The Total Interval Number of a Graph, III: Tree-like Graphs

Thomas M. Kratzke* and Douglas B. West†‡

completed October, 2003

Abstract

A multi-interval representation of a simple graph $G$ assigns each vertex a union of disjoint real intervals so that vertices are adjacent if and only if their assigned sets intersect. The total interval number $I(G)$ is the minimum of the total number of intervals used in such a representation of $G$. We present a linear-time algorithm to compute $I(G)$ when every block of $G$ is a complete graph or a cycle. Also, for an $n$-vertex cactus (every block is an edge or a cycle), the maximum of $I(G)$ is $\lfloor (18n - 12)/13 \rfloor$. For an $n$-vertex block graph (every block is a complete graph), the maximum is $\lfloor 3n/2 - 2 \rfloor$.

1 Introduction

An intersection representation $f$ of a simple graph $G$ assigns each vertex $v$ a set $f(v)$ such that two vertices $u$ and $v$ are adjacent if and only if $f(u) \cap f(v) \neq \emptyset$. A multi-interval representation is an intersection representation in which each $f(v)$ is a union of (closed) intervals on the real line. A $k$-interval is a union of $k$ disjoint real intervals. When $f(v)$ is a $k$-interval, we write $|f(v)| = k$ and say that $v$ is assigned $k$ intervals. Note that $|f(v)|$ is an abuse of notation, since $f(v)$ is an infinite subset of $\mathbb{R}$.

Graphs having multi-interval representations in which each $f(v)$ is a single interval are interval graphs; this class has many applications. To measure how far a graph is from being an interval graph, we may choose $f$ to minimize the maximum or the average of $|f(v)|$ over all vertices. The interval number $i(G)$ of a graph $G$ is the minimum $t$ such that $G$ has a multi-interval representation in which $\max_{v \in V(G)} |f(v)| = t$. The total interval number $I(G)$

---

*Metron, Inc., Reston, VA 22090, kratzke@metsci.com
†Department of Mathematics, University of Illinois, Urbana, IL 61801, west@math.uiuc.edu
‡This material is based upon work supported by the NSA under Award No. MDA904-03-1-0037, which requires the disclaimer that any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the NSA.
is the minimum $t$ such that $G$ has a multi-interval representation in which $\sum_{v \in V(G)} |f(v)| = t$ (dividing by the name of vertices yields the average of $|f(v)|$). We write $|f|$ for $\sum_v |f(v)|$ and call $|f|$ the size of $f$.

Interval number was introduced by Trotter and Harary [8] and has been studied in many papers. Although defined in [3],[8], total interval number was not studied until Aigner and Andreae [1] determined the maximum of $I(G)$ over several classes of $n$-vertex graphs, including trees ($\lceil (5n-3)/4 \rceil$), 2-connected outerplanar graphs ($\lceil 3n/2 - 1 \rceil$), triangle-free planar graphs ($2n-3$), and triangle-free graphs ($\lceil (n^2 + 1)/4 \rceil$). For the latter three classes, they conjectured that the upper bounds still hold without the restrictions to 2-connected or triangle-free graphs, respectively.

In [6], we proved these conjectures for outerplanar and general graphs on $n$ vertices, and we also proved the conjecture from [1] that $\max I(G) = \lceil (5m+2)/4 \rceil$ for connected graphs with $m$ edges. The proof that $I(G) \leq 2n-3$ for $n$-vertex planar graphs is quite lengthy and will appear in the next paper in this series. The final paper will study the maximum of $I(G)$ for $m$-edge graphs with lower bounds on minimum vertex degree, connectivity, or edge-connectivity.

In [7], we provided a linear-time algorithm to compute $I(G)$ when $G$ is a tree and observed that recognizing graphs with $I(G) = m(G) + 1$ is NP-complete, where $m(G)$ is the number of edges of $G$. In this paper, we refine the algorithm for trees to obtain a simpler algorithm that applies more generally. This algorithm applies to graphs in which every block is a complete graph or a cycle, where a block of a graph is a maximal connected subgraph with no cut-vertex. We have not been able to extend the algorithm to general outerplanar graphs. Testing $I(G) = m(G) + 1$ is NP-complete even for triangle-free 3-regular planar graphs [7], but the complexity of computing $I(G)$ for outerplanar graphs is not known.

A cactus is a connected graph in which every block is an edge or a cycle. A block graph is a graph in which every block is a complete graph (these are precisely the graphs that can be obtained as the intersection graph of the family of blocks in some graph [4]). We apply our algorithm to determine the maximum value of $I(G)$ for $n$-vertex cacti and for $n$-vertex block graphs; these are $\lceil (18n-12)/13 \rceil$ and $\lceil 3n/2 - 2 \rceil$, respectively. These two results were originally obtained in the first author’s dissertation [5] without the algorithmic approach.
2 Definitions for Representations

We allow $f(v) = \emptyset$, so isolated vertices contribute nothing to $|f|$. We henceforth use *representation* to mean multi-interval representation and *optimal representation* to mean minimum-sized representation (of $G$).

For a vertex $v$ in a graph with representation $f$, each boundary point of $f(v)$ is an *endpoint* (for $v$). The *endpoints* of a representation $f$ are all points that are endpoints for vertices. We assume that each endpoint of $f$ is an endpoint for exactly one vertex; every representation can be modified slightly to obtain this property without changing its size.

A point $x \in \mathbb{R}$ is *displayed* if $|f^{-1}(x)| = 1$. An interval is *displayed* if it contains a displayed point. A vertex $u$ is *displayed* in a representation $f$ if $f(u)$ contains a displayed point. If $f(u)$ contains a displayed endpoint, then $u$ is *end-displayed* in $f$. We further define the $u$-extent $\epsilon_u(f)$ of the representation $f$ according to how many endpoints of $f(u)$ are displayed:

<table>
<thead>
<tr>
<th>property of $u$</th>
<th>definition</th>
<th>$\epsilon_u(f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-displayed</td>
<td>$f(u)$ has no displayed point</td>
<td>$-1$</td>
</tr>
<tr>
<td>weakly-displayed</td>
<td>$u$ is displayed but not end-displayed in $f$</td>
<td>$0$</td>
</tr>
<tr>
<td>singly-displayed</td>
<td>$f(u)$ has exactly one displayed endpoint</td>
<td>$1$</td>
</tr>
<tr>
<td>doubly-displayed</td>
<td>$f(u)$ has at least two displayed endpoints</td>
<td>$2$</td>
</tr>
</tbody>
</table>

Note that $\epsilon_u(f) \geq 0$ when $u$ is displayed in $f$ and $\epsilon_u(f) \geq 1$ when $u$ is end-displayed. In a representation $f$, an endpoint of $f(u)$ that is displayed is a *$u$-point*.

The *code* of a vertex $u$ in $G$, denoted $c(G, u)$, is the maximum of $\epsilon_u(f)$ such that $f$ is an optimal representation of $G$. An optimal representation $f$ of $G$ is $u$-*optimal* if $\epsilon_u(f) = c(G, u)$.

A *component* of a representation $f$ is a maximal set of intervals in $f$ whose union is a single interval in $\mathbb{R}$, extending from a displayed left endpoint to a displayed right endpoint. A displayed endpoint is an endpoint of the component containing it.

The intersection graph of a family of multi-intervals is determined by the order of the endpoints of its intervals. In terms of the ordering of endpoints, *reflecting a component* means reversing the order of its endpoints, and *translating a component* means moving its sequence of endpoints to a different location in the sequence of components. In this combinatorial description, the geometric operation of shrinking a component has no effect. A *small interval* in $f$ is an interval whose endpoints are consecutive in the ordering of all endpoints.

We develop a recursive algorithm that combines optimal representations of subgraphs to obtain an optimal representation of a larger graph. The code of a specified vertex governs
how we combine the representations. The use of the code is analogous to loading the induction hypothesis in an inductive proof. In Section 3 we describe a way to combine $u$-optimal representations of graphs to obtain a representation of their union. The resulting representation is $u$-optimal for the full graph when $u$ is the only vertex belonging to move than one of the graphs in the decomposition. This is the situation when $u$ is a cut-vertex.

Sections 4 and 5 inductively solve the problem of constructing a $u$-optimal representation when $u$ is not a cut-vertex but $G$ is a graph in which every block is a complete graph or a cycle. Together, these results yield an inductive algorithm for finding optimal representations of such graphs. In Section 6, we apply the algorithms to solve the extremal problems for $I(G)$ on cacti and block graphs with $n$ vertices.

3 Cut-vertices

In this section we introduce a method for combining representations of several graphs. We obtain a $u$-optimal representation of the union when $u$ is the only vertex belonging to more than one of the graphs (Theorem 6). We begin by studying the properties of optimal representations.

Recall that $u$-points in a representation $f$ are displayed endpoints of $f(u)$. Our first lemma shows that an optimal representation has at most two $u$-points; otherwise, two components containing $u$-points can be combined to reduce the number of intervals. We also need a statement about when equality holds.

**Lemma 1** An optimal representation $f$ has at most two $u$-points, with equality only if the two $u$-points are the endpoints of a single component and no other component has a displayed interval for $u$.

**Proof.** If $f$ has $u$-points in distinct components, then reflections and translations of components can be used to make these endpoints consecutive. The two intervals can then be extended to meet, thus reducing $|f|$ without changing the graph represented. Thus all $u$-points lie in one component of an optimal $f$, and there are at most two of them.

If $f$ has two $u$-points in a component $A$ and has another component $B$ in which $u$ is displayed, then $A$ can be inserted into a displayed interval for $u$ in $B$ (see figure below). Geometrically, the interval in $B$ is cut, the component $A$ is inserted, and then two pairs of consecutive endpoints for $u$ are deleted (by extending intervals to meet). This operation
reduces $|f|$ by 1 without changing the graph represented.

Lemma 1 implies that if $f$ is optimal and $u$ is displayed in $f$, then $\epsilon_u(f)$ is the number of $u$-points in $f$. The proof describes two operations with respect to $u$. The operations are **splice** (merge end-displayed intervals) and **swallow** (insert a component into a displayed interval). When $u$ is the vertex for which the number of intervals decreases, we call these operations $u$-operations. Each $u$-operation reduces $|f|$, $|f(u)|$, and the number of components by 1 and reduces the number of $u$-points by two.

We use $u$-operations in combining $u$-optimal representations of graphs to obtain a representation of their union. Given graphs $G_1, \ldots, G_k$, for each $i$ let $f_i$ be a representation of $G_i$. A $u$-reduction of $\{f_i\}_{i=1}^k$ is a representation $f$ of $\bigcup G_i$ formed by concatenating $f_1, \ldots, f_k$ (that is, concatenating their endpoint sequences) and then performing $u$-operations until no more $u$-operations are available. Note that $u$ is a single fixed vertex in this process. A representation admitting no $u$-operation is $u$-reduced. By Lemma 1, optimal representations are $u$-reduced for all $u \in V(G)$.

A $u$-saturated representation is a representation in which displayed points of $f(u)$ occur only within components of $f$ whose endpoints both are $u$-points. Since Lemma 1 implies that $u$-reduced representations have at most two $u$-points, we conclude the following.

**Remark 2** A representation obtained from $f$ by a $u$-operation is $u$-saturated if and only if $f$ is $u$-saturated. A $u$-reduced representation $f$ is $u$-saturated if and only if $\epsilon_u(f) \in \{-1, 2\}$. 

Let $G = \bigcup_{i=1}^k G_i$, and for each $i$ let $f_i$ be a representation of $G_i$. The size and $u$-extent of a representation of $G$ obtained by $u$-reducing $\{f_i\}_{i=1}^k$ does not depend on the order of $u$-operations performed. We prove this by computing these parameters for the resulting $f$ in terms of their values for $f_1, \ldots, f_k$. The lemma makes no assumptions about the $u$-optimality of $f_1, \ldots, f_k$. 

5
Lemma 3  Every \( u \)-reduction \( f \) of representations \( f_1, \ldots, f_k \) has the same size and \( u \)-extent. If \( \beta \) is the total number of \( u \)-points over all \( f_i \), and \( Q \) is the statement “\( \epsilon_{f_i}(u) \in \{-1, 2\} \)” for all \( i \), then \( \epsilon_u(f) \) has the value listed below in terms of \( \beta \) and \( Q \). Furthermore, if \( \epsilon_u(f) = 2 \), then \( |f| = \sum |f_i| - \lfloor |\beta|/2 \rfloor + 1 \); otherwise, \( |f| = \sum |f_i| - \lfloor |\beta|/2 \rfloor \).

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>( Q )</th>
<th>( \epsilon_u(f) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>-1</td>
</tr>
<tr>
<td>( 2\alpha )</td>
<td>false</td>
<td>0</td>
</tr>
<tr>
<td>( 2\alpha + 1 )</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>( 2\alpha &gt; 0 )</td>
<td>true</td>
<td>2</td>
</tr>
</tbody>
</table>

Proof. We first consider the case where \( Q \) is true. In this case \( \beta \) is even, since each \( f_i \) contributes 0 or 2 \( u \)-points. Statement \( Q \) implies that the concatenation of \( \{f_i\} \) is \( u \)-saturated, which implies that \( f \) is \( u \)-saturated, by Remark 2. Thus \( \epsilon_u(f) \in \{-1, 2\} \). Since every \( u \)-operation leaves \( u \) displayed, \( \epsilon_u(f) = -1 \) if and only if \( \epsilon_{f_i}(u) = -1 \) for every \( i \), which requires \( \beta = 0 \) and \( |f| = \sum |f_i| \). When \( \epsilon_u(f) = 2 \), the \( u \)-reduction has two \( u \)-points and thus is obtained by \( \beta/2 - 1 \) \( u \)-operations.

If \( Q \) is false and there are at least two \( u \)-points, then throughout \( u \)-reduction there remain \( u \)-points in distinct components or a displayed \( u \)-interval available to receive a splice. Thus the representation remains \( u \)-unsaturated, \( \lfloor |\beta|/2 \rfloor \) \( u \)-operations are applied in the reduction, and \( \epsilon_u(f) \) is 0 if \( \beta \) is even and 1 if \( \beta \) is odd.

Corollary 4  Given graphs \( G_1, \ldots, G_k \), for each \( i \) let \( f_i \) and \( f_i' \) be \( u \)-optimal representations of \( G_i \). If \( f \) is a \( u \)-reduction of \( \{f_1, \ldots, f_k\} \) and and \( f' \) is a \( u \)-reduction of \( \{f_1', \ldots, f_k'\} \), then \( f \) and \( f' \) have the same size and \( u \)-extent.

Proof. For each \( i \), \( u \)-optimality implies that \( f_i \) and \( f_i' \) are \( u \)-reduced and have the same size and \( u \)-extent. Hence a concatenation of \( \{f_1, \ldots, f_k\} \) and a concatenation of \( \{f_1', \ldots, f_k'\} \) have the same size, number of \( u \)-points, and truth value of \( Q \). Lemma 3 then yields \( |f| = |f'| \) and \( \epsilon_u(f) = \epsilon_{f'}(u) \).

The inductive algorithms that we will present combine representations of edge-disjoint subgraphs. Under suitable conditions, \( u \)-optimality of all of \( f_1, \ldots, f_k \) implies \( u \)-optimality of each \( u \)-reduction of \( \{f_1, \ldots, f_k\} \). For parallel implementations, one may desire partitions into many subgraphs, but the essence of the sequential algorithm is captured by partitions into two subgraphs, and we reduce the proof of the theorem to this case. We begin by applying Lemma 3 to the \( u \)-reduction of two \( u \)-reduced representations.
Corollary 5 If \( f \) is a \( u \)-reduction of \( u \)-reduced representations \( f_1 \) and \( f_2 \) with \( b_i = \epsilon_{f_i}(u) \), then

\[
\begin{align*}
    b_1 + b_2 &\leq 1 \quad \Rightarrow \quad |f| = |f_1| + |f_2| \\
    b_1 + b_2 &\geq 2 \quad \Rightarrow \quad |f| = |f_1| + |f_2| - 1
\end{align*}
\]

\( \epsilon_u(f) = \max \{ b_1, b_2 \} \)

Proof. Always, \(-1 \leq b_i \leq 2\). Because \( f_i \) is \( u \)-reduced, the number of \( u \)-points in \( f_i \) is \( \max \{ 0, b_i \} \), and \( f_i \) is \( u \)-saturated if and only if \( b_i \in \{ -1, 2 \} \). With \( \beta \) and \( Q \) defined as in Lemma 3, the display below lists all possible cases; to each we apply Lemma 3. One \( u \)-operation occurs if \( b_1 + b_2 \geq 2 \), and otherwise no \( u \)-operation occurs. For example, when \( b_1 + b_2 = 4 \), each of \( f_1, f_2 \) has one component with two \( u \)-points, and a single splice or swallow combines these, but then all displayed intervals for \( u \) lie in a single component containing the two \( u \)-points, and no further operation is possible. When \( \beta = 2 \) and \( Q \) is true, we have \( \{ b_1, b_2 \} = \{ -1, 2 \} \), and no \( u \)-operation occurs. \( \blacksquare \)

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>( Q )</th>
<th>#ops</th>
<th>( b_1 + b_2 )</th>
<th>( \max { b_1, b_2 } )</th>
<th>( \epsilon_f(u) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>false</td>
<td>0</td>
<td>-1 or 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>false</td>
<td>0</td>
<td>0 or 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>1</td>
<td>2</td>
<td>2 or 1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Theorem 6 Let \( f_1, \ldots, f_k \) be \( u \)-optimal representations of graphs \( G_1, \ldots, G_k \) that pairwise share only \( u \). Every \( u \)-reduction of \( f_1, \ldots, f_k \) is a \( u \)-optimal representation of \( G = \bigcup_{i=1}^{k} G_i \). The values of \( I(G) \) and \( c(G, u) \) are determined from the numbers \( I(G_i) = |f_i| \) and \( c(G_i, u) = \epsilon_{f_i}(u) \) for each \( i \) as specified in Lemma 3, with each \( G_i \) contributing \( \max \{ 0, c(G_i, u) \} \) to \( \beta \).

Proof. By Corollary 4, it suffices to prove that some \( u \)-optimal representation of \( G \) can be obtained by \( u \)-reduction of some set of \( u \)-optimal representations of \( G_1, \ldots, G_k \). The proof is by induction on \( k \). The statement is trivial (or vacuous) when \( k = 1 \). For \( k > 2 \), we partition the indices \{1, \ldots, k\} into two nonempty collections \( A_1, A_2 \). By Lemma 3, the final size and \( u \)-extent of a \( u \)-reduction does not depend on the order of \( u \)-operations. Let \( H_1 = \bigcup_{i \in A_1} G_i \) and \( H_2 = \bigcup_{i \in A_2} G_i \). By the induction hypothesis, \( u \)-reductions of \( \{ f_i : i \in A_1 \} \) and \( \{ f_i : i \in A_2 \} \) are \( u \)-optimal for \( H_1 \) and \( H_2 \), respectively. Now the induction hypothesis for \( k = 2 \) guarantees that the final \( u \)-reduction is \( u \)-optimal for \( G \).
Hence we have reduced the claim to the case $k = 2$. Let $f$ be a $u$-reduction of $\{f_1, f_2\}$. We will prove that $f$ is $u$-optimal by comparing its size and $u$-extent to a $u$-optimal representation $f'$ of $G$. If we can choose a $u$-optimal $f'$ that is a $u$-reduction of $u$-reduced representations $f'_1, f'_2$ for $G_1, G_2$ (without knowing whether they are $u$-optimal), then we can apply Corollary 5 to compare $f'$ and $f$.

Let $A = V(G_1 - u)$ and $B = V(G_2 - u)$. We choose $f'$ to be a $u$-optimal representation of $G$ that minimizes the number of real intervals contained in $f(u)$ that separate a component of $f'(A)$ and a component of $f'(B)$. We call each such instance a bridge (an interval in $f(u)$ may contain more than one bridge). Since $G$ has no edges between $A$ and $B$, no point in $f'(u)$ belongs to both $f'(A)$ and $f'(B)$, so every bridge is displayed.

Suppose that $f'$ has two bridges not in the same component of $f'$. By cutting these bridges, reflecting and/or translating the resulting components, and reattaching the intervals for $u$, we obtain a representation with the same size and two fewer bridges. If $f'$ has three bridges in the same component, then we cut the first and third of them, reflect the intervening portion of $f'$, and reattach to reduce the number of bridges by two.

\[
\begin{align*}
A & \quad u \quad B \\
\text{two bridges} & \\
& \quad \quad \quad B \quad uA \\
& \quad \quad \quad \quad \quad A \quad u \quad B \\
\end{align*}
\]

\[
\begin{align*}
\text{no bridges} & \\
& \quad \quad \quad A \quad u \quad A \\
& \quad \quad \quad B \quad uB \\
\text{two bridges that remain} & \\
& \quad \quad \quad A \quad u \quad B \quad u \quad A
\end{align*}
\]

Hence $f'$ has at most two bridges, and if $f'$ has two bridges they involve the same component of $f'$. We now extract from $f'$ representations $f'_1$ and $f'_2$ for $G_1$ and $G_2$. First we cut the bridges, and then every component of the resulting representation establishes edges for $G_1$ or $G_2$ but not both. Collecting the components for $G_i$ yields a representation of $G_i$. This is our $u$-reduced representation $f'_i$ unless it has two components that came from cutting two bridges in the same component of $f'$, in which case it has $u$-points in distinct components, and we perform a splice. Since $f'$ was $u$-reduced, $f'_1$ and $f'_2$ are now $u$-reduced. Furthermore, $f'$ itself is a $u$-reduction of $\{f'_1, f'_2\}$, using simple concatenation, one splice, or one swallow if $f'$ has 0, 1, 2 bridges, respectively.

Let $\gamma' = |f'_1| + |f'_2|$. We have $|f'| = \gamma'$ if $f'$ has no bridges, and $|f'| = \gamma' - 1$ if $f'$ has one or two bridges. Since $f'$ is $u$-optimal, we have $e_{f'}(u) = c(G, u)$. Let $b_i = e_{f'_i}(u)$. In our $u$-reduction $f$ of the $u$-optimal representations $f_1, f_2$, let $\gamma = |f_1| + |f_2|$, and let $c_i = e_{f_i}(u)$. Corollary 5 yields
\[ b_1 + b_2 \leq 1 \quad \Rightarrow \quad |f'| = \gamma' \quad \text{and} \quad \epsilon_u(f') = \max\{b_1, b_2\} \]
\[ b_1 + b_2 \geq 2 \quad \Rightarrow \quad |f'| = \gamma' - 1 \quad \text{and} \quad \epsilon_u(f') = b_1 + b_2 - 2 \]
\[ c_1 + c_2 \leq 1 \quad \Rightarrow \quad |f| = \gamma \quad \text{and} \quad \epsilon_u(f) = \max\{c_1, c_2\} \]
\[ c_1 + c_2 \geq 2 \quad \Rightarrow \quad |f| = \gamma - 1 \quad \text{and} \quad \epsilon_u(f) = c_1 + c_2 - 2 \]

From the optimality of \( f', f_1, f_2 \), we have \(|f| \geq |f'|\) and \(|f'_i| \geq |f_i|\), and thus \( \gamma' \geq \gamma \). We consider three cases. In each case, we show first that \(|f| = |f'|\) and then that \( \epsilon_u(f) \geq \epsilon_u(f')\), which completes the proof that \( f \) is \( u \)-optimal.

Case 1: \( c_1 + c_2 \geq 2 \). Here \(|f'| \leq |f| = \gamma - 1 \leq \gamma' - 1 \leq |f'|\). Equality holds throughout, and \( \gamma = \gamma' \) implies \(|f_i| = |f'_i|\). This implies \( c_i \geq b_i \), by the \( u \)-optimality of \( f_i \). We now have \( \epsilon_u(f) = c_1 + c_2 - 2 \geq b_1 + b_2 - 2 = \epsilon_u(f') \), where the last equality follows from \(|f'| = \gamma' - 1\).

Case 2: \( b_1 + b_2 \leq 1 \). Here \(|f'| = \gamma' \geq \gamma \geq |f| \geq |f'|\). Equality holds throughout, and \( \gamma = \gamma' \) implies \(|f_i| = |f'_i|\). This again implies \( c_i \geq b_i \). We now have \( \epsilon_u(f) = \max\{c_1, c_2\} \geq \max\{b_1, b_2\} = \epsilon_u(f') \), where the first equality follows from \(|f| = \gamma\).

Case 3: \( c_1 + c_2 \leq 1 \) and \( b_1 + b_2 \geq 2 \). By symmetry in the indices, we may assume that \( b_1 > c_1 \). Now the \( u \)-optimality of \( f_1 \) yields \(|f'_1| > |f_1|\). Since \(|f'_2| \geq |f_2|\), we obtain \( \gamma' > \gamma \). Now \(|f'| = \gamma' - 1 \geq \gamma = |f| \geq |f'|\). Again equality holds throughout. Hence \(|f_1| = |f'_1| - 1\) and \(|f_2| = |f'_2|\), so \( c_2 \geq b_2 \) by the \( u \)-optimality of \( f_2 \). We now have \( \epsilon_u(f) = \max\{c_1, c_2\} \geq c_2 \geq b_2 \geq b_1 - 2 = \epsilon_u(f') \).

\[ \blacksquare \]

### 4 Clique Blocks

Theorem 6 permits a recursive step when \( u \) is a cut-vertex or when \( G \) is disconnected. To complete an algorithm, we must be able to compute \( I(G) \) and \( c(G, u) \) recursively when \( G \) is connected and \( u \) is not a cut-vertex.

In [7], we presented an algorithm for computing \( I(G) \) when \( G \) is a tree. In a tree, every non-leaf vertex is a cut-vertex. In essence, the tree algorithm deletes the root \( u \) and computes \( c(T, u) \) recursively using the resulting subtrees. Our present approach treats a root \( u \) of degree at least 2 as a cut-vertex and includes an edge to \( u \) in each subtree. We then compute \( c(T, u) \) from the resulting \( \{c(T_i, u)\} \) as described in Section 3. To complete a tree algorithm, we need only describe the recursive computation of \( c(T, u) \) and an optimal representation when the root \( u \) has degree 1.

This separates our earlier algorithm into two simpler steps that generalize. The treatment of cut-vertices in Section 3 is valid for all classes of graphs, but recursively computing the
code for a non-cut-vertex requires special properties.

We begin with a way to combine representations of disjoint graphs that adds a complete graph. A \textit{v, wsplice} is an operation of translating and reflecting components of a representation to make a \textit{v}-point and a \textit{w}-point consecutive, followed by extending their intervals to intersect (switching the \textit{v}-point and the \textit{w}-point in the endpoint ordering). If \( U \) is a set of vertices and \( x \in \mathbb{R} \) is such that \( f^{-1}(x) \subseteq U \) for some representation \( f \), then \textit{piling} \( U \) on \( x \) is the operation of adding one interval for each vertex of \( U - f^{-1}(x) \) such that the new intervals contain \( x \) but contain no endpoint of \( f \) (except \( x \) if \( x \) is an endpoint).

Let \( f_1, \ldots, f_k \) be representations of edge-disjoint graphs \( G_1, \ldots, G_k \), with \( u_i \in V(G_i) \). Let \( u \) be a vertex in none of these graphs, and let \( U' = \{ u_1, \ldots, u_k \} \) and \( U = U' \cup \{ u \} \). A \textit{U-completion} of \( f_1, \ldots, f_k \) is a representation \( f \) obtained by 1) concatenating \( f_1, \ldots, f_k \), 2) applying a single \( u_i, u_j \)-splice if one is possible, 3) piling \( U \) on a legal point \( x \) assigned to the largest possible subset of \( U \), and 4) extending the new interval for \( u \) to create as many \( u \)-points as possible. Below we illustrate the four cases of \( U \)-completion described in Lemma 7 (here \( k = 3 \)).

\begin{tabular}{c|c|c}
    \text{case} & \( |f| - \sum |f_i| \) & \( \epsilon_u(f) \) \\
    \hline
    at least two of \( c_1, \ldots, c_k \) are positive & \( k - 1 \) & \(-1\) \\
    exactly one of \( c_1, \ldots, c_k \) is positive & \( k \) & \(1\) \\
    \( \max_i c_i = 0 \) & \( k \) & \(-1\) \\
    \( \max_i c_i = -1 \) & \( k + 1 \) & \(2\) \\
\end{tabular}

\textbf{Proof.} If step (3) of the \( U \)-completion piles \( U \) on a point \( x \) that was assigned to \( l \) vertices of \( U \), then \( |f| = \sum |f_i| + k + 1 - l \). Since each \( u_i \) belongs only to \( G_i \), we have \( l \leq 2 \). Equality holds only if the \( U \)-completion begins with a \( u_i, u_j \)-splice, which happens if and only if two of \( c_1, \ldots, c_k \) are positive. In this case, no \( u \)-points result and \( u \) is not displayed in \( f \) (we may forbid isolated useless intervals from representations).

If one of \( c_1, \ldots, c_k \) is positive, then we pile \( U \) on a point in an interval containing an endpoint of its component and extend \( f(u) \) to obtain one \( u \)-point. If \( \max_i c_i = 0 \), then we pile \( U \) on a displayed interval for a neighbor of \( u \), and \( u \) is non-displayed. If \( \max_i c_i = -1 \),
then each $u_i$ is non-displayed in $f_i$; we introduce $k+1$ new intervals containing an unoccupied point on the line and extend $f(u)$ to obtain two $u$-points.

Lemma 7 implies that every $U$-completion of $f_1, \ldots, f_k$ has the same size and $u$-extent.

Consider a clique $Q$ in a graph $G$. In a representation $f$, a pile for $Q$ is a maximal interval $I \subseteq \mathbb{R}$ such that $f(Q)$ covers $I$, at least two vertices of $Q$ are assigned points of $I$, and no vertex outside $Q$ is assigned a point of $I$. The size of a pile $I$ is the number of vertices assigned points of $I$, and its base is the vertex or pair of vertices whose images are leftmost and rightmost in $I$. Because $Q$ is a clique, we may assume that the nonempty sets $f(v) \cap I$ are pairwise intersecting when $I$ is a pile for $Q$. Above we illustrated four piles for a clique $U$; each pile has size four, two have a singleton base, and two have a doubleton base. We abuse terminology by saying that a pile $I$ contains the vertices whose images intersect the interval $I$.

**Lemma 8** Let $G$ be a connected graph with a non-cut-vertex $u$. If the block of $G$ containing $u$ is a complete graph with vertex set $U$, then $G$ has a $u$-optimal representation having only one pile for $U$.

**Proof.** Let $f$ be a representation of $G$ having more than one pile for $U$. It suffices to show that in every case we can obtain a representation $f'$ with smaller size or fewer piles for $U$, with $\epsilon_u(f') = \epsilon_u(f)$ when we merely reduce the number of piles. Note that inserting or deleting non-displayed intervals does not affect any $v$-extent.

If some pile contains all of $U$, then we can delete the non-base intervals from other piles and shrink the intersections of the bases to obtain a representation with one pile for $U$. This does not increase the size or decrease any $v$-extent.

Let $m = |U|$. When $m = 2$, every pile contains all of $U$. When $m = 3$, let $U - \{u\} = \{v, w\}$. If no pile contains all of $U$, then the three edges $uv, uw, vw$ occur in disjoint piles. Since $v$ and $w$ have no common neighbor other than $u$, the edge $vw$ is displayed. Since $u$ has no neighbor outside $\{v, w\}$, we can place a small interval for $u$ in $f(v) \cap f(w)$ and delete the other two intervals for $u$ to reduce size. Thus we may assume that $m \geq 4$ (and no pile contains all of $U$).

If the entire base of a pile $P$ is an interval for $w$, then $w$ must appear in another pile $P'$, because no pile contains all of $U$. The non-displayed intervals in $P$ can be moved to become small intervals in $P'$. This reduces the number of piles without increasing size or change any of $\epsilon_f$. Thus we may assume that every pile has a doubleton base.
We may also assume that the complete graph determined by piles are edge-disjoint. Suppose that vertices \( v, w \) both appear in piles \( P \) and \( P' \) (as base intervals or not). We insert into \( P' \) small intervals for all vertices that are in \( P \) but not in \( P' \), delete from \( P \) the non-base intervals, and shrink the intersection of the base intervals in \( P \) to display their endpoints. This does not increase the size or reduce any of \( \epsilon_f \).

We may also assume that each vertex of \( U \) appears in at most two bases. If \( v \in U \) appears in bases of three piles, then we may assume that it fills the left end of two of them, with \( x, y \) filling the right ends of these bases (see Figure). Let \( A \) and \( B \) be the sets of non-base intervals in the two piles. We reflect the portion of the representation between the two piles and extend the interval for \( x \) at the right to meet the interval for \( y \). On this base we pile intervals for \( A \cup B \cup \{v\} \). Size does not increase, because we combine the two intervals for \( v \) at the left of the reflected portion. Since the endpoints of these intervals were not displayed in \( f \), we have not decreased \( \epsilon_f(v) \), but we have reduced the number of piles.

\[
\begin{array}{c}
A \quad C \\
v \quad x \\
B \quad y
\end{array}
\rightarrow
\begin{array}{c}
A \cup B \cup \{v\} \\
v \quad C' \\
x \quad y
\end{array}
\]

Since no pile contains all of \( U \), each vertex of \( U \) appears in at least two piles. Suppose that some vertex \( v \) appears in only two piles \( P, P' \). Since piles induce edge-disjoint complete graph, \( P, P' \) can share only \( v \). Since no other pile contains \( v \), all of \( U \) has intervals in \( P \) or \( P' \). Now every pile other than \( P, P' \) has only two intervals, else it would share two intervals with \( P \) or \( P' \). Thus every interval in a pile other than \( P, P' \) is a base interval. Since \( m \geq 4 \), we may assume that \( P \) has two vertices \( x, y \) other than \( v \). Let \( w \) be a base vertex of \( P' \) other than \( v \). Now \( w \) appears as a base vertex in distinct piles with all of \( v, x, y \). This contradicts our restriction that each vertex of \( U \) appears in at most two bases.

We are left only with the case where each element of \( U \) appears in at least three piles. Since each element of \( U \) appears in at most two bases, the piles contain at least \( 3m - 2m \) non-base intervals. We can now delete all the non-base intervals and pile \( U \) on a single base to reduce the size of the representation.

\[\blacksquare\]

**Theorem 9** Let \( G \) be a connected graph having a non-cut-vertex \( u \) in a block \( Q \) that is a complete graph. Let \( G_1, \ldots, G_k \) be the components of \( G - u - E(Q) \), and let \( u_i \) be the vertex of \( G_i \) in \( U = V(Q) \). For \( 1 \leq i \leq k \), let \( f_i \) be a \( u_i \)-optimal representation of \( G_i \). Every \( U \)-completion of \( f_1, \ldots, f_k \) is a \( u \)-optimal representation of \( G \).
Proof. Let \( f \) be a \( U \)-completion of \( f_1, \ldots, f_k \); by construction, \( f \) has only one pile for \( U \). Let \( f' \) be a \( u \)-optimal representation of \( G \) with one pile for \( U \), which exists by Lemma 8. We follow the method of Theorem 6, showing that \( f' \) is a \( U \)-completion of representations \( f'_1, \ldots, f'_k \) of \( G_1, \ldots, G_k \) and using the \( u \)-optimality of \( f' \) and \( \{ f_i \} \).

Since \( f' \) has only one pile for \( U \), we may assume that \( f'(u) \) is a single interval in this pile. We obtain \( f'_1, \ldots, f'_k \) by deleting the non-displayed intervals of the pile, shrinking the intervals of the base to separate them (if the base has two intervals), deleting the interval for \( u \) if it was part of the base (this happens if and only if \( \epsilon_u(f') \geq 1 \)), and letting \( f'_i \) consist of the components of the resulting representation that contain intervals for vertices of \( G_i \).

As in Theorem 6, let \( \gamma = \sum |f_i|, c_i = \epsilon_{f_i}(u), \gamma' = \sum |f'_i|, \) and \( b_i = \epsilon_{f'_i}(u) \). The difference between \( |f'| \) and \( \gamma' \) is \( k + 1 - l' \) if the base has \( l' \) vertices other than \( u \), where \( 0 \leq l' \leq 2 \). We may assume that the correspondence between \( l' \) and the values \( b_1, \ldots, b_k \) is the same as the correspondence between \( l = (k + 1) - (|f| - \gamma) \) and the values \( c_1, \ldots, c_k \) defined in the statement of Lemma 7, because otherwise a \( U \)-completion of \( \{ f'_i \} \) would be smaller than \( f' \).

We have \( |f| \geq |f'| \) and \( |f'_i| \geq |f_i| \), and thus \( \gamma' \geq \gamma \). If \( \gamma' = \gamma \), then \( |f'_i| = |f_i| \) for all \( i \). Optimality of \( f_i \) then implies that \( c_i \geq b_i \) for all \( i \). This condition is impossible when \( l' > 0 = l \) (max \( c_i = -1 < 0 = \max b_i \)) and when \( l' = 2 > l \) (at least two of \( b_i \)) and at most one of \( \{ c_i \} \) are positive). Thus we may assume that \( \gamma' > \gamma \) when \( l' > l \).

If \( l' \leq l \), then \( |f'| \leq |f| = \gamma + (k + 1) - l \leq \gamma' + (k + 1) - l' = |f'| \). Equality holds throughout, and thus \( |f| = |f'|, \gamma' = \gamma \), and \( l = l' \). When \( l = l' \), the \( u \)-extents of \( f \) and \( f' \) differ only when \( l = l' = 1 \). From the statement of Lemma 7 and the condition \( c_i \geq b_i \) that follows from \( \gamma' = \gamma \), this requires that one of \( \{ c_i \} \) is positive and \( \max b_i = 0 \). This yields \( \epsilon_u(f) = 1 \) and \( \epsilon_u(f') = -1 \), so \( f \) is at least as \( u \)-optimal as \( f' \).

When \( l' > l \), we already have \( \gamma' > \gamma \). If \( l' = l + 1 \), then \( |f'| \leq |f| = \gamma + (k + 1) - l \leq (\gamma' - 1) + (k + 1) - l' + 1 = |f'| \). Again equality holds throughout, and \( |f| = |f'| \). If \( l = 0 \), then \( \epsilon_u(f) = 2 \), while \( l' = 2 \) yields \( \epsilon_u(f') = -1 \). Thus \( \epsilon_u(f) \geq \epsilon_u(f') \).

Finally, suppose that \( l' = 2 \) and \( l = 0 \). If \( |f| = |f'| \), then \( \epsilon_u(f) = 2 > -1 = \epsilon_u(f') \). Thus we may assume that \( |f| > |f'| \) and \( \gamma' > \gamma \), which yields \( |f'| + 1 \leq |f| = \gamma + (k + 1) \leq (\gamma' - 1) + (k + 1) = |f'| + 1 \). Again equality holds throughout. With \( \gamma' = \gamma + 1 \), we have \( |f'_i| = |f_i| \) and \( c_i \geq b_i \) for all but one value of \( i \). This is impossible when \( l' = 2 \) and \( l = 0 \).

Since \( f' \) is \( u \)-optimal, we in fact have obtained contradictions in all cases unless \( |f| = |f'| \) and \( \epsilon_u(f) = \epsilon_u(f') \), so we conclude that \( f \) is \( u \)-optimal.

\( \blacksquare \)
5 Cycle Blocks

In order to complete the algorithm for cacti, we must also show how to construct a \( u \)-optimal representation when \( u \) is a non-cut-vertex whose block is a chordless cycle; the case of a 3-cycle is handled as a clique block.

Let \( f_1, \ldots, f_k \) be representations of edge-disjoint graphs \( G_1, \ldots, G_k \), with \( u_i \in V(G_i) \). Let \( u \) be a vertex in none of these graphs. Let \( C \) be the cycle consisting of \( u \) followed by \( u_1, \ldots, u_k \) in order. Let \( G \) be the graph formed by adding the edges of \( C \) to the union \( \bigcup_{i=1}^{k} G_i \). We construct a representation of \( G \) from \( f_1, \ldots, f_k \), taking advantage of displayed endpoints for vertices of \( C \) to minimize the number of added intervals.

Let \( c_i = c(G_i, u_i) \). Let \( \sigma \) denote the cycle code sequence \( u, c_1, \ldots, c_k \). ****Describe operations for producing the desired \( u \)-optimal representation for a cycle block. Requires thought to determine the best presentation. An analogue of Lemma 7 specifies effect of each operation. A theorem is needed to prove \( u \)-optimality of the result.

6 The Algorithm and Applications

****State the algorithm semi-formally. Cite the theorems proved along the way to justify that the algorithm works.

****Apply the algorithm to solve the extremal problems for cacti and block graphs. This has never been written down.

References

