Chapter 6
Efficient Local Representations of Graphs

Edward Scheinerman

Abstract Informally, an *efficient local representation* of a graph G is a scheme in which we assign short labels (representable by a "small" number of bits, hence *efficient*) to G's vertices so that we can determine if two vertices are adjacent simply by examining the labels assigned to the pair of vertices (hence *local*). For some classes of graphs (such as planar graphs), one can devise local representations, but for others (such as bipartite graphs), this is not possible.

We present a conjecture due to Muller [22] and to Kannan, Naor, and Rudich [15] that distinguishes those hereditary classes of graphs (closed under induced subgraphs) for which an efficient local representation is feasible from those for which it is not.

Notation All graphs in this chapter are *simple*: their edges are undirected, and they have neither loops nor multiple edges. For a graph $G = (V, E)$, the number of vertices is nearly always denoted by the letter n. The notation $v \sim w$ indicates that vertices v and w are adjacent, i.e., $vw \in E$. For a positive integer n, we write $[n]$ for the set $\{1, 2, \ldots, n\}$. We write $\lg n$ for the base-2 logarithm of n. We also write $\log n$ but that is invariably wrapped in big-oh notation, so the base is irrelevant.

6.1 Seeking an Efficient Data Structure for Graphs

The *efficient local representation of graphs* problem is due to Muller [22] and to Kannan, Naor, and Rudich [15].

Here's the overarching question: *How do we efficiently represent a graph in a computer?*

Perhaps the simplest method is via an adjacency matrix; the memory to store this matrix uses $\Theta(n^2)$ bits, but this may be wasteful if the graph does not have

E. Scheinerman (✉)
Department of Applied Mathematics and Statistics, Johns Hopkins University,
Baltimore, MD 21218, USA
e-mail: ers@jhu.edu

R. Gera et al. (eds.), *Graph Theory*, Problem Books in Mathematics,
DOI 10.1007/978-3-319-31940-7_6

too many edges. In the latter case, adjacency lists may be a better option. See, for example, [8, 12], or [31] for a discussion of data structures one may use to represent a graph.

If we think of a graph as modeling relations between its vertices, then determining if two vertices are adjacent "should" only depend on some shared property of the two nodes under consideration.

To illustrate our thoughts, we begin by discussing *interval representations* of graphs.

6.1.1 Interval Representations of Graphs

Let G be a graph. We say that G has an *interval representation* if we can assign to each vertex $v \in V(G)$ a real interval J_v so that for distinct vertices v and w we have

$$v \sim w \iff J_v \cap J_w \neq \varnothing.$$

For example, let G be the path graph with $1 \sim 2 \sim 3 \sim 4$. The following assignment

$$1 \mapsto [1,3], \quad 2 \mapsto [2,5], \quad 3 \mapsto [4,7], \quad \text{and} \quad 4 \mapsto [6,8].$$

gives an interval representation for G. See [11] and [19]. Interval representations of graphs are a special case of intersection representations [7, 12, 21, 24, 25].

This representation provides a terrific way to store a graph in a computer. For each vertex, we only need to hold two numbers: the left and right end points of its interval. To check if two vertices are adjacent, we just do some quick checks on four numbers.

How much storage space does such a representation consume? We might be concerned that we may need a great deal of precision to specify the intervals' end points. However, it's not hard to show that if G has an interval representation, then we can find a representation in which the end points are distinct values[1] in $[2n]$.

This implies that for each vertex of the graph, we hold a scant $O(\log n)$ bits of information, and we can test adjacency simply by examining the information attached to just the two vertices of interest.

Informally, this is what we mean by an *efficient local representation* of a graph. To each vertex v we attach a "short" [hence *efficient*] label $\ell(v)$, and adjacency

[1]Here's why: Closed intervals are compact. Therefore, given a finite collection of intervals, there is a positive ε such that the sizes of the gaps between nonintersecting pairs of these intervals are all greater than ε. This means we can enlarge intervals by moving left end points to the left and right end points to the right by amounts less than $\varepsilon/2$ and not create any additional intersections. In this way, we may modify the representation so that all $2n$ end points of the intervals are distinct. To determine if two intervals intersect, one only needs to know the relative order of the four end points. Therefore, we may reassign the end points to be distinct values in $\{1, 2, \ldots, 2n\}$ so long as we preserve their order.

between vertices v and w can be tested via a calculation whose inputs are just $\ell(v)$ and $\ell(w)$ [hence *local* as we do not consider labels on any other vertices nor do we reference a global data structure such as an adjacency matrix].

Sadly, interval representations are not the ideal we seek because not all graphs admit such a representation. A moment's doodling shows that the four cycle C_4 has no such representation.

Graphs that have interval representations are called *interval graphs*, and for that family, an efficient local representation is available. What about other families of graphs?

6.1.2 Additional Examples

Trees

Let T be a tree with n vertices. Arbitrarily select a vertex of T to be called the *root* and give the edges directions so that on any path to the root, all edges point toward the root. Thus every vertex (other than the root) has a unique parent.

We now attach labels to the vertices of T. Each label $\ell(v)$ is an ordered pair of integers (a_v, b_v) drawn from the set $[n]$. The a-values are assigned arbitrarily; the only requirement is that they be distinct. The b-labels depend on the direction of the edge. If the edge from v to w is oriented $v \to w$, then $b_v = a_w$. That is, the second element of v's label is the first element of v's parent. Since the root r does not have a parent, its label is simply (a_r, a_r). See Figure 6.1.

We now observe that the vertex labels are $2 \lg n$ bits in length, and testing vertices for adjacency only requires examining the labels of the two vertices:

$$v \sim w \iff a_v = b_w \text{ or } a_w = b_v. \tag{6.1}$$

This efficient local representation method easily extends from trees to acyclic graphs (forests). Indeed, the scheme works for graphs that are not trees. For example, consider an n-cycle and label its vertices, in order, as follows:

$$(1,2) \quad (2,3) \quad (3,4) \quad \cdots \quad (n-1,n) \quad (n,1).$$

Test (6.1) applies here as well.

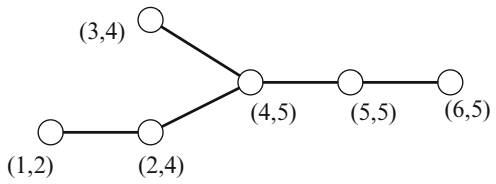

Fig. 6.1 An efficient local representation for a tree. Each label is an ordered pair (a, b). Observe that the a-values are distinct. The vertex labeled $(5, 5)$ is the root and the b-label of every other vertex is the a-label of its parent

Planar Graphs

We can extend this technique to planar graphs. The minimum degree of planar graph is at most 5. Let G be a planar graph with n vertices and assume that $V(G) = [n]$. Here is how we label $V(G)$.

Let v be a vertex of minimum degree. The label we assign to v is a tuple of the form $\ell(v) = (v; v_1, v_2, v_3, v_4, v_5)$ where the v_i's are v's neighbors (or 0s if v has fewer than 5 neighbors). Now delete v from G and repeat. Since $G - v$ is planar, we select a vertex, say w, of minimum degree in $G - v$ and assign to it the label $\ell(w) = (w; w_1, w_2, w_3, w_4, w_5)$ where the w_i's are (up to) five of its neighbors in $G - v$. Delete w and continue until the entire graph is consumed. This labeling uses $6 \lg n$ bits per vertex.

Once this labeling is created, we can test the adjacency of any two vertices x and y just by examining their labels $\ell(x) = (x; x_1, \ldots, x_5)$ and $\ell(y) = (y; y_1, \ldots, y_5)$: we simply check if $x = y_i$ or $y = x_i$ for some i.

k-Degenerate Graphs

The representation method we presented for planar graphs relies on an idea that we can generalize. For a positive integer k, a graph G is called *k-degenerate* if $\delta(H) \leq k$ for all subgraphs $H \subseteq G$. We write $\mathscr{D}_k$ for the set of k-degenerate graphs. Note that trees are 1-degenerate and planar graphs are 5-degenerate.

The labeling method we used for planar graphs readily extends to graphs in $\mathscr{D}_k$. Let G be an n-vertex graph in $\mathscr{D}_k$. Without loss of generality we assume $V(G) = [n]$. Choose a minimum degree vertex v of G, and let $\ell(v) = (v; v_1, v_2, \ldots, v_k)$ where the v_is are neighbors of v (or 0s if $d(v) < k$). Delete v from G and repeat. The labels assigned to vertices use $(k + 1) \lg n$ bits, and adjacency testing is the same as for planar graphs.

Complete Multipartite (Turan) Graphs

It is no surprise that interval graphs have efficient local representations; this is nearly immediate from their definition. The existence of efficient local representations for trees and planar graphs are instances of the same idea that shows graphs in $\mathscr{D}_k$ have efficient local representations. Here is another example to illustrate the central idea.

Recall that a graph G is a *complete multipartite* graph if we can partition its vertex set as $V(G) = I_1 \dot{\cup} \cdots \dot{\cup} I_k$ where each I_j is an independent set, and for $i \neq j$, every vertex in I_i is adjacent to every vertex in I_j.

Here's an efficient local representation. Given the partition of $V(G)$, let $\ell(v) = j$ if $v \in I_j$. We have $v \sim w$ exactly when $\ell(v) \neq \ell(w)$. Since the number of parts in the partition of G is at most n, the number of bits used in the labels is bounded by $\lg n$, and so this is an efficient local representation.

Circular Arc Graphs

The family of *circular arc graphs* [32] are a natural extension of interval graphs. In this case, we assign to each vertex v of a graph G an arc A_v of some fixed circle. Vertices v and w are adjacent exactly when their arcs A_v and A_w intersect.

As in the case of interval graphs, there is no loss of generality in assuming that the $2n$ end points of the arcs representing G are distinct and therefore may placed at the corners of a regular $2n$-gon. Therefore, the arcs can be described using just $2 \lg n$ bits, and hence we have an efficient local representation.

String Graphs

The family of *string graphs* [16, 17] generalize the circular arc graphs. In a circular arc graph, each vertex is represented by a special type of curve: an arc on a fixed circle. In a string graph, we use arbitrary planar curves. A graph G is a *string graph* provided we can assign to each vertex v a curve S_v in the plane so that vertices v and w are adjacent if and only if $S_v \cap S_w \neq \varnothing$. As shown by Sinden [29], not all graphs are string graphs.

Clearly a string representation of a graph is local, but is it efficient? Clearly we cannot represent an arbitrary curve with just a handful of bits. Is there a trick (such as the one we used for interval and circular arc graphs) that enables us to bound the number of bits needed for each vertex? We shall see that the answer is no (Proposition 2.5).

6.2 Efficient Local Representations

6.2.1 Main Definitions

The problem can be described as follows. Let $\mathscr{P}$ be a property of graphs. We want to know if $\mathscr{P}$-graphs have efficient local representations. This means n-vertex graphs in $\mathscr{P}$ can be labeled with "short" labels—$O(\log n)$ bits—and adjacency can be tested just by comparing the labels on two vertices. The adjacency test is the same for all graphs with property $\mathscr{P}$.

Let's be more precise. By a *graph property* we mean an isomorphism-closed set of graphs: $G \in \mathscr{P} \wedge H \cong G \Longrightarrow H \in \mathscr{P}$. It is sometimes more comfortable to refer to a property as a *class of graphs*.

A graph property $\mathscr{P}$ is *hereditary* if it is closed under taking induced subgraphs; that is, $G \in \mathscr{P} \wedge H \leq G \Longrightarrow H \in \mathscr{P}$ (where $H \leq G$ denotes that H is an induced subgraph of G). It is natural in this context to focus on hereditary properties because if G has an efficient local representation, then $G - v$ does as well.

Local representations are associated with graph classes (and not with individual graphs).

Definition 2.1. Let $\mathscr{P}$ be a hereditary property of graphs. A *local representation* for $\mathscr{P}$ is a symmetric function $A : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \{0, 1\}$ such that for every graph $G \in \mathscr{P}$ there is a labeling function $\ell : V(G) \to \mathbb{Z}^+$ such that for distinct vertices v, w of G, we have $v \sim w \iff A[\ell(v), \ell(w)] = 1$.

In other words, A is an adjacency test used by all graphs in $\mathscr{P}$. Each graph G in $\mathscr{P}$ has its own labeling function ℓ, and adjacency of two distinct vertices can be tested by applying the test A to the labels assigned to the pair.

[Note that Definition 2.1 requires that the vertex labels be positive integers, but in the examples in Section 6.1, the labels are tuples of integers. This is a minor technicality as we could perform kludges such as converting a pair of nonnegative integers (a, b) into a ternary number by writing a and b in binary and separating the values with a 2, like this: $(18, 11) \mapsto \texttt{1001021011}_{\text{three}}$.]

Definition 2.1 omits any notion of efficiency and, consequently, is not interesting. Indeed, *all* hereditary properties have local representations using this definition; here's how.

Let G be any graph and assume that $V(G) = [n]$. We label vertex v by $\ell(v) = [v; a(v)]$ where $a(v)$ is an n-tuple of 0s and 1s that indicates the neighbors of v. That is, the jth entry in $a(v)$ is 1 exactly when $v \sim j$. Stated differently, $a(v)$ is the v^{th} row of G's adjacency matrix. The test function A applied to labels $\ell(v) = [v; a(v)]$, and $\ell(w) = [w; a(w)]$ simply returns the w^{th} entry in $a(v)$.

In other words, if we allow long labels, the neighborhood of a vertex can be trivially encoded in the label. We therefore impose a bound on the size of the vertex labels. We have previously expressed this as a bound on the number of bits that specify the label as being $O(\log n)$. These bits can be merged to form a single positive integer, and the restriction on the number of bits translates to requiring labels to lie in a set of the form $[n^k]$ where k is a given positive integer.

Definition 2.2. Let $\mathscr{P}$ be a hereditary property of graphs. An *efficient local representation* for $\mathscr{P}$ is a symmetric function $A : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \{0, 1\}$ and a positive integer k such that every graph $G \in \mathscr{P}$ there is a labeling function $\ell : V(G) \to [n^k]$ (where $n = |V(G)|$) such that for distinct vertices v, w of G we have $v \sim w \iff A[\ell(v), \ell(w)] = 1$.

As before, the adjacency test A depends only on the property $\mathscr{P}$. Likewise the positive integer k is fixed for the class (does not depend on n). That labels take values in $[n^k]$ is tantamount to saying that the labels are (at most) $k \lg n$ bits in length. It is in this sense, the representation is (space) *efficient*.

6.2.2 The Problem

When we failed to restrict the number of bits in the labels, the class of all graphs $\mathscr{G}$ has a local representation. The interesting question is: What happens when we require efficiency? Is there an efficient local representation for all graphs?

This would be fantastic, but not surprisingly, the answer is no. To see why, we use a counting argument. To that end, we recall the definition of the *speed* of a hereditary property of graphs [1, 28].

Definition 2.3. Let $\mathscr{P}$ be a hereditary property of graphs and let n be a positive integer. Define $\mathscr{P}(n)$ to be the number of graphs in $\mathscr{P}$ with vertex set $[n]$. The function $\mathscr{P}(\cdot)$ is called the *speed* of the property.

For example, let $\mathscr{G}$ denote the class of all graphs. Then $\mathscr{G}(n) = 2^{\binom{n}{2}}$. Or let $\mathscr{P}$ be the property of having at most one edge. The set of graphs with vertex set $[n]$ that has at most one edge consists of the edgeless graph $\overline{K_n}$ and $\binom{n}{2}$ graphs with exactly one edge. Therefore $\mathscr{P}(n) = 1 + \binom{n}{2}$.

To show that not all hereditary properties admit an efficient local representation, we prove the following.

Proposition 2.4. *Let $\mathscr{B}$ be the class of bipartite graphs. Then $\mathscr{B}$ does not admit an efficient local representation.*

Proof. Suppose for contradiction that $\mathscr{B}$ admits an efficient local representation (A, k). Let n be a positive integer.

Every $G \in \mathscr{B}$ with $V(G) = [n]$ has a labeling function $\ell : [n] \to [n^k]$, and, necessarily, different graphs have different labeling functions.

The number of functions from $[n]$ to $[n^k]$ is n^{kn} which implies that $\mathscr{B}(n) \leq n^{kn}$. However, it is easy to see that $\mathscr{B}(n) \geq 2^{n^2/4}$ which exceeds n^{kn} once n is large enough.

In a similar spirit, we show that the class of string graphs does not admit an efficient local representation.

Proposition 2.5. *Let $\mathscr{S}$ be the class of string graphs. Then $\mathscr{S}$ does not admit an efficient local representation.*

Proof. Recall that a graph G is a *split graph* [10] provided we can partition $V(G) = K \cup I$ where K is a clique and I is an independent set. We show that the class of string graphs contains all split graphs. Since the number of split graphs on vertex set $[n]$ is at least as large as the number of bipartite graphs on $[n]$, the result follows exactly as in the proof of Proposition 2.4.

Let G be a split graph with $V(G) = K \cup I$. To see that G is a string graph, assign disjoint curves to the vertices in I. For the vertices in K, we choose curves that emanate from a common point. If $v \in K$ is adjacent to $i_1, i_2, \ldots, i_d \in I$, then v's curve can be chosen to intersect the disjoint curves that represent $i_1, i_2, \ldots, i_d$ while avoiding all other I-curves as in Figure 6.2. □

Note that the key fact we used in the proof of Proposition 2.4 is that *there are too many graphs on n vertices*. In order for a hereditary property $\mathscr{P}$ to admit an efficient local representation, the speed of $\mathscr{P}$ must be bounded by a function of the form n^{kn} for a specific integer k.

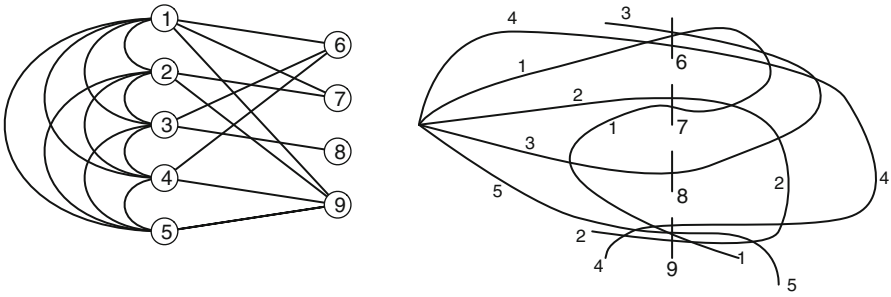

Fig. 6.2 We present a split graph and its string representation. The curves representing the vertices in the independent set $I = \{6, 7, 8, 9\}$ are the disjoint vertical line segments. The curves representing the vertices in the clique $K = \{1, 2, 3, 4, 5\}$ must intersect each other and exactly those curves for vertices in I to which they are adjacent. For example, the curve representing vertex 1 intersects vertical segments 6, 7, and 9, but not 8

Thus a hereditary property can fail to have an efficient local representation if it contains too many graphs—it violates the "speed limit" $\mathscr{P}(n) \leq n^{kn}$ for all fixed values of k.

What else could go wrong? That's a great question. Raised by [15] and [22], this is the question we offer our readers and present as our contribution to this compendium of favorite conjectures:

Conjecture 2.6 (Muller 1987; Kannan, Naor, Rudich 1992). *Let $\mathscr{P}$ be a hereditary property of graphs. Then $\mathscr{P}$ admits an efficient local representation if and only if there is an integer k such that $\mathscr{P}(n) \leq n^{kn}$.*

6.3 Challenging Examples

Conjecture 2.6 holds for a wide swath of hereditary graph properties including some we have considered (acyclic graphs, planar graphs, interval graphs, circular arc graphs) and many more we have not (such as permutation graphs [9] and threshold graphs [20]).

The following examples are—as best we know—possible counterexamples. The challenge is to find efficient local representations for these classes or show that none exist. See [2, 6], or [21] to find other hereditary classes as possible challenges.

Line Segment Intersection Graphs

The class of line segment intersection graphs lies between interval and string graphs. A graph G is a *line segment intersection graph* if we can assign to each $v \in V(G)$ a planar line segment L_v in such a way that $v \sim w$ if and only if $L_v \cap L_w \neq \varnothing$. Let $\mathscr{L}$ denote the class of line segment intersection graphs. See [3, 5, 18, 24].

Using the techniques described in [26], one can derive an upper bound for $\mathscr{L}(n)$ of the form n^{4n}, and so—if we believe Conjecture 2.6—this class should admit an efficient local representation.

Creating a local representation is easy; checking that it is efficient is difficult! Here's the representation. If $G \in \mathscr{L}$, then we know there is an assignment $v \mapsto L_v$ mapping vertices to line segments. These line segments can be represented as a 4-tuple of numbers (x, y, z, w) that specify the end points of the segment as (x, y) and (z, w). It's easy to check that there is no loss of generality in assuming that these coordinates are positive rational numbers (because $\mathbb{Q}^2$ is dense in $\mathbb{R}^2$). Therefore the labeling becomes an 8-tuple of integers. (Furthermore, by clearing denominators, we may assume all the coordinates are positive integers and a 4-tuple will suffice.)

Checking adjacency (i.e., intersection of the line segments) can be reduced to evaluating a pair of polynomial functions on the end points (details in [26]).

The only issue that remains is this: How many bits do we need to specify the end points? Do we need a high level of precision, or can this be accomplished with $O(\log n)$ bits?

Cographs

The class of *complement reducible graphs* [4], or *cographs* for short, can be described recursively as follows:

- K_1 is a cograph.
- If G is a cograph, so is its complement $\overline{G}$.
- If G and H are cographs, then so is their disjoint union. (This is the graph formed by simply taking copies of G and H on disjoint vertex sets and no additional edges.)

Let $\mathscr{C}$ denote the class of cographs. It is easy to see that $\mathscr{C}$ is a hereditary property. It is well known that cographs are exactly those graphs that do not contain the path P_4 as an induced subgraph.

With a bit of work, an upper bound of the form $\mathscr{C}(n) \leq n^{kn}$ can be derived. See sequence A000669 in [30].

Therefore, if Conjecture 2.6 holds, property $\mathscr{C}$ admits an efficient local representation.

Tolerance Graphs

The class of *tolerance graphs* [13, 14] may be considered as a generalization of interval graphs. We say that a graph G has a tolerance representation if we can assign to each vertex v of G a pair (I_v, t_v) where I_v is a closed, real interval and t_v is a positive real number so that $v \sim w$ in G if and only if the length of the intersection $I_v \cap I_w$ is at least $\min\{t_v, t_w\}$. Graphs with such a representation are called *tolerance graphs*.

An application of the methods in [26] gives a speed bound of the form n^{3n}, and therefore tolerance graphs satisfy the hypothesis of Conjecture 2.6. Do they satisfy the conclusion?

Geometric Graphs

Let $(\mathscr{X}, d)$ be a metric space. A graph G has a *geometric representation* in $(\mathscr{X}, d)$ if there is a mapping $f : V(G) \to \mathscr{X}$ such that $v \sim w$ if and only if $d[f(v), f(w)] \leq 1$. See [23].

A particularly natural example is $\mathscr{X} = \mathbb{R}^2$ together with the Euclidean metric. In this case, we can think of such graphs as having an intersection representation by unit discs.

Still with the Euclidean metric, for $\mathscr{X} = \mathbb{R}^k$, the number of geometric graphs with vertex set $[n]$ is bounded by an expression of the form n^{kn} and therefore satisfies the hypothesis of Conjecture 2.6. Do they satisfy the conclusion?

6.4 Variations

There are some natural variations on the core problem to consider.

6.4.1 Computation Concerns

We have focused on *space-efficient* representations. Each vertex holds a modest quantity of information. The combined information held by two vertices is sufficient to determine if they are adjacent. But at what cost? The authors of [15] also require that the function $A(\cdot, \cdot)$ be efficiently computable—in time polynomial in the size of the inputs. That is, the number of computational steps to check if v and w are adjacent in an n vertex graph should be bounded by an expression of the form $(\log n)^t$ for some fixed exponent t.

This is a perfectly reasonable requirement but appears to make this difficult problem only harder. Their conjecture (which we may dub the *strong efficient local representation conjecture*) is that hereditary properties that satisfy the speed limit $\mathscr{P}(n) \leq n^{kn}$ have an efficient local representation A that is polynomial-time computable. Clearly the strong version of the conjecture implies the weaker.

6.4.2 Other Label Sizes

Why do we want label sizes with $O(\log n)$ bits? This size is natural because already to name the vertices (e.g., specify vertex names in $[n]$) requires $\lg n$ bits. Thus the desire that $\ell(v)$ be represented in $O(\log n)$ bits is akin to saying that the labels are "about the same size" as the vertex names.

If, however, we are willing to modify this requirement, we can generate a host of additional problems.

Suppose we permit vertex labels to be larger—say, $O(\sqrt{n})$ bits. Then, presumably, such local representations could encompass more hereditary graph properties. If this is permitted, can we represent all hereditary properties that satisfy the speed limit $\mathscr{P}(n) \leq n^{kn}$? For the case of $O(\sqrt{n})$ bits, the counting argument shows that the properties must obey a bound of the form $n^{k\sqrt{n}}$; if $\mathscr{P}$ does obey such a bound, must it admit an $O(\sqrt{n})$ bits-per-vertex local representation?

On the other hand, we might consider using smaller labels, i.e., with $o(\log n)$ bits per vertex. As an extreme example, suppose $\mathscr{P}$ is the property of being a complete bipartite graph. Then we can label vertices with just a single bit (0 for vertices in one part of the bipartition and 1 for vertices in the other) and use the function $A(x, y) = \mathbf{1}[x \neq y]$.

Using the results in [28], we have the following result from [27].

Theorem 4.1. *Let $\mathscr{P}$ be a hereditary property of graphs and let k be a positive number with $k < \frac{1}{2}$. If, for all n sufficiently large, we have $\mathscr{P}(n) \leq n^{kn}$, then $\mathscr{P}$ admits a local representation in which the labels have $O(1)$ bits.* □

In other words, the "super efficient" [$o(\log n)$ bits per vertex] local representation conjecture is true. Furthermore, if a property has a representation using $o(\log n)$ bits per vertex, then it has a representation using a constant number of bits per vertex.

Acknowledgements Many thanks to Ralucca Gera and Craig Larson for the invitation to present this problem at the 2012 SIAM Discrete Mathematics Conference in Halifax, Nova Scotia, and to contribute to this volume. Thanks also to my students Elizabeth Reiland and Yiguang Zhang, as well as to a highly dedicated referee, for many helpful comments on drafts of this chapter.

References

1. Balogha, J., Bollobás, B., Weinreich, D.: The speed of hereditary properties of graphs. J. Comb. Theory Ser. B **79**, 131–156 (2000)
2. Brandstädt, A., Le, V.B., Spinrad, J.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia (1999)
3. Chalopin, J., Gonçalves, D.: Every planar graph is the intersection graph of segments in the plane. In: STOC '09 Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 631–638 (2009)
4. Corneil, D.G., Lerchs, H., Burlingham, L.S.: Complement reducible graphs. Discret. Appl. Math. **3**(3), 163–174 (1981)
5. de Fraysseix, H., de Mendez, P.O., Pach, J.: Representation of planar graphs by segments. Int. Geogr. **63**, 109–117 (1991)
6. de Ridder, H.N., et al.: Information system on graph classes and their inclusions. http://www.graphclasses.org/
7. Erdős, P., Goodman, A.W., Pósa, L.: The representation of a graph by set intersections. Can. J. Math. **18**(1), 106–112 (1966)
8. Even, S.: Graph Algorithms, 2nd edn. Cambridge University Press, Cambridge (2011)
9. Even, S., Pneuli, A., Lempel, A.: Permutation graphs and transitive graphs. J. ACM **19**(3), 400–410 (1972)
10. Földes, S., Hammer, P.L.: Split graphs. Congr. Numer. **19**, 311–315 (1977)

11. Gilmore, P.C., Hoffman, A.J.: A characterization of comparability graphs and of interval graphs. Can. J. Math. **16**, 539–548 (1964)
12. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, 2nd edn. North Holland, Amsterdam (2004)
13. Golumbic, M.C., Trenk, A.N.: Tolerance Graphs. Cambridge University Press, Cambridge (2004)
14. Golumbic, M.C., Monma, C.L., Trotter, W.T.: Tolerance graphs. Discret. Appl. Math. **9**(2), 157–170 (1984)
15. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. SIAM J. Discret. Math. **5**, 596–603 (1992)
16. Kratochvíl, J.: String graphs I. The number of critical nonstring graphs is infinite. J. Comb. Theory (B) **52**, 53–66 (1991)
17. Kratochvíl, J.: String graphs II. Recognizing string graphs is NP-hard. J. Comb. Theory (B) **52**, 67–78 (1991)
18. Kratochvíl, J., Nesetril, J.: Independent set and clique problems in intersection defined graphs. Comment. Math. Univ. Carol. **31**(1), 85–93 (1990)
19. Lekkerkerker, C.G., Boland, J.C.: Representation of a finite graph by a set of intervals on the real line. Fundam. Math. **51**, 45–64 (1962)
20. Mahadev, N.V.R., Peled, U.N.: Threshold Graphs and Related Topics. North-Holland, Amsterdam (1995)
21. McKee, T., McMorris, F.R.: Topics in Intersection Graph Theory. Society for Industrial and Applied Mathematics, Philadelphia (1999)
22. Muller, J.H.: Local structure in graph classes. Ph.D. thesis, Georgia Institute of Technology (1987)
23. Penrose, M.: Random Geometric Graphs. Oxford University Press, Oxford (2003)
24. Scheinerman, E.: Intersection classes and multiple intersection parameters of graphs. Ph.D. thesis, Princeton University (1984)
25. Scheinerman, E.: Characterizing intersection classes of graphs. Discret. Math. **55**(2), 185–193 (1985)
26. Scheinerman, E.: Geometry. In: Beineke, L., Thomas, R. (eds.) Graph Connections, pp. 141–154. Clarendon Press, Oxford (1997)
27. Scheinerman, E.: Local representations using very short labels. Discret. Math. **203**, 287–290 (1999)
28. Scheinerman, E., Zito, J.: On the size of hereditary properties of graphs. J. Comb. Theory (B) **61**, 16–39 (1994)
29. Sinden, F.: Topology of thin-film RC-circuits. Bell. Syst. Technol. J. **45**(9), 1639–1662 (1966)
30. Sloane, N.J.: The on-line encyclopedia of integer sequences. http://oeis.org/
31. Spinrad, J.: Efficient Graph Representations. The Fields Institute for Research in Mathematical Sciences. The American Mathematical Society, Providence (2003)
32. Tucker, A.C.: Matrix characterizations of circular-arc graphs. Pac. J. Math. **39**, 535–545 (1971)