(v, w) \notin E$.

Let $(\{X_i | i \in I\}, T = (I, F))$ be a tree decomposition of G of width k. From this tree decomposition one can easily derive a rooted

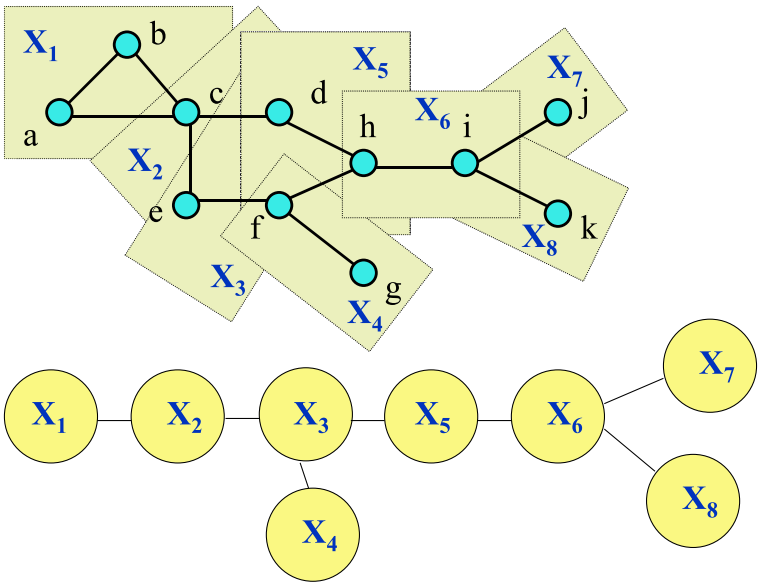


Figure 1: Tree decomposition of width 2 for example graph G. This is the minimum possible treewidth, since G contains a K_3 as a minor.

binary tree decomposition of the same width. Define for all $i \in I$:

$$Y_i = \{v \in X_j | j = i \text{ or } j \text{ is a descendant of } i\}$$

and let $G[Y_i]$ denote the vertex induced subgraph of G with vertices Y_i.

The key observation is that if one wants to extend a maximum independent set W_i of $G[Y_i]$ to a maximum independent set W of G, *only* vertices in X_i, particularly in the separator $X_i \cap X_{\text{father}(i)}$ have to be considered. Of the vertices in $Y_i \setminus X_i$, only their number is important.

This observation can be utilized in a dynamic programming approach which gives a linear time algorithm for the calculation of a maximum independent set of G. For $i \in I$ and $Z \subseteq X_i$ define $s_i(Z)$ to be the size of a maximum independent subset in $G[Y_i]$ with $W \cap X_i = Z$, or $-\infty$ if no such set exists. For leaf nodes X_i, all $2^{|X_i|}$ values of $s_i(Z)$ are found by:

$$s_i(Z) = \begin{cases} |Z|, & \text{if } \forall v, w \in Z : (v, w) \notin E \\ -\infty, & \text{if } \exists v, w \in Z : (v, w) \in E \end{cases}$$

For internal nodes i and children j and k and if $\forall v, w \in Z : (v, w) \notin E$ then

$$s_i(Z) = \max_{\substack{Z \cap X_j = Z' \cap X_i \\ Z \cap X_k = Z'' \cap X_i}} \left\{ \begin{array}{c} s_j(Z') + s_k(Z'') + \\ |Z \cap (X_i \setminus X_j \setminus X_k)| \\ -|Z \cap X_j \cap X_k| \end{array} \right\}$$

and $s_i(Z) = -\infty$ if $\exists v, w \in Z : (v, w) \in E$. Hereby, Z' and Z'' are defined as Z but for the left and right subtree of i resp. If i has no left (right) son, then Z' (Z'') is the empty set.

Theorem 3. *The above algorithm generates a maximum independent set for a graph $G = (V, E)$ if given a tree decomposition of width k in time*

$$(O)(n \cdot 2^{3k})$$

where $n = |V|$.

Proof: Simple induction by the structure of the decomposition tree, for details see [Bod93].

Remark 4. 1. The independent set itself can be reconstructed by interpretation of the dynamic programming tables s_i.

2. Although the running time of the algorithm is linear in the number of nodes, the complexity factor is very large. This is typical for algorithms exploiting bounded treewidth.

3. The above technique can be generalized to many other problems. The key idea is: n tables carry information about a *bounded* number (in k) of equivalence classes of partial solutions.

2.2 A language for graph problems

Many graph problems can be formulated within monadic second order logic, i.e. by merely using logical operations $(\vee, \wedge, \neg)$, quantification $(\exists, \forall)$, membership tests $(\in, \subseteq)$ and adjacancy tests $((v, w) \in E)$. Courcelle proved a powerful result about efficient solvability of such problems for the class of graphs with bounded treewidth:

Theorem 5 (Courcelle). *Any graph problem expressible by monadic second order logic can be solved in linear time for bounded treewidth graphs with given tree decompositions*

Proof: See [CM93].

Several extensions additionally allow solving certain optimization problems. As an example, the monadic second order logic expression of the graph three-coloring problem is:

$$\begin{aligned}&\exists W_1 \subseteq V : \exists W_2 \subseteq V : \exists W_3 \subseteq V : \\ &\forall v \in V : (v \in W_1 \vee v \in W_2 \vee v \in W_3) \wedge \\ &\forall v \in V : \forall w \in V : (v, w) \in E \Rightarrow \\ &(\neg(v \in W_1 \wedge w \in W_1) \wedge \neg(v \in W_2 \wedge \\ &w \in W_2) \wedge \neg(v \in W_3 \wedge w \in W_3))\end{aligned}$$

2.3 Other solvable problems for bounded treewidth graphs

There are other problems, which are in general computationally complex and which turn out to be efficiently solvable for graphs with bounded treewidth. The techniques used differ from the dynamical programming approach above. These problems include:

- *Graph isomorphism* – Given two graphs H and K, does there exist a permutation of the vertices of H s.t. the permuted graph H' is identical to K?

- *Recognition of graph classes* – Using the deep graph theoretic results on graph minors by Seymour and Robertson, one can prove that every graph class which doesn't contain all planar graphs and which is closed under taking minors can be recognized in linear time.

- *Tutte polynomial and related problem* – The Tutte polynomial of a graph G allows to determine important combinatorial quantities (e.g. no. of spanning trees, chromatic polynomial, nowhere-zero flows,...) by specializations. For general graphs, it is $\mathcal{NP}$-hard to compute the Tutte polynomial; for graphs with *bounded treewidth* , a polynomial algorithm is sketched in section 4.

3 Finding tree decompositions

To utilize that a given graph has bounded treewidth, it is important to know a (optimal) tree decomposition. For general graphs it can be shown, that it is a $\mathcal{NP}$-complete problem to determine their treewidths. However, a theorem of Bodlaender states:

Theorem 6 (Bodlaender). *For all positive integers k there exists a linear-time algorithm that tests whether a given graph G has treewidth at most k, and if so, outputs a tree decomposition of G of width at most k.*

The important point is, that k is being regarded as a constant and *not* belonging to the

problem instance. Additionally there are polynomial time approximation algorithms to determine the treewidth of a given graph which have a performance ratio of $\mathcal{O}(\log n)$.

3.1 Complexity of determining treewidth for special graph classes

Although the general problem of determining the treewidth of general graphs is $\mathcal{NP}$-complete , there are certain graph classes, for which their treewidth can be calculated efficiently.

- *Linear time* – For each graph class with bounded tree width, its treewidth can be computed in linear time.
- *Polynomial time* – There exist polynomial time algorithms to determine the treewidth of e.g. chordal graphs, interval graphs and circle graphs.
- *$\mathcal{NP}$-complete problem* – The problem of determining the treewidth is $\mathcal{NP}$-complete even for e.g. graphs with bounded degree and for bipartite graphs.
- *Open problem* – Whether the treewidth of planar graphs can be found in polynomial time is still an open problem.

4 The Tutte polynomial

The Tutte polynomial is a result of algebraic graph theory; it is an invariant derived from a graphs spanning trees and the associated quantities of fundamental cycles and cocycles. Specializations of the Tutte polynomial allow to compute important combinatorial quantities of a graph, e.g. its chromatic polynomial (and thus its chromatic number), the number of spanning trees, nowhere-zero-flows, the reliability.

4.1 Definitions

Some terminology has to be established.

Definition 7. Let $G = (V, E)$ a graph, $\leq$ a total ordering on the edges E and let T be a spanning tree of G.

An edge $e \in G \setminus T$ is called *externally active* if it is the largest edge in the unique cycle $T \cup \{e\}$

$$\mathrm{ea}(T) = |\{e \in G \setminus T | e \text{ externally active}\}|$$

An edge $e \in T$ is called *internally active* if it is the largest edge in the unique cocycle $(G \setminus T) \cup \{e\}$

$$\mathrm{ia}(T) = |\{e \in G \setminus T | e \text{ internally active}\}|$$

A cocycle $C' \subseteq E$ in G is a minimal disconnecting subset of G with edges contained in C'.

Figure 2 gives an examples of the above definitions.

Definition 8. The Tutte polynomial of a graph G and a total ordering $\leq$ on its edges is defined as

$$t(G) = t(G;\ \leq;\ x, y) = \sum_{\substack{T \subseteq G \\ T \text{sp. tree}}} x^{\mathrm{ia}(T)} y^{\mathrm{ea}(T)}$$

Theorem 9 (Tutte). *The definition of the Tutte polynomial is independent of the total ordering $\leq$, i.e. it is well-defined.*

Proof: Is given in [Big93].

There is also a recursive characterization of the Tutte polynomial:

Theorem 10. *The Tutte polynomial of a graph $G = (V, E)$ can be found recursively: If $E = \emptyset$ then $t(G) = 1$; if $e \in E$ then:*

(R1) $t(G) = t(G \setminus \{e\}) + t(G/\{e\})$ *if e is not a loop or a bridge*

(R2) $x \cdot t(G \setminus \{e\})$, *if e is a bridge*

(R3) $y \cdot t(G \setminus \{e\})$, *if e is a loop*

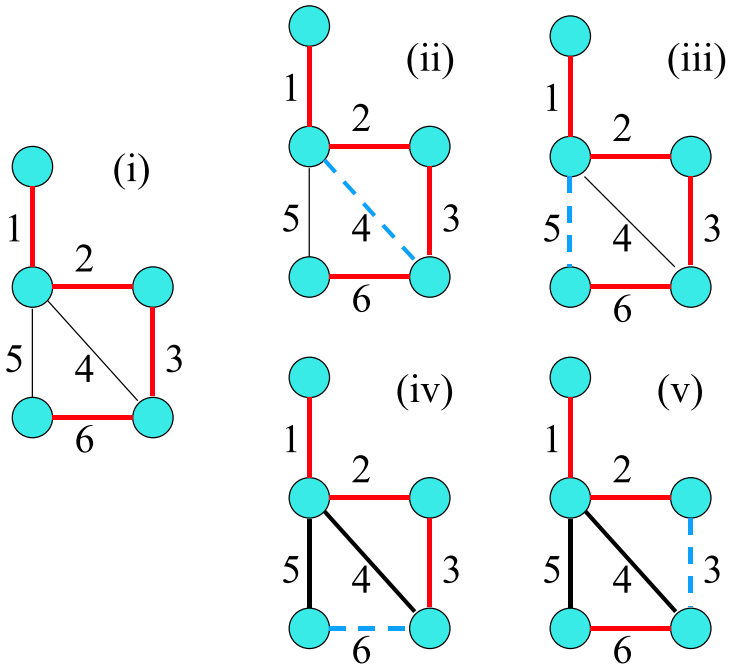


Figure 2: An example of the above definitions. In (i) a graph G is given together with a spanning tree T (bold red edges), a total ordering is given by edge weights. In (ii), the cycle in $T \cup \{4\}$ consists of edges $\{2,3,4\}$. The largest edge is edge 4, thus 4 is externally active. In (iii), 5 is not externally active since 6 is the largest edge in {2,3,5,6}. In (iv), the cocycle induced by edge 6 is $\{5,6\}$, thus 6 is internally active. In (v), the cocycle induced by 3 is {3,4,5}, thus 3 is not internally active. It can easily be seen, that 1 is internally active, while 2 is not, thus $\mathrm{ea}(T) = 1$ and $\mathrm{ia}(T) = 2$.

Proof: Is given in [BO92]

Hereby is $G \setminus \{e\}$ the graph obtained from G by removal of edge e and $G/\{e\}$ the graph obtained from G by contraction of edge e (removal of edge e and identification of adjacent vertices).

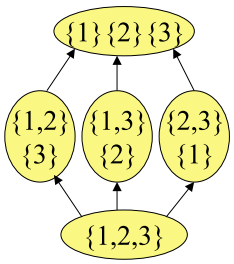
It can easily be seen that the naive calculation of the Tutte polynomial of a complete graph by using the above recursion would take exponential time in $|E|$. In general it has been shown that evaluating the Tutte polynomial for planar graphs at general points (excluding some special conditions) is $\#\mathcal{P}$ hard [Ver].

4.2 Divide and conquer

To utilize the tree decomposition of a graph with bounded treewidth, one can use a divide and conquer strategy as sketched below. Let $K = (V, E)$, $H = (W, F)$ be two graphs, $E \cap F = \emptyset$ and $G = K \cup H$ their graph union. Call $U = V \cap W$ the separator of K and H and set $r = |U|$. In the case $r = 1$, it can be seen that $t(G) = t(K) \cdot t(H)$. For $r \geq 2$, Negami proved the validity of a *splitting formula* which relates the Tutte polynomial of G to Tutte polynomials derived from graphs related to K and H.

4.3 Partition of the separator

Let $\Gamma(U)$ be the lattice of partitions of U and write $P_1 \preceq P_2$ iff P_2 is a refinement of P_1, i.e. every block of P_1 is a union of blocks of P_2. The top of the lattice P^1 (the unique maximal element w.r.t. $\preceq$) is the partition consisting of $|U|$ singleton blocks. For $P_1, P_2 \in \Gamma(U)$ let $P_1 \wedge P_2$ denote their meet, i.e. the greatest lower bound of P_1 and P_2. The r-th Bell number $s(r)$ is the number of partitions in $\Gamma(U)$.



The partition lattice $\Gamma(\{1,2,3\})$. *The arrows denote* $P_1 \to P_2$ *iff* $P_1 \preceq P_2$. *Thus* $s(3) = 5$.

For graph K and partition $P \in \Gamma(U)$ let $K /\!\!/ P$ denote the graph obtained from K by identification of vertices within the same block of P. Figure 3 gives an example of such an operation.

4.4 The splitting formula

Let $K = (V, E)$, $H = (W, F)$ be two graphs, $E \cap F = \emptyset$ and $G = K \cup H$, $U = V \cap W$ and $r = |U|$. The splitting formula states, that the Tutte polynomial of G can be calculated by simple matrix vector multiplication (where the matrices and vertices are over the ring of polynomials of two variables), if the Tutte polynomials and number of connected components of all graphs, which are obtained from K and H by partition identification are known ($c(G)$ denotes the number of connected components of graph G).

Theorem 11 (Negami). *Let* T_r, C_r *be* $s(r) \times s(r)$ *matrices,*

$$\begin{aligned}(T_r)_{i,j} &= [(x-1)(y-1)]^{|P_1 \wedge P_2|} \\ (C_r)_{i,j} &= (y-1)^{|P_1|+|P_2|-r} \cdot (T_r^{-1})_{i,j}\end{aligned}$$

and let k_r, h_r *be* $s(r)$ *vectors,*

$$k_r = \begin{bmatrix} (x-1)^{c(K/\!\!/P_1)} t(K /\!\!/ P_1) \\ \vdots \\ (x-1)^{c(K/\!\!/P_{s(r)})} t(K /\!\!/ P_{s(r)}) \end{bmatrix}$$

$$h_r = \begin{bmatrix} (x-1)^{c(H/\!\!/P_1)} t(H /\!\!/ P_1) \\ \vdots \\ (x-1)^{c(H/\!\!/P_{s(r)})} t(H /\!\!/ P_{s(r)}) \end{bmatrix}$$

Then it holds that

$$t(G) = (x-1)^{-c(G)} k_r^T C_r h_r$$

Proof: Given in [Neg87]

4.5 An algorithm for the Tutte polynomial

This splitting formula gives rise to a divide and conquer algorithm: Suppose a graph G has binary tree decomposition $(\{X_i | i \in I\}, T = (I, F))$ and suppose, an internal node i has two children j and k (the other cases are simpler. Let $V = Y_{ls(i)}$ and $W = Y_{rs(i)} \cup X_i$ with notation as in 2.1, let K and H' be the vertex induced subgraphs of V and W resp. and let H be the graph obtained from H' by removal of all edges in the separator U. Then K and H fulfill the requirements of Theorem 11 – the partition of G is shown in Fig. 4. To determine vectors h_r and k_r, $2s(r)$ Tutte polynomials and numbers of connected components have to be determined. The maximal recursion depth is the height of the tree T. The following lemma allows to bound this height.

Lemma 12 (Bodlaender). *Let* k *be constant. Given a tree decomposition of width* k *and a graph* G *with* n *vertices, one can compute a rooted, binary tree decomposition of* G *of depth at most* $2\lceil \log_{5/4}(2n) \rceil$ *and width at most* $r = 3k + 2$ *in time* $\mathcal{O}(n)$ *(using a sequential algorithm).*

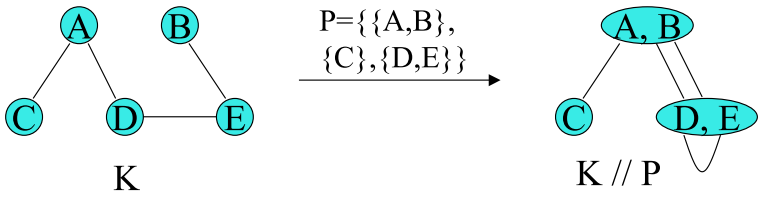


Figure 3: An example of the graph partition identification $K /\!\!/ P$

Since the recursion depth is logarithmic in the number of vertices and the maximal branching is bound by a constant (i.e. $2s(r+1)$), it can be seen, that the number of recursive calls is polynomial in the number of vertices. Within each recursive call, the matrix vector multiplications are done in constant time (in n) and the number of connected components of all graphs can be determined in polynomial time, i.e. in $\mathcal{O}(n^2)$ time.

There are two cases left: interior nodes with one son and leaf nodes. Suppose i is an internal node of T and j is its only child. Then let $V = Y_j$, $W = X_i$ and K, H as above. For H, vector h_r can be computed using rules (R1), (R2) and (R3), since the number of edges in H is bounded by $r^2/2$. For K, k_r is calculated recursively, which can be done in polynomial time as shown above. For leaf nodes i, the Tutte polynomial of $G[X_i]$ can be calculated in constant time using rules (R1), (R2) and (R3).

A detailed formulation of the algorithm including a proof of a polynomial running time bound can be found in [And98]. In his paper, the following theorem is established using the algorithm sketched above:

Theorem 13 (Andrzejak). *For each positive integer k there is an algorithm which decides in linear time if a given graph G with n vertices has treewidth at most k and if so, computes the Tutte polynomial of G in total time $\mathcal{O}(n^{2+7\log_2(s(3k+3))})$.*

Remark 14. Although the result of Andrzejak gives a polynomial time algorithm for the calculation of the Tutte polynomial for graphs with *bounded treewidth* , the exponents of the polynomial are very large. The table below gives the exponent $e(k)$ for fixed values of k

k	2	3	4	5
e(k)	97	145	197	252

5 Further remarks

5.1 Locally bounded treewidth

As the fact, that even sparse graphs such as planar graphs do not have bounded treewidth, one is interested in generalizations of the concept of bounded treewidth. One such generalization is Eppsteins concept of locally bounded treewidth (s. [Epp99]).

Definition 15. A graph $G = (V, E)$ is said to have *locally bounded treewidth* if for each vertex $v \in V$ the treewidth of the vertex induced subgraph with vertices of distance less than r is bounded by a function of r.

It can be shown, that planar graphs have small locally bounded treewidth. Some algorithms for graphs with *bounded treewidth* can be extended to graphs with locally bounded treewidth, e.g. an algorithm for subgraph isomorphism (s. [Haj]).

5.2 Tree decompositions and graph minors

The concept of bounded treewidth and tree decompositions was first introduced by

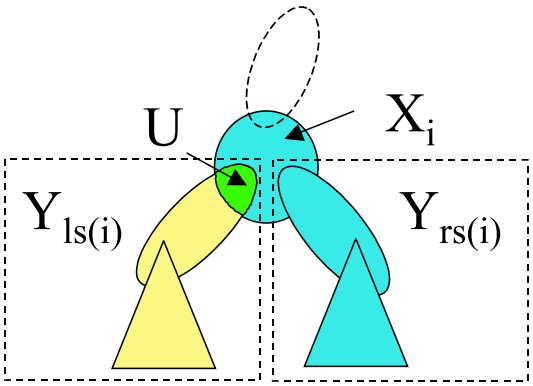


Figure 4: Partition of graph G in terms of its tree decomposition

Robertson and Seymour in their famous series of papers about graph minors (s. [RS86]). In 1960, Kruskal proved that trees are quasi-well-ordered (i.e. every infinite set of trees contains trees T_1 and T_2 such that T_1 is a minor of T_2). Seymour and Robertson proved in 1990 that for a fixed integer k, graphs with treewidth less than k are quasi-well-ordered. This was an important step towards proving their Minor Theorem, a deep result of graph theory which states, that finite graphs are quasi-well-ordered under the minor relation. For an overview of the complex of Minor Theory, see [Die00].

References

[And98] A. Andrzejak. An algorithm for the tutte polynomials of graphs of bounded treewidth. *Disc Math*, 190, 1998.

[Big93] N. Biggs. *Algebraic Graph Theory*, chapter 13, The Tutte Polynomial. Cambridge Univ. Press, 2nd edition, 1993.

[BO92] T. Brylawski and J. Oxley. The tutte polynomial and its applications, matroid applications. *Encycl Math Appl*, 40:123–225, 1992.

[Bod93] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybern*, 11, 1993.

[CM93] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoret Comp Sci*, 109:49–82, 1993.

[Die00] R. Diestel. *Graphentheorie*, chapter 10., Minoren, Baeume und WQO, pages 252–279. Springer, 2000.

[Epp99] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3, 1999.

[Haj] M. Hajiaghayi. Subgraph isomorphism, log-bounded fragmentation and graphs of (locally) bounded treewidth.

[Neg87] S. Negami. Polynomial invariants of graphs. *Trans Am Math Soc*, 299:601–672, 1987.

[RS86] N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journ Alg*, 7:309–322, 1986.

[Ver] V. L. Vertigan. The computational complexity of tutte invariants for planar graphs. *to appear*.