European Journal of Combinatorics 58 (2016) 61–65

Contents lists available at ScienceDirect

European Journal of Combinatorics

journal homepage: www.elsevier.com/locate/ejc

A short proof that every finite graph has a tree-decomposition displaying its tangles

Johannes Carmesin

University of Cambridge, Wilberforce Road, Cambridge CB3 0WB, United Kingdom

ARTICLE INFO

Article history:
Received 10 December 2015
Accepted 26 April 2016
Available online 8 June 2016

ABSTRACT

We give a short proof that every finite graph (or matroid) has a tree-decomposition that displays all maximal tangles.

This theorem for graphs is a central result of the graph minors project of Robertson and Seymour and the extension to matroids is due to Geelen, Gerards and Whittle.

1. Introduction

Robertson and Seymour [5] proved as a corner stone of their graph minors project:

Theorem 1.1 (*Rough Version*)**.** *Every graph*[1] *has a tree-decomposition whose separations distinguish all maximal tangles.*

Additionally, it can be ensured that this tree-decomposition separates the tangles in a 'minimal way'. This theorem was extended to matroids by Geelen, Gerards and Whittle [4]. Here we give a short proof of both of these results. A key idea is that we prove the following strengthening:

Theorem 1.2 (*Rough Version of* Theorem 2.4)**.** *Any tree-decomposition such that each of its separations distinguishes two tangles in a minimal way can be extended to a tree-decomposition that distinguishes any two maximal tangles in a minimal way.*

Our new proof does not yield the strengthening of Theorem 1.1 proved in [2]. However, it can be extended from tangles to profiles, compare Remark 4.1. For tree-decompositions as in Theorem 1.1 that additionally have as few parts as possible see Corollary 4.3.

E-mail address: j.carmesin@dpmms.cam.ac.uk.

1 In this paper all graphs and matroids are finite.

http://dx.doi.org/10.1016/j.ejc.2016.04.007
0195-6698/

2. Notation

Throughout we fix a finite set E. A *separation* is a bipartition (A, B) of E, and A and B are called the *sides* of (A, B). A function f mapping subsets of E to the integers is *symmetric* if $f(X) = f(X^\complement)$ for every $X \subseteq E$, and it is *submodular* if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for every $X, Y \subseteq E$. Throughout we fix such a function f. Since f is symmetric, it induces a function o on the separations: $o(A, B) = f(A) = f(B)$, which we call the *order* of a separation.[2] Since f is submodular o satisfies:

$$o(A, B) + o(C, D) \geq o(A \cap C, B \cup D) + o(A \cup C, B \cap D). \tag{1}$$

For example, one can take for E the edge set of a matroid and for f its connectivity function. Or one can take for E the edge set of a graph, where the order of a separation (A, B) is the number of vertices incident with edges from both A and B.

A *tangle* of order $k + 1$ picks a *small* side of each separation (A, B) of order at most k such that no three small sides cover E. Moreover, the complement of a single element of E is never small.[3] In particular, if A is small, then its complement B cannot be included in a small set and we say that B is *big*. Thus a tangle can be thought of as pointing towards a highly connected piece, which 'lies' on the big side of every low of order separation. In this spirit, we shall also say that a tangle $\mathcal{T}$ *orients* a separation (A, B) towards B if B is big in $\mathcal{T}$.

A tangle is *maximal* if it is not included in any other tangle (of higher order). A separation (A, B) *distinguishes* two tangles if these tangles pick different small sides for (A, B). It distinguishes them *efficiently* if it has minimal order amongst all separations distinguishing these two tangles.

A *tree-decomposition* consists of a tree T and a partition $(P_t | t \in V(T))$ of E consisting of one (possibly empty) partition class for every vertex of T. For $X \subseteq V(T)$, we let $S(X) = \bigcup_{t \in X} P_t$. There are two separations *corresponding* to each edge e of T, namely $(S(X), S(Y))$ and $(S(Y), S(X))$. Here X and Y are the two components of $T - e$. We say that a tree-decomposition *distinguishes two tangles efficiently* if there is a separation corresponding to an edge of the decomposition-tree distinguishing these tangles efficiently.

The following implies Theorem 1.1 and its matroid counterpart mentioned in the Introduction if we plug in the particular choices for the order function mentioned above.[4]

Theorem 2.1. *Let E be a finite set with an order function. Then there is a tree-decomposition distinguishing any two maximal tangles efficiently.*

Two separations (A_1, A_2) and (B_1, B_2) are *nested*[5] if $A_i \subseteq B_j$ for some pair $(i, j) \in \{1, 2\} \times \{1, 2\}$. A set of separations is *nested* if any two separations in the set are nested. A set of separations N is *symmetric* if $(A, B) \in N$ if and only if $(B, A) \in N$. Note that any nested set N is contained in a nested symmetric set, which consists of those separations (A, B) such that (A, B) or (B, A) is in N. It is clear that:

Remark 2.2. Given a tree-decomposition, the set of separations corresponding to the edges of the decomposition-tree is nested and symmetric.

The converse is also true:

Lemma 2.3 ([4]). *For every nested symmetric set N of separations, there is a tree-decomposition such that the separations corresponding to edges of the decomposition-tree are precisely those in N.*

---

2 For the sake of readability, we write $o(A, B)$ instead of $o((A, B))$.

3 This 'moreover'-property is never used in our proofs and thus the results are also true for a slightly bigger class. However, the new objects are trivial.

4 In [5], the authors use a slightly different notion of separation for graphs. From a separation (A, B) in the sense of this paper, the corresponding separation in their setting is $(V(A), V(B))$, where $V(X)$ denotes the set of vertices incident with edges from X. However, it is well-known that these two notions of separations give rise to the same notion of tangle and so Theorem 2.1 implies their version.

5 Other authors use *laminar* instead.