Graphs, Branchwidth, and Tangles! Oh My!

Illya V. Hicks
Industrial Engineering, Texas A & M University, Zachry Engineering Center, College Station, Texas 77843-3131

Branch decomposition-based algorithms have been used in practical settings to solve some NP-hard problems like the travelling salesman problem (TSP) and general minor containment. The notions of branch decompositions and branchwidth were introduced by Robertson and Seymour to assist in proving the Graph Minors Theorem. Given a connected graph G and a branch decomposition of G of width k where k is at least 3, a practical branch decomposition-based algorithm to test whether a graph has branchwidth at most $k-1$ is given. The algorithm either constructs a branch decomposition of G of width at most $k-1$ or constructs a tangle basis of order k, which offers a lower bound on the branchwidth of G. The algorithm is utilized repeatedly in a practical setting to find an optimal branch decomposition of a connected graph, whose branchwidth is at least 2, given an input branch decomposition of the graph from a heuristic. This is the first algorithm for the optimal branch decomposition problem for general graphs that has been shown to be practical. Computational results are provided to illustrate the effectiveness of finding optimal branch decompositions. A tangle basis is related to a tangle, a notion also introduced by Robertson and Seymour; however, a tangle basis is more constructive in nature. Furthermore, it is shown that a tangle basis of order k is coextensive to a tangle of order k. © 2004 Wiley Periodicals, Inc. NETWORKS, Vol. 45(2), 55–60 2005

Keywords: branchwidth; branch decomposition; tangle; tangle basis

1. INTRODUCTION

Robertson and Seymour [14] introduced the notion of branch decompositions and branchwidth to assist in their proof of the Graph Minors Theorem. However, branch decompositions are of algorithmic importance, because Courcelle [7] and Arnborg et al. [2] showed that several NP-hard problems can be solved in polynomial time using dynamic programming techniques on input graphs with bounded branchwidth, which are referred to as branch decomposition-based algorithms. These results were originally proved for graphs with bounded treewidth, a notion also introduced by Robertson and Seymour [13]; however, treewidth and branchwidth bound each other by constants [14].

Despite the promising theoretical algorithmic results of Courcelle [7] and Arnborg et al. [2], there has been little effort in developing practical branch decomposition-based and tree decomposition-based algorithms. As for branch decomposition-based algorithms, the work of Cook and Seymour [6] on the TSP and the work of Hicks [9] on general minor containment are the most notable. One is also referred to the work of Christian [4]. For practical tree decomposition-based algorithms, one is referred to the work of Koster et al. [12], Telle and Proskurowski [17], and Alber and Neidermeier [1]. Bodlaender and Thilikos [3] also gave a tree decomposition-based algorithm for finding optimal branch decompositions but it appears to be impractical. These examples illustrate the potential of branch decomposition-based and tree decomposition-based algorithms for other NP-hard problems. A more thorough survey of relevant literature on this research topic can be found in Hicks [9].

Computing the branchwidth of a general graph is NP-hard [16]. Thus, one has to rely on heuristics to find branch decompositions for most practical branch decomposition-based algorithms unless the graph is planar where there exists a polynomial time algorithm [10, 11, 16]. This article offers a practical branch decomposition-based algorithm to test whether an input graph has branchwidth at most $k-1$ for some integer k at least 3. The algorithm will either find a branch decomposition whose width is at most $k-1$ or find a *tangle basis* of order k. This algorithm is utilized repeatedly in a practical setting to find an optimal branch decomposition of a connected graph G whose branchwidth is at least 2, given an input branch decomposition of width at least 3 for G by a heuristic. The notion of a tangle basis is related to tangles, which share a min–max relationship with the branchwidth of graphs with branchwidth at least 2 [14]. This article also shows that tangles and tangle bases are coextensive for graphs with branchwidth at least 2.

Foundational definitions are given in Section 2. Section 3 introduces the notion of tangle basis and its relevant importance. Section 4 offers the descriptions of the branch

Received August 2003; accepted November 2004
Correspondence to: I.V. Hicks; e-mail: ivhicks@tamu.edu
Contract grant sponsor: NSF; Contract grant number: DMI-0217265
DOI 10.1002/net.20050
Published online in Wiley InterScience (www.interscience.wiley.com).

decomposition algorithm, and Sections 5 and 6 are reserved for computational results and conclusions, respectively.

2. BRANCHWIDTH

Let G be a graph (or hypergraph) with node set $V(G)$ and edge set $E(G)$. Let T be a tree having $|E(G)|$ leaves in which every nonleaf node has degree 3. Let ν be a bijection (one-to-one and onto function) from the edges of G to the leaves of T. The pair (T, ν) is called a *branch decomposition* of G. Notice that removing an edge, say e, of T partitions the leaves of T and the edges of G into two subsets A_e and B_e. The *middle set* of e and of (A_e, B_e), denoted by $mid(e)$ or $mid(A_e, B_e)$, is the set $V(G[A_e]) \cap V(G[B_e])$ where $G[A_e]$ is the subgraph of G induced by A_e and similarly for $G[B_e]$. The *width* of a branch decomposition (T, ν) is the maximum order of the middle sets over all edges in T. The *branchwidth* of G, denoted by $\beta(G)$, is the minimum width over all branch decompositions of G. A branch decomposition of G is *optimal* if its width is equal to the branchwidth of G. For example, Figure 1 gives an optimal branch decomposition of an example graph where some of the middle sets of the edges of the branch decomposition are provided. The pair (T, ν) is called a *surjection branch decomposition* if ν is only a surjection (onto) function instead of a bijection.

Graphs of small branchwidth are characterized by the following theorem.

Theorem 1 ***(Robertson and Seymour [14]).*** *A graph G has branchwidth $(a.)$ 0 if and only if every component of G has ≤ 1 edge; $(b.) \leq 1$ if and only if every component of G has ≤ 1 node of degree ≥ 2; $(c.) \leq 2$ if and only if G has no K_4 minor.* □

Other classes of graphs with known branchwidth are grids and complete graphs. The branchwidth of an $a \times b$-grid is the minimum of a and b while the branchwidth of a complete graph G with at least 3 nodes is $\lceil (2/3)\ |V(G)| \rceil$.

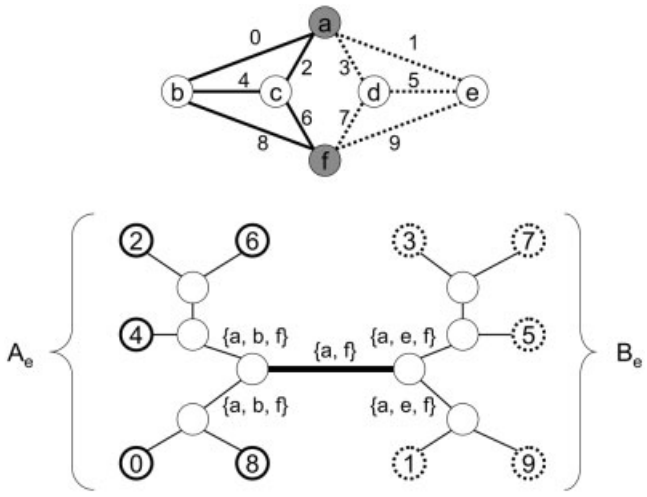

FIG. 1. Example graph G with optimal branch decomposition (T, ν) with width 3.

3. TANGLE BASES

Let G be a graph (or hypergraph) and let $k \geq 1$ be an integer. A *separation* of a graph G is a pair (G_1, G_2) of subgraphs of G with $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2)) = G$, $E(G_1) \cap E(G_2) = \emptyset$ and the *order* of this separation is defined as $|V(G_1) \cap V(G_2)|$ where $V(G_1) \cap V(G_2)$ is called the *middle set* of the separation. For a hypergraph G, define $I(G)$ to be the bipartite graph such the nodes of $I(G)$ correspond to the nodes and edges of G and an edge ev in $I(G)$ corresponds to the edge e of G being incident with the node v in G. A hypergraph G is called *connected* if $I(G)$ is connected. Also, let $\gamma(G)$ denote the largest cardinality of a set of nodes incident to an edge of G. Thus, for an edge e, $\gamma(e)$ is the number of nodes incident with e. A *tangle* in G of *order* k is a set $\mathcal{T}$ of separations of G, each of order $< k$, such that:

(T1) for every separation (A, B) of G of order $< k$, one of (A, B), (B, A) is an element of $\mathcal{T}$;

(T2) if (A_1, B_1), (A_2, B_2), $(A_3, B_3) \in \mathcal{T}$ then $A_1 \cup A_2 \cup A_3 \neq G$; and

(T3) if $(A, B) \in \mathcal{T}$ then $V(A) \neq V(G)$.

These are called the first, second, and third tangle axioms. The *tangle number* of G, denoted by $\theta(G)$, is the maximum order of any tangle of G. Figure 2 gives an example of a tangle of order 3 for the graph in Figure 1. Notice in Figure 2 that the inclusion of separations of the graph of order 3 in the tangle would result in a violation of one of the tangle axioms. A tangle $\mathcal{T}$ of G with order k can be thought of as a "k-connected" component of G because some "k-connected" component of G will either be on one side or the other for any separation of $\mathcal{T}$. Robertson and Seymour [14] proved a strong min–max relation between tangles and branch decompositions which is given below.

Theorem 2 ***(Robertson and Seymour [14]).*** *For any hypergraph G such that $E(G) \neq \emptyset$, $\max\{\beta(G), \gamma(G)\} = \theta(G)$.* □

For our algorithm we will need a more concrete version of a tangle. For an integer k and hypergraph G, a *tangle basis* $\mathcal{B}$ of order k is a set of separations of G with order $< k$ such that:

Separation of order 0
$(\emptyset, G)$
Separations of order 1
$(v, G)\ \forall\ v \in V(G)$
Separations of order 2
$(\{v, w\}, G)\ \forall\ v, w \in V(G)$
$(G[e], G[E(G) \setminus e])\ \forall\ e \in E(G)$
$(G[0, 2, 4, 6, 8], G[1, 3, 5, 7, 9])$

FIG. 2. Tangle of order 3 for the graph of Figure 1.

(B1) $(G[e], G[E(G) \setminus e]) \in \mathcal{B}, \forall\, e \in E(G)$ if $\gamma(e) < k$

(B2) if $(C, D) \in \mathcal{B}$ and $\nexists\, e \in E(G)$ such that $G[e] = C$ if and only if $\exists\ (A_1, B_1),\ (A_2, B_2) \in \mathcal{B}$ such that $A_1 \cup A_2 = C$ and $B_1 \cap B_2 = D$

(B3) $\mathcal{B}$ obeys the tangle axioms T2 and T3.

We will refer to B1, B2, and B3 as the tangle basis axioms. The concept for a tangle basis comes from two lemmas about tangles given by Robertson and Seymour [14], which are the following.

Lemma 1 ***(Robertson and Seymour [14]).*** *Let $\mathcal{T}$ be a set of separations of a hypergraph G, each of order $< k$, some integer, and satisfying T1 and T2. Then $\mathcal{T}$ is a tangle if and only if $(e, G \setminus e) \in \mathcal{T}$ for every $e \in E(G)$ such that $\gamma(e) < k$.* □

Lemma 2 ***(Robertson and Seymour [14]).*** *If $\mathcal{T}$ is a tangle of order k and (A, B), $(A', B') \in \mathcal{T}$ and $(A \cup A', B \cap B')$ has order $< k$ then $(A \cup A', B \cap B') \in \mathcal{T}$.* □

A tangle basis $\mathcal{B}$ in G of order k is *connected* if every separation (A, B) of $\mathcal{B}$ has A connected. Define the *connected tangle basis number* of G, denoted by $\theta'(G)$, as the maximum order of any connected tangle basis of G. An example of a connected tangle basis for the graph in Figure 1 is given by $(G[e],\ G[E(G) \setminus e])\ \forall\ e \in E(G)$. Notice that the number of separations of the connected tangle basis of Figure 1 is lower than the number of separations of the tangle of Figure 1 offered by Figure 2 but still contains the essential members of the tangle. This will be helpful in the speedup of the algorithm. In addition, there are certain separations that the tangle basis prohibits. These types of separations (C, D) satisfy the order restriction of a tangle basis but the separation cannot be constructed from other members of the tangle basis using the tangle basis axiom B2. For example, the separation $(G[0, 2, 4, 6, 8], G[1, 3, 5, 7, 9])$ is excluded from the connected tangle basis for the graph of Figure 1. Below, a min–max theorem relating connected tangle bases and branchwidth is offered.

Theorem 3. *If hypergraph G is connected such that $\beta(G) \geq \gamma(G)$, then $\beta(G) = \theta'(G)$.*

Proof. Clearly, $\theta'(G) \geq \theta(G)$ by Theorem 2 and Lemmas 1 and 2. To prove $\theta'(G) \leq \beta(G)$, we will show that an optimal branch decomposition of G bounds $\theta'(G)$ by showing that for any connected subgraph C $(E(C) \neq \emptyset)$ of G, the separation $(C, G[E(G) \setminus E(C)])$ is in the tangle basis of order $\theta'(G)$ (including $(G, \emptyset)$) if $\theta'(G) > \beta(G)$.

Suppose $\theta'(G) > \beta(G)$ and let $\mathcal{B}$ be a connected tangle basis of order $\theta'(G)$ and let (T, ν) be an optimal branch decomposition of G. Let st be an edge of T. Subdivide st by deleting st and introducing a node p and edges sp and pt. Root T at p. So every nonleaf node u has two children, l and r. For tree node u, denote by G_u the subgraph induced by the edges of G corresponding to the leaves of T in the subtree rooted at u. For a tree node u, let B_u denote the subset of separations (A, B) of $\mathcal{B}$ such that A (connected) is a subgraph of G_u. So for a leaf w of T, B_w would only consist of $(\nu^{-1}(w),\ G[E(G) \setminus \nu^{-1}(w)])$, by tangle basis axiom B1. For a nonleaf tree node u that is not p, let (C, D) denote a separation of G such that C is a connected subgraph of G_u. Because $\theta'(G) > \beta(G)$, the subtree of T rooted at the least common ancestor (common ancestor with greatest depth) of the leaves of T corresponding to the edges of C dictate a way to use the tangle basis axiom B2 to derive C from members of $B_l \cup B_r$. Thus, B_u will contain separations (C, D) such that C is a connected component of G_u. For the root p, B_p will contain separations (C, D) such that C is a connected component of G_s and B_p will also contain separations (F, H) such that F is a connected component of G_t. Let (C_s, D_s) and (C_t, D_t) be members of B_p such that C_s is a connected component of G_s, C_t is a connected component of G_t, and $V(C_s) \cap V(C_t) \neq \emptyset$. Because the order of $(C_s \cup C_t, D_s \cap D_t)$ is at most $|mid(st)|$, then $(C_s \cup C_t, D_s \cap D_t) \in B_p$ by the tangle basis axiom B2. If $C_s \cup C_t = G$, then stop, else there exists a separation (C_x, D_x) such that C_x is a connected component of G_s or G_t and $C_x \cap (C_s \cup C_t)$ is maximal (one exists because G is connected). Because C_x is either a connected component of G_s or a connected component of G_t (but not both) then the order of $(C_x \cup C_s \cup C_t, D_x \cap D_s \cap D_t)$ is at most $|mid(st)|$ and $(C_x \cup C_s \cup C_t, D_x \cap D_s \cap D_t) \in B_v$ by the tangle basis axiom B2. Continue this process until the separation $(G, \emptyset)$ is constructed. So, $(G, \emptyset)$ can be derived from the members of B_s and B_t using the tangle basis axiom B2 because G is connected. Thus, $(G, \emptyset)$ would be a member of B_v and also of $\mathcal{B}$, which contradicts tangle axiom T3. Therefore, $\theta'(G) \leq \beta(G)$. ∎

Tangle bases are similar in nature to respectful tangles associated with a surface Σ in the fact that only certain separations are needed to make the min–max relationship between branch decomposition and tangles valid. Respectful tangles were discussed in the work of Robertson and Seymour [15] and created the foundation for the Seymour and Thomas [16] result for planar graphs. Because we have shown the min–max relationship between connected tangle bases and branch decompositions, we can now discuss how the two can be used to create an algorithm to test if a given graph has branchwidth at most $k - 1$ where $k \geq 3$.

4. AN ALGORITHM FOR A TANGLE BASIS

Assume we are given a connected graph G with branchwidth at least 2 and a branch decomposition (T, ν) of G of width $k > 2$. We describe an algorithm that either gives a connected tangle basis or order k, or a branch decomposition of width $k - 1$ for G. This algorithm is somewhat described in the proof of Theorem 3. In addition, each separation (A, B) created will also have a surjection branch decomposition of G associated with it where the edges of B will be mapped

to one leaf node. The construction of these surjection branch decompositions are discussed in Section 4.1.

Initially, this branch decomposition-based algorithm includes making the tree of the branch decomposition into a rooted binary tree by subdividing an edge st of the branch decomposition and the newly created node p will be the root of the tree. The process also involves visiting the nodes of the tree in postdepth first-search order where each node of the tree will correspond to a set of distinct subgraphs of G, a *tangle set*. Because one member of a separation dictates the separation, each member of a tangle set will be considered the first member of its corresponding separation. A graph is in the tangle set if it is connected and the order of its corresponding separation is at most $k - 1$.

For the leaves of the rooted branch decomposition, if the leaf u corresponds to edge $ab \in E(G)$ then the graph of ab would be the only member of u's tangle set because $k > 2$. For a nonleaf node u in the rooted branch decomposition, u will have children l and r. Suppose that graph G_l is a graph in the tangle set of l and G_r is a graph in the tangle set of r. If $G_l \cup G_r$ is connected and the separation associated with $G_l \cup G_r$ has order at most $k - 1$, then $G_l \cup G_r$ becomes a member of the tangle set for u. Also, the members of the tangle sets of l and r become members of the tangle set for u. To find all possible members for the tangle set of u, we keep joining any two members of the tangle set for u until either no more new members are created or G is created. If G is a member of the tangle set for the root p, then we also have a branch decomposition of G of width at most $k - 1$. If G is not a member of the tangle set for p, then the tangle set corresponds to a connected tangle basis of G with order k. For the complexity of the algorithm, there are $O(e)$ nodes in the rooted branch decomposition tree where $e = |E(G)|$. Thus, the complexity of the algorithm is $O(n^{2k-2}e)$ because a tangle basis can have $O(n^{k-1})$ members where $n = |V(G)|$.

4.1. Joining Surjection Branch Decompositions

This section shows how to construct the associated surjection branch decompositions for leaf nodes and how to join surjection branch decompositions associated with G_l and G_r to form a surjection branch decomposition of G associated with $G_l \cup G_r$. Given a graph G_1 in the tangle set of some node of the input branch decomposition, the surjection branch decomposition (T_1, ν_1) of G associated with G_1 will consist of distinct leaves of T_1 for each edge of G_1 but only one leaf h_1 for all edges in $E(G) \setminus E(G_1)$. Also, the middle set of the edge of T_1 incident with h_1 will be $V(G_1) \cap V(G[E(G) \setminus E(G_1)])$. These surjection branch decompositions are used to conserve the composition structure of the separations.

The surjection branch decomposition (T', ν') associated with a leaf node of T that corresponds to some edge ab will be an edge vh', where $\nu'(ab) = u$ and $\nu'(E(G) \setminus \{ab\}) = h'$. For the joining of two surjection branch decompositions (T_l, ν_l) and (T_r, ν_r), there are two cases to consider: $G_l \cup G_r \neq G$ and $G_l \cup G_r = G$. Figure 3 illustrates the joining procedure

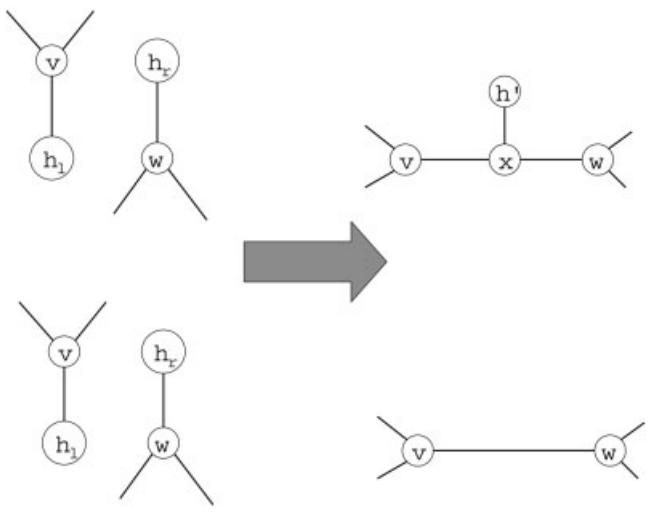

FIG. 3. Joining surjection branch decompositions.

for both cases where the top case is for $G_l \cup G_r \neq G$. For the first case, let vh_l be the edge of (T_l, ν_l) and let wh_r be the edge of (T_r, ν_r), where $\nu_l(h_l) = E(G) \setminus E(G_l)$ and $\nu_r(h_r) = E(G) \setminus E(G_r)$, respectively. Create the new surjection branch decomposition (T', ν') of G by deleting the nodes h_l and h_r, introducing new nodes x and h' such that x is incident with v, w, and h', and ν' such that

$$\nu'(e) = \begin{cases} \nu_l(e) & \text{if } e \in E(G_l), \\ \nu_r(e) & \text{if } e \in E(G_r), \\ h' & \text{else} \end{cases}$$

For the final case, let vh_l and wh_r be defined as earlier. Create the new branch decomposition of G by deleting the nodes h_l and h_r and adding the edge vw. Because, the separation associated with $G_l \cup G_r$ has order at most $k - 1$ then the associated surjection branch decomposition of G for $G_l \cup G_r$ will have width at most $k - 1$, and if $G_l \cup G_r = G$, then the new surjection branch decomposition of G will have width at most $k - 1$ and will be a branch decomposition of G.

5. COMPUTATIONAL RESULTS

The usage of the connectedness and only joining edge-disjoint members increased the runtime speedup and decreased memory requirements. Dominance was also used in the algorithm. Suppose F and H are in the tangle set of some tree node u and $V(F) \cap V(G[E(G) \setminus E(F)]) = V(H) \cap V(G[E(G) \setminus E(H)])$ (i.e., the middle set associated with F is equivalent to the middle set associated with H). If F is a strict subgraph of H then H dominates F and F can be deleted from the tangle set of u. This is true, because any graph K that is not a subgraph of H and can join with F, can also join with H. Thus, for every possible subset of nodes with at most $k - 1$ elements, there will be at most one member of the tangle set with the subset as its corresponding

TABLE 1. Optimal branch decomposition results.

Graphs	Nodes	Edges	Start width	β	Time (seconds)
bad.tele133	133	212	6	5	29,818
c3xc4	12	24	6	6	6.4
starfish	20	30	6	5	16.4
advbnd	196	242	3	3	2.5
celbnd	178	215	3	3	1.4
colbur	40	54	3	3	0.0
decomp	117	150	4	3	2.2
diagns	48	61	3	3	0.1
field	746	919	4	4	60,817
fmin	78	104	3	3	0.0
fmtgen	59	77	4	4	5.9
heat	69	95	3	3	0.0
init	348	449	3	3	5.0
injcon	80	105	3	3	0.0
integr	50	70	3	3	0.0
lasden	66	85	3	3	0.0
putb	138	175	3	3	0.6
radbg	639	795	3	3	6.7
radfg	654	812	3	3	8.7
rkfs	122	165	5	5	5,544
smooth	298	361	3	3	2.3
svd	320	412	4	4	386.1
trans	73	95	4	4	13.4
twldrv	264	362	5	4	57,956

middle set, which is why a tangle basis can have $O(n^{k-1})$ members where $n = |V(G)|$.

The computational results for computing optimal branch decompositions are given in Tables 1, 2, and 3. All computations were performed on a SGI IRIX6.4 and the software was written in the C++ language. The computational results include a recursive call to the algorithm discussed in Section 4. Once a new branch decomposition of G of lower width is found, the algorithm is called again with the new branch decomposition as the input branch decomposition. This step is repeated until a tangle basis of the input graph is found, which means the last input branch decomposition

TABLE 2. Optimal branch decomposition results.

Graphs	Nodes	Edges	Start width	β	Time (seconds)
cim.60.166	60	166	8	?	$>$ 100,000
comp.20.30	20	30	5	5	1.8
fac.foro.8.24	8	24	5	5	0.1
fac.foul.8.24	8	24	6	6	0.5
fac.leun.10.44	10	44	9	7	10.9
hims.34.45	34	45	4	4	0.4
hims.46.64	46	64	5	4	1.7
hims.48.69	48	69	6	5	10,196
jaya.10.22	10	22	5	5	0.2
kant.45.85	45	85	4	4	0.8
mart.cb450.30.6.56	30	56	4	4	1.1
mart.cb450.45.8.98	45	98	5	5	786.9
mart.cb450.47.8.99	47	99	6	5	9,223
mart.cb450.47.9.101	47	101	8	?	$>$ 100,000
mart.cl.17.4.39	17	39	5	5	2.3
sug.43.62	43	62	4	4	1.6

TABLE 3. Optimal branch decomposition results.

Graphs	Nodes	Edges	Start width	β	Time (seconds)
class12	12	45	7	7	565.5
class8	8	18	4	4	0.0
g1.cimi	10	21	5	4	0.0
g1	10	22	5	5	0.2
g10	25	71	4	4	0.4
g11	25	72	5	5	57.8
g2.cimi	60	166	8	?	$>$ 100,000
g2	45	85	4	4	0.8
g3.cimi	28	75	4	4	1.1
g4.cimi	10	22	5	5	0.2
g4	10	25	4	4	0.0
g5.cimi	45	85	4	4	0.8
g5	10	26	4	4	0.0
g6.cimi	43	63	6	5	173.3
g6	10	27	4	4	0.0
g7	10	34	6	6	5.3
g9	25	70	4	4	1.4
t1	11	22	4	4	0.0
t2	17	25	5	5	1.1

was optimal. Most of the graphs given in Table 1 come from instances of control-flow graphs from actual C compilations. These test instances were provided by Keith Cooper at Rice University, and usually have small branchwidth [18]. The graphs given in Tables 2 and 3 are test instances for the maximum planar subgraph problem. These graphs were provided by Petra Mutzel at Vienna University of Technology, Brett Peters of Texas A&M University, and Mauricio Resende at AT&T Labs. The maximum planar subgraph problem is important to network visualization and facility layout design. The input branch decompositions were given by an heuristic developed by Hicks [8], which is competitive with the heuristic developed by Cook and Seymour [5, 6].

For the tables, the column labeled "start width" offers the width of the initial branch decomposition. The column labeled "β" gives the branchwidth of the graphs, if they were obtained. The column labeled "time" only offers the runtime (in seconds) for computing the optimal branch decomposition from the initial branch decomposition. The run times do not include the time needed by the heuristic to construct an initial branch decomposition. The code was stopped if the run time exceeded 100,000 seconds. As one can tell from the tables, the algorithm relies heavily on the starting width of the input branch decomposition and the number of edges in the graphs. For instance, the run times for field and twldrv in Table 1 are similar, but the starting width and number of edges for the field were 4 and 919, respectively, while the starting width and number of edges for twldrv were 5 and 362. The code also failed to find optimal branch decompositions when the input width was 8 and the graph had at least 100 edges. In contrast, the code performed well for graphs with input width at most 7. Although not apparent, the run times also depended upon the number of edges of the branch decomposition with large middle sets.

6. CONCLUSIONS AND FUTURE WORK

In conclusion, we presented a more constructive object similar to a tangle called a tangle basis, and showed that tangles and tangle bases are coextensive for graphs with branchwidth of at least 2. Furthermore, we used the tangle bases to construct a branch decomposition-based algorithm to test if a connected graph G has branchwidth at most $k - 1$, where $k \geq 3$. This algorithm was used repeatedly to find an optimal branch decomposition of a graph given an input branch decomposition from a heuristic. We also showed through computational results that the algorithm is dependent upon the input width and the number of edges in the graph. Thus, the algorithm performed well for input graphs with a small number of edges and low initial width branch decompositions. Future directions in this area include finding more ways to decrease the number of members in a tangle set and the algorithm's possible use to find members of obstruction sets of graphs with a branchwidth of at most a constant k.

Acknowledgments

The author would like to thank Bill Christian and Bill Cook for our discussions about the research. The author would like to acknowledge the anonymous referees who assisted in the presentation of the research.

REFERENCES

[1] J. Alber and R. Niedermeier, Improved tree decomposition based algorithms for domination-like problems, Proc 5th Latin American Theoretical Informatics, Springer-Verlag, Heidelberg, Germany, 2002, pp. 613–627.

[2] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, J Algorithms 12 (1991), 308–340.

[3] H. Bodlaender and D. Thilikos, Constructive linear time algorithms for branchwidth, Proc 24th International Colloquium on Automata, Languages, and Programming, P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela (Editors), Springer-Verlag, Heidelberg, Germany, 1997, pp. 627–637.

[4] W.A. Christian, Linear-time algorithms for graphs with bounded branchwidth, PhD thesis, Rice University, 2003.

[5] W. Cook and P.D. Seymour, An algorithm for the ring-router problem, technical report, Bellcore, 1994.

[6] W. Cook and P.D. Seymour, Tour merging via branch-decomposition, INFORMS J Comput 15 (2003), 233–248.

[7] B. Courcelle, The monadic second-order-logic of graphs I: Recognizable sets of finite graphs, Informat Comput 85 (1990), 12–75.

[8] I.V. Hicks, Branchwidth heuristics, Congressus Numerantium 159 (2002), 31–50.

[9] I.V. Hicks, Branch decompositions and minor containment, Networks 43 (2004), 1–9.

[10] I.V. Hicks, Planar branch decompositions I: The ratcatcher, INFORMS J Comput (2005), to appear.

[11] I.V. Hicks, Planar branch decompositions II: The cycle method, INFORMS J Comput (2005), to appear.

[12] A. Koster, S. van Hoesel, and A. Kolen, Solving partial constraint satisfaction problems with tree-decomposition, Networks 40 (2002), 170–180.

[13] N. Robertson and P.D. Seymour, Graph minors I: Excluding a forest, J Comb Theory Series B 35 (1983), 39–61.

[14] N. Robertson and P.D. Seymour, Graph minors X: Obstructions to tree-decompositions, J Comb Theory Series B 52 (1991), 153–190.

[15] N. Robertson and P.D. Seymour, Graph minors XI: Circuits on a surface, J Comb Theory Series B 60 (1994), 72–106.

[16] P.D. Seymour and R. Thomas, Call routing and the ratcatcher, Combinatorica 14 (1994), 217–241.

[17] J.A. Telle and A. Proskurowski, Algorithms for vertex partitioning problems on partial k-trees, SIAM J Discrete Math 10 (1997), 529–550.

[18] M. Thorup, All structured programs have small tree width and good register allocation, Informat Comput 142 (1998), 159–181.