Degree-constrained decompositions of graphs: bounded treewidth and planarity[*]

Cristina Bazgan [†] Zsolt Tuza [‡] Daniel Vanderpooten [†]

Abstract

We study the problem of decomposing the vertex set V of a graph into two nonempty parts V_1, V_2 which induce subgraphs where each vertex $v \in V_1$ has degree at least $a(v)$ inside V_1 and each $v \in V_2$ has degree at least $b(v)$ inside V_2. We give a polynomial-time algorithm for graphs with bounded treewidth which decides if a graph admits a decomposition, and gives such a decomposition if it exists. This result and its variants are then applied to designing polynomial-time approximation schemes for planar graphs where a decomposition does not necessarily exist but the local degree conditions should be met for as many vertices as possible.

Keywords: graph decomposition, treewidth, planar graph, polynomial algorithm, PTAS.

1 Introduction

Given a graph $G = (V, E)$ and a subset $S \subseteq V(G)$, we denote by $d_S(v)$ the degree of a vertex $v \in V$ in $G[S]$, the subgraph of G induced by S. For $S = V$, the subscript is omitted, hence $d(v)$ stands for the degree of v in G.

Our starting point is the following general problem:

Decomposition

Input: A graph $G = (V, E)$, and two functions $a, b : V \to \mathbb{N}$ such that $a(v), b(v) \leq d(v)$, for all $v \in V$.

Question: Is there a nontrivial partition (V_1, V_2) of V such that $d_{V_1}(v) \geq a(v)$ for every $v \in V_1$ and $d_{V_2}(v) \geq b(v)$ for every $v \in V_2$?

A partition satisfying the above property is called a *decomposition* of G. If G admits a decomposition, we also say that G is *decomposable*. Moreover a vertex v is said to be *satisfied* when if $v \in V_1$ we have $d_{V_1}(v) \geq a(v)$ and if $v \in V_2$, we have $d_{V_2}(v) \geq b(v)$.

The decision problem Decomposition is *NP*-complete. Indeed, the rather special case where $a = b = \lceil \frac{d}{2} \rceil$, introduced in [14] as Satisfactory Partition, has been shown to be

[*] This research was supported by the bilateral research cooperation Balaton between EGIDE (France) and the Ministry of Education (Hungary) under grant numbers 07244RJ and F-29/2003. The second author was also supported in part by the Hungarian Scientific Research Fund, grant OTKA T-042710.

[†] LAMSADE, Université Paris-Dauphine, Place du Marechal de Lattre de Tassigny, 75775 Paris Cedex 16, France. Email: {bazgan,vdp}@lamsade.dauphine.fr

[‡] Computer and Automation Institute, Hungarian Academy of Sciences, H-1111 Budapest, Kende u. 13-17, Hungary; and Department of Computer Science, University of Veszprém, Hungary. Email: tuza@sztaki.hu

NP-complete in [5]. Moreover, it is *NP*-complete in the range $\lceil \frac{d}{2} \rceil < a = b \leq d-1$, as proved in [4].

Even if the problem is *NP*-complete, polynomial instances of this problem may arise when (i) restricting the structure of the graph, or (ii) imposing constraints on a and b, or (iii) both.

Concerning case (ii), Stiebitz [21] proved that, when a and b are such that $d(v) \geq a(v) + b(v) + 1$ for all $v \in V$, any graph admits a decomposition. His result is not constructive. A polynomial-time algorithm that finds such a decomposition is given in [6].

In case (iii), Kaneko [19] showed that any triangle-free graph such that $d(v) \geq a + b$ for all $v \in V$, where a and b are positive integer constants, admits a decomposition. Diwan [12] showed that any graph with girth at least 5 such that $d(v) \geq a + b - 1$ for all $v \in V$, where again a and b are positive integers ≥ 2 independent of v, admits a decomposition. These two results were presented for constants a and b instead of functions $a(v)$ and $b(v)$. Diwan's result was extended recently to the case of functions in [16]. However, the proofs of all these results are not constructive. In [6] we gave algorithms that find a decomposition in polynomial time for the general case of functions, provided that their sum (or sum minus 1) does not exceed the degree function.

In this paper we study Decomposition in case (i), i.e. without any restrictions on the functions a, b (apart from the trivial one that they should not exceed the degree function d), but imposing restrictions on the class of graphs. We are not aware of any previous result concerning this case. We show here that, for graphs with bounded treewidth, one can decide in polynomial time if a graph is decomposable, and give in polynomial time a decomposition when it exists.

It should be noted that the general result developed by Courcelle [11] — stating that any problem expressible in second-order monadic logic is polynomial-time solvable for graphs of bounded treewidth — cannot be applied directly here. The technical difficulty concerning the applicability of this result for Decomposition is that if $a(v)$ and $b(v)$ cannot be expressed for almost all $v \in V$ in a uniform way (e.g., $a(v) = a$ and $b(v) = b$ are constants, or $\frac{d(v)}{a(v)}$ and $\frac{d(v)}{b(v)}$ are independent of v), then the formula for the problem in second-order monadic logic is not of constant length. A similar technical complication yields the non-applicability of Gerber and Kobler's result [15] on graphs of bounded clique-width.

We also study some variants of Decomposition with additional constraints on the size of the vertex classes. Let $t = t(n)$ be an integer-valued function such that $0 \leq t(n) \leq n$ for every $n \in \mathbb{N}$. We consider the following problem:

t-Decomposition

Input: A graph $G = (V, E)$, and two functions $a, b : V \rightarrow \mathbb{N}$ such that $a(v), b(v) \leq d(v)$, for all $v \in V$.
Question: Is there a partition (V_1, V_2) of V with $|V_1| = t(|V|)$ such that $d_{V_1}(v) \geq a(v)$ for every $v \in V_1$ and $d_{V_2}(v) \geq b(v)$ for every $v \in V_2$?

A partition (V_1, V_2) such that $|V_1| = t(|V|)$ is called a *t-partition.* A t-partition where all vertices are satisfied is called a *t-decomposition*. If G admits a t-decomposition, we also say that G is *t-decomposable*.

A particular interesting case of this problem is when the graph has an even number of vertices and we consider only balanced partitions ($t = \frac{n}{2}$, where n is the number of vertices), giving rise to the following problem.

Balanced Decomposition
Input: A graph $G = (V, E)$ with an even number of vertices, and two functions $a, b : V \to \mathbb{N}$ such that $a(v), b(v) \le d(v)$, for all $v \in V$.
Question: Is there a partition (V_1, V_2) of V with $|V_1| = |V_2|$ such that $d_{V_1}(v) \ge a(v)$ for every $v \in V_1$ and $d_{V_2}(v) \ge b(v)$ for every $v \in V_2$?

Since an input graph may not have any (t-)decomposition, it is of interest to study the corresponding *optimization* problem where we try to satisfy as many of the vertices as possible. (In this setting it is not necessary to assume anymore that the vertex degree $d(v)$ is an upper bound on $a(v)$ and $b(v)$.) We consider then the two following problems.

Max Satisfying Decomposition
Input: A graph $G = (V, E)$ and two functions $a, b : V \to \mathbb{N}$.
Solution: A nontrivial partition (V_1, V_2) of V.
Value: The number of satisfied vertices v, i.e. those with $d_{V_1}(v) \ge a(v)$ if $v \in V_1$ and $d_{V_2}(v) \ge b(v)$ if $v \in V_2$.

Max Satisfying t-Decomposition
Input: A graph $G = (V, E)$ and two functions $a, b : V \to \mathbb{N}$.
Solution: A partition (V_1, V_2) of V such that $|V_1| = t(|V|)$.
Value: The number of satisfied vertices v, i.e. those with $d_{V_1}(v) \ge a(v)$ if $v \in V_1$ and $d_{V_2}(v) \ge b(v)$ if $v \in V_2$.

The particular case where cardinalities of the two vertex classes are imposed to be equal corresponds to a problem that we call Max Satisfying Balanced Decomposition.

Max Satisfying Balanced Decomposition was studied in [5] for the restricted functions $a(v) = b(v) = \lceil \frac{d(v)}{2} \rceil$. This problem is not only *NP*-hard but also has no polynomial-time approximation scheme, unless *P=NP*. The strongest positive result known so far on Max Satisfying Balanced Decomposition is a polynomial-time 3-approximation.

In Section 2 of this paper we prove that all the previous problems — and also their search versions — can be solved in polynomial time on graphs of bounded treewidth (Theorem 1). This result is then applied in Section 3 to design a polynomial-time approximation scheme for Max Satisfying t-Decomposition and Max Satisfying Decomposition on planar graphs (Theorem 3). Note that these problems are not known to be *NP*-hard on planar graphs. Our approach is to combine Baker's method [3] of dividing the input planar graph into families of k-outerplanar graphs and our method of finding maximum t-partitions of bounded-treewidth graphs.

In fact, the approximability of Max Satisfying t-Decomposition on planar graphs is a rather particular instance of a quite general principle that we formulate as Theorem 2 in Section 3. This result provides the frame for a technique to design approximation algorithms on a structure class when an efficient exact algorithm (or just a PTAS, or an algorithm of guaranteed approximation ratio) is already available on another class. We have chosen the formulation of Theorem 2 in a way to make it applicable not only for approximation schemes but also for less accurate approximations. On applying this method, the approximation scheme given in Theorem 3 is derived from Theorem 1. The main ingredients of this approach can already be found in Baker's paper [3], and have been applied to a number of problems e.g. in [10, 17, 18, 22]. One of the objectives in Section 3.1 is to point out that the technique can be split in a clear manner into two well-defined and completely independent parts; namely, one dealing with structural decomposability and the other concerning optimization problems that become approximable when certain structural conditions are imposed. In connection with the former, Eppstein [13] investigated how far one can extend the class of planar graphs in order that we still have a well-structured vertex partition into subgraphs of bounded treewidth. The combination of his results with our Theorems 1 and 2 implies a PTAS for our problems on a wider class of graphs, too, though it is formulated in Theorem 3 for planar graphs only.

2 Decomposition of graphs with bounded treewidth

Many graph problems, including a very large number of well-known *NP*-hard problems, have been shown to be solvable in polynomial time on graphs with treewidth bounded by a constant k [1, 7, 11]. In this section we prove that this is the case for deciding the existence of a (t-)decomposition and for maximizing the number of vertices that are simultaneously satisfied.

First, let us recall some necessary definitions.

Definition A *tree representation* $\mathcal{T} = (T, \mathcal{H})$ of a graph $G = (V, E)$ consists of a tree $T = (X, F)$ with node set X and edge set F, and a set system $\mathcal{H}$ over V whose members $H_x \in \mathcal{H}$ are labeled with the nodes $x \in X$, such that the following conditions are met:

- $\bigcup_{x \in X} H_x = V$.
- For each $uv \in E$ there is an $x \in X$ with $u, v \in H_x$.
- For each $v \in V$, the node set $\{x \in X \mid v \in H_x\}$ induces a subtree of T.

The third condition is equivalent to assuming that if $v \in H_{x'}$ and $v \in H_{x''}$ then $v \in H_x$ holds for all nodes x of the (unique) x'–x'' path in T. The *width* of a tree representation $\mathcal{T}$ is $w(\mathcal{T}) = \max_{x \in X} |H_x| - 1$ and the *treewidth* of G is defined as

$$tw(G) = \min_{\mathcal{T}} w(\mathcal{T})$$

where the minimum is taken over all tree representations $\mathcal{T} = (T, \mathcal{H})$ of G. The '-1' in the definition of $w(\mathcal{T})$ is included for the convenience that trees have treewidth 1 (rather than 2).

The determination of the treewidth of a graph is *NP*-hard [2]. However, for constant k, Bodlaender [8] gave a linear-time algorithm that determines whether the treewidth of G is at most k, and if so, finds a tree decomposition of G with treewidth at most k.

As indicated for example in [20], any tree representation $\mathcal{T} = (T, \mathcal{H})$ of a graph can be transformed in linear time into a so-called *nice tree representation* $\mathcal{T}' = (T', \mathcal{H}')$ with $w(\mathcal{T}') = w(\mathcal{T})$, with size $|T'| \le c|T|$ (for some absolute constant c) and with $H'_x \ne \emptyset$ for all $H'_x \in \mathcal{H}'$, where T' is a *rooted* tree satisfying the following conditions:

(a) Each node of T' has at most two children.

(b) For each node x with two children y, y', we have $H'_y = H'_{y'} = H'_x$.

(c) If a node x has just one child y, then

$$H'_x \subset H'_y \quad \text{or} \quad H'_y \subset H'_x \qquad \text{and} \qquad \big|\,|H'_x| - |H'_y|\,\big| = 1\,.$$

Concerning the appearance of substructures, one can see that the subtree T_x of T rooted at node x represents the subgraph G_x induced by precisely those vertices of G which occur in at least one H_y where y runs over the nodes of T_x.

Theorem 1 *Let $k > 1$ be any fixed integer. On the class of graphs with treewidth less than k, the problems* Decomposition *and t-*Decomposition*, for any function $t = t(n)$, can be decided in polynomial time. Moreover, decompositions for them — if they exist — and also an optimum solution for* Max Satisfying Decomposition *and* Max Satisfying t-Decomposition*, for any function $t = t(n)$, can be found in polynomial time.*

Proof: Let G be any input graph, say on n vertices $v_1, \ldots, v_n$. We consider a tree representation of width less than k, which can be obtained in linear time by the algorithm proposed in [8]. Let $\mathcal{T} = (T, \mathcal{H})$ be a nice tree representation, rooted in r, obtained from the previous one.

The essential part of the algorithm is dynamic programming, organized as a *postorder* traversal of (T, r). For each node x of T, a collection of 4-tuples

$$R_j(x) = (P_j, \mathbf{v}_j, f_j, s_j) \qquad j = 1, 2, \ldots$$

will be calculated, where

- $P_j = (A, B)$ is a bipartition of H_x,
- $\mathbf{v}_j$ is a vector of length n, all of its coordinates are integers between 0 and $n-1$, and its ith coordinate is positive only if $v_i \in H_x$,
- f_j and s_j are integers between 0 and n.

Since $w(T) < k$, at most $2^k \cdot n^k \cdot (n+1)^2 = O(2^k n^{k+2})$ records are maintained for each x. In addition,

- if x is not a leaf, then one or two pointers from each $R_j(x)$ to one record of each child y of x is registered, indicating which partition at the node y has been used when creating $R_j(x)$.

The components of $R_j(x)$ are interpreted as follows. Suppose that (A_x, B_x) is a vertex partition of G_x, such that $A \subseteq A_x$ and $B \subseteq B_x$. Then, for $v_i \in H_x$, the ith coordinate of vector $\mathbf{v}_j$ is equal to $d_{A_x}(v_i)$ if $v_i \in A_x$ and $d_{B_x}(v_i)$ if $v_i \in B_x$. Moreover, we define $f_j = |A_x|$ (cardinality of the *f*irst partition class) and s_j to be the number of vertices that are satisfied in G_x under (A_x, B_x), i.e. with $d_{A_x}(v_i) \geq a(v_i)$ or $d_{B_x}(v_i) \geq b(v_i)$, respectively. A 4-tuple — combination of those data — occurs as one such $R_j(x)$ if and only if there exists at least one partition (A_x, B_x) with exactly the values listed in $R_j(x)$. It is essential to observe that the number of the records $R_j(x)$ for any one node x remains just polynomial in n, despite the number of vertex bipartitions of G_x becomes exponential as G_x gets large.

If it does not cause ambiguity, we sometimes omit the subscript j or the argument (x) from $\mathbf{v}_j$, $R_j(x)$, etc. However, since $H_x = H_y$ may occur, in the formalism it may be necessary e.g. to write $f(x)$ or $f_j(x)$ for f, to indicate that the object in question belongs to a specific record (numbered j) *at the node* x. Analogously, the coordinate for $v_i \in H_x$ in $R_j(x)$ will be denoted by $\mathbf{v}_j(x : i)$.

In the trivial case where T consists of just one node, G can have at most k non-isolated vertices, therefore the existence of a (t-) decomposition can be decided by brute force in constant time, since k is fixed. Similarly, a t-partition with the maximum number of satisfied vertices can be found efficiently. Hence, we assume that T has at least one leaf.

Depending on the position of x in T, and on the type of H_x, the 4-tuples R_j are computed as follows.

Leaf. If $x \in X$ is a leaf of T, then $P = (A, B)$ runs over all partitions of H_x (also including the two trivial ones), i.e. $j = 1, 2, \ldots, 2^{|H_x|}$. The coordinates of $\mathbf{v}_j$ and the values f_j and s_j are computed directly.

Two children. Let $x \in X$, its two children y' and y''. Consider any partition $P(x) = (A, B)$ of H_x. In order to create the set of records $R_j(x)$ where (A, B) appears, all $R_{j'}(y')$ and $R_{j''}(y'')$ containing (A, B) are collected, and for each such pair of 4-tuples an $R_j(x)$ is generated and the two pointers from $R_j(x)$ are adjusted to $R_{j'}(y')$ and $R_{j''}(y'')$ (unless the 4-tuple has already been obtained from an earlier pair). The coordinates of $\mathbf{v}_j$ remain zero outside H_x, while for $v_i \in A$ we have $\mathbf{v}_j(x : i) = \mathbf{v}_{j'}(y' : i) + \mathbf{v}_{j''}(y'' : i) - d_A(v_i)$ (and analogously for $v_i \in B$). Similarly, $f_j(x) = f_{j'}(y') + f_{j''}(y'') - |A|$. To compute $s(x)$ from $s(y') + s(y')$, we need to subtract the number of those v_i which are satisfied in both $R_{j'}(y')$ and $R_{j''}(y'')$, and add the number of those not satisfied in either of $R_{j'}(y')$ and $R_{j''}(y'')$ but satisfied in $R_j(x)$. All this is easily done by comparing $a(v_i)$ or $b(v_i)$ with $\mathbf{v}(z : i)$ for $z \in \{x, y', y''\}$.

Larger child. Assume $H_x = H_y \setminus \{v_i\}$, where y is the child of x. For each $R_j(y)$ we set $A_j(x) = A_j(y) \setminus \{v_i\}$ and $B_j(x) = B_j(y) \setminus \{v_i\}$, and if A_j gets decreased, we write $f_j(x) = f_j(y) - 1$. The only change in $\mathbf{v}$ is to reset $\mathbf{v}_j(x : i) = 0$. In this step, s remains unchanged. Also here, each possible 4-tuple is kept only once; and when $R_j(x)$ is created, a pointer from it to $R_j(y)$ is introduced.

Smaller child. Assume $H_x = H_y \cup \{v_i\}$, where y is the child of x. From each $R(y)$ with a partition $P = (A, B)$ of H_y, two of the records $R(x)$ are generated: one with the partition $P' = (A \cup \{v_i\}, B)$ and the other one with $P'' = (A, B \cup \{v_i\})$. For the former, the value of f is increased by 1. Moreover, from each $\mathbf{v}(y)$, the coordinates of the corresponding $\mathbf{v}(x)$ are obtained by increasing the coordinates at the neighbors of v_i in A or in B by 1, and adjusting $\mathbf{v}(x : i)$ to $d_A(v_i)$ or $d_B(v_i)$. Similarly to the case of two children, $s(x)$ is computed from $s(y)$ by comparing $a(v_i)$ or $b(v_i)$ with $\mathbf{v}(x : i)$ and $\mathbf{v}(y : i)$. Then, the pointer specifies $R(y)$ for both $R(x)$ generated from $R(y)$.

Root. Investigating the records $R_j(r)$ at the root r of T, the following necessary and sufficient conditions hold.

- Decomposition has an affirmative answer if and only if there is a j such that $0 < f_j(r) < n$ and $s_j(r) = n$ are valid in $R_j(r)$.
- t-Decomposition has an affirmative answer if and only if $f_j(r) = t$ and $s_j(r) = n$ hold in some $R_j(r)$.
- Max Satisfying Decomposition has the following optimum:

$$\max \{s_j(r) \mid 0 < f_j(r) < n\}$$

- Max Satisfying t-Decomposition has the following optimum:

$$\max \{s_j(r) \mid f_j(r) = t\}$$

These requirements can be tested in an obvious way, once the records $R_j(r)$ at the root have been computed. Having found one affirmative or optimal case (for decision or maximization, respectively), from $R_j(r)$ one can trace back a sequence of records down to all the leaves of T along the pointers registered. This sequence determines a vertex partition of the entire G, in which the degree (and size) conditions hold, and the number of satisfied vertices is maximum.

Time analysis. Since only $O(2^k n^{k+2})$ records are maintained at each node, and every non-leaf node has one or two children, at most $O(4^k n^{2k+4})$ combinations are considered in each step. One can see that each of them requires only a polynomial number of elementary operations; and also the occurring redundancies (more than one copy of the same 4-tuple at the same node) can be eliminated efficiently. Altogether, the algorithm terminates in polynomial time. □

3 Approximation schemes and planar graphs

In connection with the Decomposition problem, the contribution of this section is a polynomial-time approximation scheme for Max Satisfying t-Decomposition and Max Satisfying Decomposition on planar graphs. Since the proof is based on a much more general approach, however, it seems reasonable to split the section into two parts, hence separating the method itself from its application.

3.1 Extending an approximation to another structure class

Theorem 2 *Let* $\mathbf{P}$ *be a maximization [minimization] problem,* $c \geq 1$ *any real number, and* $\mathcal{G}$, $\mathcal{H}$ *structure classes. Suppose that there exist reals* $c', c'' \geq 1$ *such that the following conditions are met:*

(a) $\mathbf{P}$ *has a polynomial* c'*-approximation on* $\mathcal{H}$.

(b) *For every instance* $G \in \mathcal{G}$ *of* $\mathbf{P}$ *it is possible to find in polynomial time an instance* $H \in \mathcal{H}$ *such that* $Opt_{\mathbf{P}}(H) \geq \frac{1}{c''} Opt_{\mathbf{P}}(G)$ *[respectively,* $Opt_{\mathbf{P}}(H) \leq c'' \, Opt_{\mathbf{P}}(G)$ *].*

(c) *Each solution of* H *can be extended in polynomial time to a solution on* G *without making its value worse.*

(d) $c'c'' \leq c$.

Then $\mathbf{P}$ *has a polynomial* c*-approximation on* $\mathcal{G}$.

Proof: Based on the conditions above, the following procedure can be designed:

1. Given G and c, find an instance $H \in \mathcal{H}$ that satisfies (b).

2. Obtain a c'-approximation for H.

3. Extend this solution to one on G, whose value is not worse than that on H.

This procedure is well-defined and runs in polynomial time, since Step 1 can be done efficiently by assumption, Step 2 is guaranteed by Condition (a), Step 3 only needs the assumptions given in (c).

Since the optimum on H is just c'' away from the optimum on G, and the solution found for H is within c' from its optimum, Condition (d) implies that a c-approximation on G is obtained. □

A way of implementing Condition (b) is to find in polynomial time a bounded number, $m = m_{c''}(n)$ of instances $H_i \in \mathcal{H}$ for all $i = 1, \dots, m$ (where n is the size of G) such that $\max_{1\le i\le m} Opt_{\mathbf{P}}(H_i) \ge \frac{1}{c''} Opt_{\mathbf{P}}(G)$ [respectively, $\min_{1\le i\le m} Opt_{\mathbf{P}}(H_i) \le c'' \, Opt_{\mathbf{P}}(G)$]. Let us note that in this case for every feasible c'', the function $m_{c''}(n)$ is a polynomial in n. Nevertheless, in the application of the second subsection, m will be a constant for every fixed c.

An interesting special case of Theorem 2 is when $c' = 1$, i.e. if $\mathbf{P}$ admits an exact solution on $\mathcal{H}$. It often happens for well-structured classes (e.g. interval systems) or classes with a restricted invariant, such as the class of graphs with bounded treewidth. This latter one is the approach of [18] for Max Bisection on planar graphs. We follow a similar approach for decomposition problems on planar graphs.

3.2 Max Satisfying (t-)Decomposition on planar graphs

Applying Theorem 1 and the method described in the previous subsection, here we prove the following result.

Theorem 3 Max Satisfying t-Decomposition, *for any function* $t = t(n)$, *and* Max Satisfying Decomposition *admit a polynomial-time approximation scheme on planar graphs.*

Proof: For every $\varepsilon > 0$, we describe a polynomial-time algorithm that gives a $(1+\varepsilon)$-approximation for Max Satisfying t-Decomposition in any planar input graph. We shall use the notation of Theorem 2.

Given ε, let $m \ge 2 + \frac{2}{\varepsilon}$ be any integer (independent of input size). We choose $c' = 1$ and $c'' = 1 + \varepsilon$, so that Condition (d) holds automatically.

We define $\mathcal{G} = \{\text{planar graphs}\}$ and $\mathcal{H} = \{\text{graphs of treewidth less than } 3m\}$. Theorem 1 then implies the validity of (a). Hence, what remains is to ensure Condition (b). For this purpose we will select subgraphs $H_1, \dots, H_m$, as indicated after the proof of Theorem 2, for one of which we can guarantee that the optimum solution on it will not be far from the optimum on G. Though the H_i will be different from those applied in [18], the method of finding them is fairly similar.

Let $G = (V, E)$ be the planar input graph. First we embed G in the plane, and set $G_0 = G$ and $V_0 = \emptyset$. Assuming that G_i and V_i are at hand, and G_i is nonempty, let V_{i+1} be the set of vertices on the boundary of the infinite region of G_i, and G_{i+1} the induced subgraph $G_i - V_{i+1}$. At the end of this procedure (which clearly can be implemented in polynomial time) we obtain a partition of V into the nonempty sets $V_1, \dots, V_k$, for some k.

Now, for $i = 1, 2, \dots, m$ let H_i be the graph obtained from G by deleting the edges between V_ℓ and $V_{\ell+1}$ for all ℓ such that $1 \le \ell < k$ and $\ell \equiv i \pmod{m}$. Moreover, in each H_i, if v was incident to a deleted edge of G, then we modify the values a, b to $a(v) = b(v) = d_G(v)$. On all the other vertices of H_i, the functions a, b remain unchanged. That is, every vertex gets modified values in at most two of the H_i.

The subgraphs H_i are so-called m-outerplanar graphs and have bounded treewidth $3m-1$ [9] i.e. $H_i \in \mathcal{H}$ holds and hence Theorem 1 applies, verifying Condition (a). It is also clear that any t-partition of H_i is one of G as well, and — since no vertex of modified a, b can be satisfied in H_i — the number of vertices satisfied in G under the same vertex partition cannot be smaller than that in any H_i.

Finally, to prove the validity of Condition (b), suppose that an optimum solution (V', V'') on G satisfies the vertices of the set S (i.e., $Opt(G) = |S|$). Consider the subsets S_i obtained from S by deleting those vertices whose a, b has been modified in H_i. Since every vertex gets modified values of functions a and b in at most two of the H_i, $|S_1| + \ldots + |S_m| \geq (m-2)\,|S|$, consequently there exists an i with $|S_i| \geq (1 - \frac{2}{m})\,|S| \geq \frac{1}{1+\varepsilon}|S|$. In the corresponding H_i, the partition (V', V'') satisfies all vertices of S_i, thus $Opt(H_i) \geq |S_i| \geq \frac{1}{1+\varepsilon} Opt(G)$ as required.

In order to obtain a polynomial-time approximation scheme for Max Satisfying Decomposition, we just need to iterate the previous polynomial-time approximation scheme for $t = 1, \ldots, \frac{n}{2}$ and retain the best solution found. □

References

[1] S. Arnborg, *Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey*, BIT, 25 (1985), 2–23.

[2] S. Arnborg, D. G. Corneil and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM Journal on Algebraic and Discrete Methods, 8 (1987), 277–284.

[3] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, Journal of the ACM, 41 (1994), 153–180.

[4] C. Bazgan, Zs. Tuza and D. Vanderpooten, *On the existence and determination of satisfactory partitions in a graph*, Proceedings of the 14th ISAAC 2003, LNCS 2906, 444–453.

[5] C. Bazgan, Zs. Tuza and D. Vanderpooten, *Complexity and approximation of satisfactory partition problems*, Proceedings of the 11th COCOON 2005, LNCS 3595, 829–838.

[6] C. Bazgan, Zs. Tuza and D. Vanderpooten, *Efficient algorithms for decomposing graphs under degree constraints*, submitted, 2005.

[7] H. L. Bodlaender, *Dynamic programming algorithms on graphs with bounded treewidth*, Proceedings of the 18th ICALP 1988, LNCS 317, 105–119.

[8] H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on Computing, 25 (1996), 1305–1317.

[9] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoretical Computer Science, 209 (1998), 1–45.

[10] Z.-Z. Chen, *Practical approximation schemes for maximum induced-subgraph problems on $K_{3,3}$-free or K_5-free graphs*, Proceedings of the 23rd ICALP 1996, LNCS 1099, 268–279.

[11] B. Courcelle, *The monadic second-order logic of graphs. III. Tree-decompositions, minors and complexity issues*, RAIRO Informatique Théorique Appliquée, 26 (1992), 257–286.

[12] A. Diwan, *Decomposing graphs with girth at least five under degree constraints*, Journal of Graph Theory, 33 (2000), 237–239.

[13] D. Eppstein, *Diameter and treewidth in minor-closed graph families*, Algorithmica, 27 (2000), 275–291.

[14] M. Gerber and D. Kobler, *Algorithmic approach to the satisfactory graph partitioning problem*, European Journal of Operation Research, 125 (2000), 283–291.

[15] M. Gerber and D. Kobler, *Algorithms for vertex-partitioning problems on graphs with fixed clique-width*, Theoretical Computer Science, 299 (2003), 719–734.

[16] M. Gerber and D. Kobler, *Classes of graphs that can be partitioned to satisfy all their vertices*, Australasian Journal of Combinatorics, 29 (2004), 201–214.

[17] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns, *A unified approach to approximation schemes for NP- and PSPACE-hard problems for geometric graphs*, Proceedings of the 2nd ESA 1994, LNCS 855, 424–435.

[18] K. Jansen, M. Karpinski, A. Lingas and E. Seidel, *Polynomial time approximation schemes for* Max-Bisection *on planar and geometric graphs*, Proceedings of the 18th STACS 2001, LNCS 2010, 365–375.

[19] A. Kaneko, *On decomposition of triangle-free graphs under degree constraints*, Journal of Graph Theory, 27 (1998), 7–9.

[20] T. Kloks, *Treewidth. Computations and Approximations*, LNCS 842, 1994.

[21] M. Stiebitz, *Decomposing graphs under degree constraints*, Journal of Graph Theory, 23 (1996), 321–324.

[22] D. M. Thilikos and H. L. Bodlaender, *Fast partitioning l-apex graphs with applications to approximating maximum induced-subgraph problems*, Information Processing Letters, 61 (1997), 227–232.