Characterization of $[1, k]$-Bar Visibility Trees

Guantao Chen [1,*], Joan P. Hutchinson[2], Ken Keating[3], Jian Shen[4]

[1] Georgia State University, Atlanta, GA 30303
gchen@gsu.edu

[2] Macalester College, St. Paul, MN 55105
hutchinson@macalester.edu

[3] Emory University, Atlanta, GA 30322
kekeati@emory.edu

[4] Texas State University, San Marcos, TX 78666
js48@txstate.edu

Submitted: May 27, 2006; Accepted: Oct 19, 2006; Published: Oct 27, 2006
Mathematics Subject Classification: 05E25

Abstract

A *unit bar-visibility graph* is a graph whose vertices can be represented in the plane by disjoint horizontal unit-length bars such that two vertices are adjacent if and only if there is a unobstructed, non-degenerate, vertical band of visibility between the corresponding bars. We generalize unit bar-visibility graphs to $[1, k]$-*bar-visibility graphs* by allowing the lengths of the bars to be between $1/k$ and 1. We completely characterize these graphs for trees. We establish an algorithm with complexity $O(kn)$ to determine whether a tree with n vertices has a $[1, k]$-bar-visibility representation. In the course of developing the algorithm, we study a special case of the knapsack problem: Partitioning a set of positive integers into two sets with sums as equal as possible. We give a necessary and sufficient condition for the existence of such a partition.

1 Introduction

A *bar-visibility graph*, or BVG for short, is a graph whose vertices can be represented in the plane by disjoint horizontal bars such that two vertices are adjacent if and only if there

*Partially supported by NSA grant H98230-04-1-0300 and NSF grant DMS-0500951

is an unobstructed, non-degenerate, vertical band of visibility between the corresponding bars. The study of BVGs is motivated by VLSI design. Several layout compaction strategies for VLSI are based on the concept of visibility between parallel segments. Two parallel segments are visible if they can be joined by a segment orthogonal to them without intersecting any other segment. This precisely matches the definition of a BVG.

Wismath [11] and Tamassia-Tollis [9] independently characterize BVGs as planar graphs having a planar embedding with all cutpoints on a common face. A BVG is called a *unit bar visibility graph*, or UBVG for short, if all horizontal bars have length 1. A *caterpillar* is a tree in which a single path (the *spine*) is incident to (or contains) every edge. Dean and Veytsel [6] show that a tree T is a UBVG if and only if T is a subdivision of a caterpillar with maximum degree three. Dean, Gethner, and Hutchinson [4] give some combinatorial and geometric characterizations of the triangulated polygons (2-connected maximal outer-planar graphs) that are UBVGs. Bose-Dean-Hutchinson-Shermer [3] and Dean-Hutchinson [5] study the rectangular visibility graphs where the adjacency of rectangles is determined by horizontal and vertical visibility.

A BVG is called a $[1, k]$-*Bar Visibility Graph*, or kBVG for short, if all bar lengths are between $1/k$ and 1. Equivalently, the ratio of the length of the longest bar to the length of the shortest bar is at most k. We characterize all $[1, k]$-Bar Visibility trees, or kBVTs for short, and establish an $O(kn)$ algorithm to determine whether a tree with n vertices is a kBVT.

We follow the notations in [10]. All graphs considered here are simple graphs. Let G be a graph. A path P of G is *maximal* if there is no path Q containing P as a proper subpath. For any $v \in V(G)$, let $N(v)$ denote the set of neighbors of v and $d(v) := |N(v)|$ be the degree of v. A *rooted tree* is a tree with a specific vertex as its root. Let T be a rooted tree and $v_0 \in V(T)$ be the root. A vertex $v \in V(T) - \{v_0\}$ is called a *leaf* if $d(v) = 1$. Let L denote the set of all leaves of T.

2 Characterization of kBVT

We begin by generalizing the definition of a caterpillar. Let k be a nonnegative integer. A tree T is called a generalized k-*caterpillar* if there exists a spine P such that each component of $T - V(P)$ is a rooted tree with at most k leaves, where the root is the vertex adjacent to the spine. The set of all leaves of the trees in $T - V(P)$ is denoted by $L(\overline{P})$. Note that $L(\overline{P}) \cap V(P) = \emptyset$. According to the definition of k-caterpillar, a generalized 0-caterpillar is an ordinary caterpillar and in this case $L(\overline{P}) = \emptyset$; a generalized 1-caterpillar is a subdivision of a caterpillar. An example of a generalized 2-caterpillar is shown in Figure 1.

Let T be a tree with the spine P. Let $\{X, Y\}$ be a *partition* of $L(\overline{P})$, i.e. $X \cup Y = L(\overline{P})$ and $X \cap Y = \emptyset$. A partition $\{X, Y\}$ of $L(\overline{P})$ is called a *proper partition with respect to* P if, for any $x \in X$ and any $y \in Y$, x and y are in different components of $T - V(P)$. A

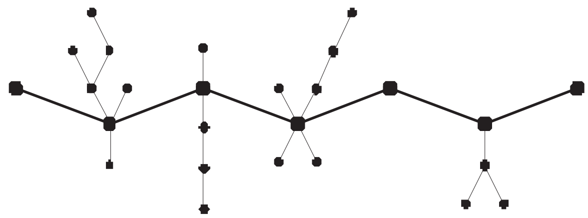

Figure 1: A generalized 2-caterpillar

proper partition of $L(\overline{P})$ is shown in Figure 2.

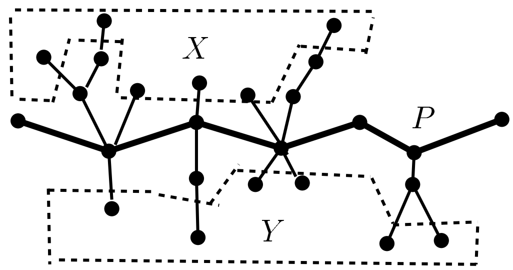

Figure 2: A proper partition of L

A proper partition $L(\overline{P}) = X \cup Y$ with respect to P is called *strictly k-bounded* if $|X \cap V(T_i)| \leq k$ and $|Y \cap V(T_i)| \leq k-1$ for each component T_i of $T - E(P)$. Note that a tree T_i in $T - E(P)$ contains several trees in $T - V(P)$ and a vertex on P. So if $L(\overline{P})$ has a strictly k-bounded partition with respect to a path P of T, then T is a generalized k-caterpillar with the spine P and the converse may not hold. Figure 3 shows that the proper partition of Figure 2 is strictly 3-bounded.

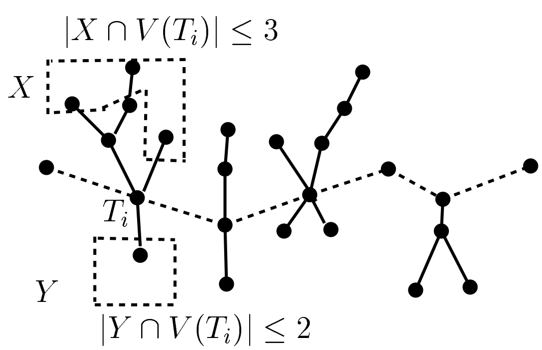

Figure 3: A strictly 3-bounded partition

Theorem. (2.1) *A tree T is a kBVT if and only if there exists a spine P such that $L(\overline{P})$ has a strictly k-bounded partition with respect to P.*

Proof. "$\Longrightarrow$" Let T be a kBVT and $\mathcal{I}$ be the set of bars representing the vertices of T. Let $P := I_1 I_2 \cdots I_m$ be a *maximal* path in T, where I_1 and I_m are the bars with the left-most endpoint and the bar with the right most endpoint, respectively. Note that there may be more than one bar with the left-most endpoint and more than one bar with the right most endpoint. Let $I_i = [a_i, b_i]$, where a_i and b_i denote the left and right endpoints of I_i, respectively. Clearly, $a_i < b_{i-1} \leq b_i$ and $a_i \leq a_{i+1} < b_i$ for each $1 < i < m$. Let T_i, $1 \leq i \leq m$, be the component of $T - E(P)$ containing I_i.

For each $i = 1, \ldots, m$, we view T_i as a rooted tree with root I_i. If one of bars I_{i-1} or I_{i+1} is below bar I_i, let X_i be the set of leaves of T_i above I_i and Y_i the set of leaves of T_i below I_i. Otherwise, let X_i be the set of leaves of T_i above I_i and Y_i be the leaves of T_i below I_i. Then no bars of X_i are visible by each other or by bars I_{i-1} or I_{i+1}; or else T would contain a cycle. Since bar I_i has length at most 1 and each bar of X_i has length at least $1/k$, we have $|X_i| \leq k$. By our definition of Y_i, the total length of all bars of Y_i is at most $b_i - b_{i-1} < 1$ and thus $|Y_i| \leq k - 1$.

Let $X = \bigcup_{1 \leq i \leq m} X_i$ and $Y = \bigcup_{1 \leq i \leq m} Y_i$. Then $X \cup Y$ forms a strictly k-bounded partition of $L(\overline{P})$ with respect to P.

"$\Longleftarrow$" Suppose $\{X, Y\}$ is a strictly k-bounded partition with respect to a path $P = v_1 v_2 \ldots v_m$. For each i, $1 \leq i \leq m$, we define the corresponding bar for v_i as $I_i = [i-1-(i-1)\epsilon, i-(i-1)\epsilon]$, where ϵ is a positive real number much smaller than $1/m$. We arrange the bars I_1, I_2, …, I_m such that I_1, I_3, I_5, … are on the horizontal line $y = 1/2$ and I_2, I_4, I_6, … are on the horizontal line $y = -1/2$.

Let T_i be the component of $T - E(P)$ containing v_i. Let $X_i = X \cap V(T_i)$ and $Y_i = Y \cap V(T_i)$. We establish an algorithm to define the corresponding bars for all vertices in each T_i. Without loss of generality, we may assume that i is odd and $|X_i| \geq |Y_i|$. Since T_i is a tree, T_i has a plane presentation such that no two vertices are in the same vertical line and all vertices of X_i are above v_i and all vertices of Y_i are below v_i. Furthermore, we assume that in the plane representation the vertices x_1, x_2, … of X_i are ordered from left to right and the vertices y_1, y_2, … of Y_i are ordered from left to right too. Let I_{x_1}, I_{x_2}, … be disjoint bars each of length $1/k$ listed from left to right above I_i.

For each x_j let $P_j = P[v_i, x_j]$ be the unique path of T_i from v_i to x_j, and similarly for each y_j let $Q_j = P[v_i, y_j]$. Let $T_i(X) = \bigcup_j P[v_i, x_j]$ and $T_i(Y) = \bigcup_j Q[v_i, y_j]$. Then each vertex of T_i is either in $T_i(X)$ or $T_i(Y)$. For each $w \in T_i(X)$, by the plane representation of T_i, the j's such that $w \in P_j$ form a sequence of integers s, $s+1$, …, $s+t$. Then, define I_w to be the bar covering from the left end point of I_s to the right end point of I_t. Further, we arrange the heights of the bars according to the distance between the vertex and v_i such that if $dist(v_i, w) = \ell$, then I_w is placed on the line $y = \ell$. Similarly, we can define the corresponding bars for all vertices in $T_i(Y)$. Thus we obtain those bars whose bar-visibility graph is T. □

Corollary. (2.2) *The maximum degree of a kBVT is at most* $2k+1$.

Proof. By Theorem (2.1), the leaf set of a kBVT has a strictly k-bounded partition with respect to a path (the spine). Thus each vertex on the spine has degree at most

$2 + k + (k-1) = 2k+1$ and each vertex not on the spine has degree at most $k+1$. $\square$

3 Evenly partitioning a set of integers

In this section, we establish some foundation for an algorithm to realize kBVTs. A key part of the algorithm depends on a special case of the following knapsack problem. Suppose there is a knapsack of capacity $c > 0$ and N items. Each item has a value $v_i > 0$ and a weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if selected, and $\delta_i = 0$ otherwise) that fits

$$\sum_{i=1}^{N} \delta_i \cdot w_i \leq c, \text{ and the total value} \sum_{i=1}^{N} \delta_i \cdot v_i \text{ is maximized.}$$

This problem is also named as the 0-1 or binary knapsack (each item may be taken (1) or not (0)), in contrast to the fractional knapsack problem. It is also called the bounded knapsack (BKP) because there are a limited number of items, in contrast to the unbounded knapsack problem. The decision problem is, given some items of different values and weights of a knapsack, to determine whether there is a subset with total value exceeding a certain number? The decision problem is NP-complete. For literature on the Knapsack problem, we referrer to Silvano Martello and Paolo Toth [8].

A special case of the Knapsack problem is called the *minimum partition problem.* Let S be a set of positive integers. The minimum partition problem is to find a partition of $S = A \cup B$ such that $\left|\sum_{a_i \in A} a_i - \sum_{a_i \in B} a_i\right|$ is minimum. The minimum partition problem is NP-complete. Let P be an optimization problem, and let A be an approximation algorithm for P. The *domination ratio* $domr(A, n)$ is the maximum real q such that the solution $x(I)$ obtained by A for any instance I of P of size n is not worse than at least a fraction q of the feasible solutions of I. Recently, Alon, Gutin, and Krivelevich [2] found a deterministic, polynomial-time algorithm for the problem whose domination ratio is $1 - o(1)$, improving an earlier algorithm [7] with a domination ratio of $1/2$. We are interested in the following partition: A partition of $S = A \cup B$ is called *k-balanced* if $\sum_{a_i \in A} a_i \leq k$ and $\sum_{a_i \in B} a_i \leq k-1$. A key part of our algorithm for the recognition of kBVTs in Section 4 depends on finding a k-balanced partition of S. Also the problem may be interesting in its own right.

Question (3.1) *Let $S = \{a_1, a_2, \ldots, a_d\}$ be a set of positive integers such that $\sum_{1 \leq i \leq d} a_i \leq 2k-1$. When does S have a k-balanced partition?*

One may assume in Question (3.1) that $\sum_{a_i \in S} a_i = 2k-1$, since otherwise one can add $2k-1-s$ elements of 1's. So we consider the following equivalent question.

Question (3.2) *Let $S = \{a_1, a_2, \ldots, a_d\}$ be a set of positive integers with sum equal to s. Is there an efficient algorithm to determine whether S can be partitioned into two sets with sums as equal as possible; that is, to determine whether S has a partition $A \cup B$*

such that the sum of all integers in A is $\lceil s/2 \rceil$ (and thus the sum of all integers in B is $\lfloor s/2 \rfloor$)?

Let $C[i, S]$ denote the logical statement that S contains a subset A such that the sum of all elements of A is i. To determine the truth value of $C[i, S]$ for a general i is NP-complete in terms of d. The following algorithm with complexity $O(dk)$ for finding the truth values of all $C[i, S]$, $1 \le i \le k$, is suggested in [1]. (This is not surprising since k may be as big as 2^d.)

Algorithm 1.

Input: A positive integer i and a set $S = \{a_1, a_2, \cdots, a_d\}$ of positive integers.

Output: "**True**" if $C[i, S]$ is true, and "**False**" otherwise.

Initialization: $A = \emptyset$, $C[0, A] =$ true, $C[i, A] =$ false for all $i \le k$.

For $j = 1 \dots d$
    For $i = 1 \dots k$
        $C[i, A \cup \{a_j\}] := C[i, A] \vee C[i - a_j, A]$
    $A := A \cup \{a_j\}$

% $A \cup \{a_j\}$ has a subset with sum i if and only if A has a subset with sum either i or $i - a_j$. Define $C[i - a_j, A] =$ false if $i - a_j < 0$.

Now we give a necessary and sufficient condition to determine whether S can be partitioned into two sets with sums as equal as possible. Define $S_1 = \{a_1, \dots, a_{d-2}, a_{d-1} + a_d\}$ and $S_2 = \{a_1, \dots, a_{d-2}, |a_{d-1} - a_d|\}$.

Lemma. (3.3) *A set $S = \{a_1, a_2, \dots, a_d\}$ can be partitioned into two sets with sums as equal as possible if and only if at least one of S_1 and S_2 can be partitioned into two sets with sums as equal as possible.*

Proof. Suppose S can be partitioned into two sets with sums as equal as possible. Then a_{d-1} and a_d are in the same set of the partition if and only if S_1 can be partitioned into two sets with sums as equal as possible; and a_{d-1} and a_d are in different sets of the partition if and only if S_2 can be partitioned into two sets with sums as equal as possible. □

The following condition is suggested by the sequence: 2^i, $i = 0, 1, \dots, d$.

Lemma. (3.4) *Suppose the elements of S are in non-descending order: $a_1 \le a_2 \le \ldots \le a_d$. Suppose*

$$a_i \le 1 + \sum_{j=1}^{i-1} a_j \text{ for all } i,\ 1 \le i \le d.$$

Then S can be partitioned into two sets with sums as equal as possible. Furthermore, an algorithm with complexity $O(d)$ exists for partitioning S into two sets with sums as equal as possible.

Proof. We prove by induction on d. If $d = 1$, then $a_1 \le 1$ and the lemma is trivial. Now suppose $d \ge 2$ and the lemma holds for $d - 1$. We re-order the elements of S_2: $a_1 \le a_2 \le \ldots \le a_k \le a_d - a_{d-1} \le a_{k+1} \le \ldots \le a_{d-2}$, where $0 \le k \le d - 2$. By Lemma (3.3), it suffices to show that S_2 satisfies the inequalities in Lemma (3.4). Since S satisfies these inequalities, one only needs to verify

$$a_d - a_{d-1} \le \begin{cases} 1 + \sum_{j=1}^{d-2} a_j = 1 + \sum_{j=1}^{k} a_j & \text{if } k = d-2, \\ a_{k+1} \le 1 + \sum_{j=1}^{k} a_j & \text{if } k < d-2. \end{cases}$$

The above proof also suggests an algorithm with complexity $O(d)$ to partition S into two sets with sums as equal as possible. $\square$

A set S is called *good* if (after reordering if necessarily) the elements of S satisfy the inequality conditions in Lemma (3.4). A set S is called *potentially good* if either S itself is good or S_1 is potentially good or S_2 is potentially good.

Theorem. (3.5) *S can be partitioned into two sets with sums as equal as possible if and only if S is potentially good.*

Proof. "$\Longleftarrow$" (Use induction on d.) If S is good, the theorem follows from Lemma (3.4). If either S_1 or S_2 is potentially good, then by the induction hypothesis, either S_1 or S_2 can be partitioned into two sets with sums as equally as possible. Thus, by Lemma (3.3), S can be partitioned into two sets with sums as equal as possible.

"$\Longrightarrow$" (Use induction on d.) Suppose S can partitioned into two sets with sums as equal as possible into A, B. If a_{d-1} and a_d are in the same set, then S_1 can be partitioned into two sets with sums as equal as possible. By the induction hypothesis, S_1 is potentially good. Similarly, if a_{d-1} and a_d are in different sets, then S_2 is potentially good. Thus S is potentially good. $\square$

Theorem (3.5) does not help much in general since one needs to check $O(2^{d-1})$ sets in the worst case to determine whether S is potentially good. Nevertheless, it might be helpful for the "average case", especially when S contains "many" elements.

Define $S^{(1)} = \{S_1, S_2\}$ and $S^{(t)} = S_1^{(t-1)} \cup S_2^{(t-1)}$ for $t \ge 2$. We propose the following two questions.

Question (3.6) *Suppose $a_1 \le a_2 \le \ldots \le a_d = N$. Find the average number, denoted $t(d, N)$, such that $S^{(t(d,N))}$ has a good set if S is potentially good.*

Question (3.7) *Suppose $a_1 \le a_2 \le \ldots \le a_d = N$. Also suppose d is "very big" (for example, say, $d \ge N^{1-\epsilon}$ for some $\epsilon > 0$). Does there exist a function $s(d, N)$ such that $S^{(s(d,N))}$ has a good set for almost all potentially good sets S?*

If the answer to Question (3.7) is "yes", then we can have an algorithm with complexity $O(2^{s(d,N)})$ to solve Question (3.1) for almost all S when d is "very big".

4 Algorithm for the recognition of kBVT

In this section we present an algorithm to determine whether a tree can be represented as a kBVT. We start with some lemmas.

Lemma. (4.1) *Let $1 \leq a_1 \leq \ldots \leq a_d$. Then there is a partition of $[d] = A \cup B$ such that*

$$\left|\sum_{i \in A} a_i - \sum_{i \in B} a_i\right| \leq \max\{a_d - a_1, a_1\}.$$

where $[d] := \{1, 2, \cdots, d\}$,

Proof. We use the following algorithm to partition $[d]$ into A and B. We first initialize $A = B = \emptyset$. Then we add $i = d, d-1, \ldots, 1$ to either A or B one by one as follows: If $\sum_{j \in A} a_j \leq \sum_{j \in B} a_j$, let $A := A \cup \{i\}$; otherwise let $B := B \cup \{i\}$.

We prove the lemma by induction on d. Obviously, the lemma holds for $d = 1$. Now suppose $d \geq 2$. Let $A' = A - \{1\}$ and $B' = B - \{1\}$. Applying induction hypothesis to the $d-1$ numbers $a_2, \ldots, a_d$, we have

$$\left|\sum_{i \in A'} a_i - \sum_{i \in B'} a_i\right| \leq \max\{a_d - a_2, a_2\}.$$

By the above algorithm, 1 is added to either A or B whichever has a smaller sum, we have

$$\begin{aligned} \left|\sum_{i \in A} a_i - \sum_{i \in B} a_i\right| &\leq \max\left\{\left|\sum_{i \in A'} a_i - \sum_{i \in B'} a_i\right| - a_1, a_1\right\} \\ &\leq \max\{a_d - a_2 - a_1, a_2 - a_1, a_1\} \leq \max\{a_d - a_1, a_1\}. \end{aligned}$$

□

Corollary. (4.2) *Let $1 \leq a_1 \leq \ldots \leq a_d$ with $\sum a_i \leq 2k$. Then $\{a_i : 1 \leq i \leq d-1\}$ has a k-balanced partition.*

Proof. By lemma (4.1), there is a partition of $[d-1] = A \cup B$ with

$$\left|\sum_{i \in A} a_i - \sum_{i \in B} a_i\right| \leq \max\{a_{d-1} - a_1, a_1\} \leq a_d$$

and

$$\sum_{i \in A} a_i + \sum_{i \in B} a_i \leq 2k - a_d.$$

Thus $\max\{\sum_{i \in A} a_i, \sum_{i \in B} a_i\} \leq k$ and $\min\{\sum_{i \in A} a_i, \sum_{i \in B} a_i\} \leq k-1$. □

Lemma. (4.3) *Suppose the elements of S are in non-descending order: $a_1 \leq a_2 \leq \ldots \leq a_d$. Suppose $S - \{a_j\}$ has a k-balanced partition for some k, $1 \leq k \leq d$. Then $S - \{a_d\}$ has a k-balanced partition.*

Proof. Suppose $[d] - \{j\}$ has a partition $A \cup B$ such that

$$\sum_{i \in A} a_i \leq k \quad \text{and} \quad \sum_{i \in B} a_i \leq k - 1.$$

If $d \in A$, let $A' := A \cup \{a_j\} - \{a_d\}$ and $B' := B$. Otherwise, let $A' := A$ and $B' = B \cup \{a_j\} - \{a_d\}$. Then,

$$\sum_{i \in A'} a_i \leq \sum_{i \in A} a_i \leq k \quad \text{and} \quad \sum_{i \in B'} a_i \leq \sum_{i \in B} a_i \leq k - 1.$$

□

Let T be a tree with ℓ leaves. For each vertex $v \in N(u)$, let $l(u, v)$ be the number of leaves w of T such that either $w = v$ or v is on the unique path connecting u and w; that is, $l(u, v)$ is the number of leaves in the branch containing v of T rooted at u. Thus if $(u, v) \in E$, then $l(u, v) + l(v, u) = \ell$. Let $\{l(u, v) : v \in N(u)\}$ be ordered $l_1(u) \geq l_2(u) \geq \ldots \geq l_d(u)$, where $d = d(u)$ is the degree of u. Obviously, if T is a $kBVT$ and $\ell - l_1(u) \geq k + 1$, then u must be on the spine of T.

Lemma. (4.4) *Let T be a tree with ℓ leaves. If $\ell \leq 2k$, then T is a kBVT; If $\ell \geq 2k+1$, then there exists some vertex u with $\ell - l_1(u) \geq k + 1$.*

Proof. We first contract all vertices of degree 2; that is, for each u with $d(u) = 2$, we delete u and connect the two neighbors of u by an edge. So, we may assume all non-leaves of T have degrees at least 3. The lemma is trivial if T is a star. Now suppose T is not a star. Among all edges (u, v) with u, v being non-leaves, choose one so that $\max\{l(u, v), l(v, u)\}$ reaches minimum. Without loss of generality, let's assume $l(u, v) \leq l(v, u)$.

Claim 1: $l(u, v) = l_1(u) = \max\{l(u, w) : w \in N(u)\}$.
Or else, suppose $l(u, w) > l(u, v)$ for some $w \in N(u)$. Let $l(u, v) = A$, $l(u, w) = B$, and $\sum_{j \in N(u) \setminus \{v,w\}} l(u, j) = C$. By our assumption, we have $A = l(u, v) \leq l(v, u) = B + C$.Since $\max\{l(u, v), l(v, u)\} \leq \max\{l(u, w), l(w, u)\}$, we have $B + C \leq \max\{B, A + C\}$. This implies that $B + C \leq A + C$, since $B + C > B$. Thus, $B \leq A$ a contradiction.

If $\ell \geq 2k+1$, by Claim 1, $\ell - l_1(u) = \ell - l(u, v) = l(v, u) \geq \lceil (l(v, u) + l(u, v))/2 \rceil = \lceil \ell/2 \rceil \geq k+1$. Now suppose $\ell \leq 2k$. Since $\sum_{j \in N(u)} l(u, j) = \ell \leq 2k$, by Claim 1 and Corollary (4.2), $\{l(u, j) : j \in N(u) \setminus v\}$ has a k-balanced partition. Also $\{l(v, j) : j \in N(v) \setminus u\}$ has a k-balanced partition since $\sum_{j \in N(v) \setminus u} l(v, j) = l(u, v) \leq (l(u, v) + l(v, u))/2 \leq k$. Thus T is a representation of a kBVT with the edge (u, v) as its spine. □

In order to establish an efficient algorithm to locate the u such that $\ell - \ell_1(u) \geq k + 1$, we impose a weight on each leave of T. Let $L(T)$ denote the set of leaves of T and

$\omega : L(T) \mapsto \mathbb{N}^+$. Initially, we assume $\omega = 1$ if no weight function is mentioned. The following algorithm calculates $l(u, v)$ for all $uv \in E(T)$ with complexity of $O(n)$ for any tree T with n vertices.

Algorithm 2.

Input: A tree T with n vertices and a weight function ω on $L(T)$.

Output: $\ell(u, v)$ for all pairs of adjacent vertices u and v.

1. Using the Breadth-first Search, BFS, to form a sequence of nested subtrees $T_0 = T \supset T_1 \supset \cdots \supset T_m$ such that T_m is a star and $T_{i-1} - T_i$ is a set of isolated vertices *adjacent to a common vertex* $v_i \in V(T_i)$ and v_i is a *leaf* of T_i for each $i = 1, \ldots, m$. Define $\omega_0 = \omega$ and for $i \geq 1$

$$\omega_i(v) = \begin{cases} \omega_{i-1}(v) & \text{if } v \neq v_i \\ \sum_{w \in V(T_{i-1} - T_i)} \omega_{i-1}(w) & \text{if } v = v_i \end{cases}$$

% The complexity of this step is $O(n)$

2. Let u_0 be the center of the star T_m and let $\ell = \sum_{v \in L(T)} \omega(v)$. Set

$$\begin{aligned} l(u_0, v) &= \omega_m(v), \text{ and} \\ l(v, u_0) &= \ell - \omega_m(v). \end{aligned}$$

% The complexity of this step is bounded by $3|V(T_m)|$.

3. For each $i < m$ let

$$\begin{aligned} l(u, v) &:= l(u, v) \quad \text{for all } u, v \notin V(T_i - T_{i+1}); \\ l(v_{i+1}, v) &:= \omega_i(v) \quad \text{for each } v \in V(T_i - T_{i+1}); \\ l(v, v_{i+1}) &:= l - \omega_i(v) \text{for each } v \in V(T_i - T_{i+1}). \end{aligned}$$

% The complexity of this step is bounded by $3 \sum_{i=0}^{m-1} |V(T_i) - V(T_{i+1})|$. So the total complexity is $O(n)$.

By Lemma (4.4), we may suppose T has at least $2k + 1$ leaves for the following algorithm to determine whether a given tree is a kBVT.

Algorithm 3.

Input: An integer $k \geq 1$ and a tree T with n vertices and ℓ leaves, where $\ell \geq 2k + 1$.

Output: "**True**" if T has a kBVT representation, and "**False**" otherwise.

1. Contract all vertices of T with degree 2.
% The complexity of this step is $O(n)$.

2. Find a vertex u such that $\ell - l_1(u) \geq k+1$.
% By Lemma (4.4), such a vertex does exist. By Algorithm 2, the complexity of this step is $O(n)$.

3. Initialize the spine: Let P be formed by the vertex u found in Step 2.
% Recall that any vertex u with the condition $\ell - l_1(u) \geq k+1$ must be on the spine if T is a kBVT.

4. If P contains a single vertex u and if $l(u,i) \geq k+1$ for some $i \in N(u)$, extend P to iu.
% Such a vertex i must be on the spine if T is a kBVT.

5. Suppose the current spine P is $p_1 p_2 \cdots p_t$. Define

$$N^0(p_1) = \begin{cases} N(p_1) & \text{if } t = 1, \\ N(p_1) \setminus p_2 & \text{if } t \geq 2. \end{cases}$$

% $N^0(p_1)$ contains all neighbors of p_1 outside of P.

- **Case 1:** $N^0(p_1) = \emptyset$. Then the extension at p_1 is complete.
- **Case 2:** $l(p_1, i) \geq k+1$ holds for at least two $i \in N^0(p_1)$. Then output **False**.
% T is not a generalized k-caterpillar and thus is not a kBVT.
- **Case 3:** $N^0(p_1) \neq \emptyset$ and $l(p_1, i) \geq k+1$ holds for at most one $i \in N^0(p_1)$. Choose $p_0 \in N^0(p_1)$ with $l(p_1, p_0) = \max\{l(p_1, i) : i \in N^0(p_1)\}$ and check if $\{l(p_1, i) : i \in N^0(p_1) \setminus p_0\}$ has a k-balanced partition. If no, output **False**; otherwise extend P to $p_0 p_1 p_2 \cdots p_t$. Return to Step 5.
% By Algorithm 1, the complexity of this case is $O\left(k \cdot |N^0(p_1)|\right) = O(k \cdot d(p_1))$.
% If $\{l(p_1, i); i \in N^0(p_1) \setminus p_0\}$ does not have a k-balanced partition, then by Lemma (4.3), no set of $\{l(p_1, i) : i \in N^0(p_1) \setminus i_0\}$ has a k-balanced partition for any $i_0 \in N^0(p_1)$, and thus T is not a kBVT. Now suppose $\{l(p_1, i); i \in N^0(p_1) \setminus p_0\}$ has a k-balanced partition. Note that p_0 is the only possible vertex i such that $i \in N^0(p_1)$ and $l(p_1, i) \geq k+1$. If $l(p_1, p_0) \geq k+1$, then p_0 must be on the spine and thus P is forced to be extended to p_0 from p_1. On the other hand, if $l(p_1, p_0) \leq k$, although the extension of P at p_1 might not be unique, we choose to extend P to p_0. This does not cause any future problem since there is no more need to extend P at p_0 when $l(p_1, p_0) \leq k$.

6. Similar to Step 5, extend P at the other endpoint till the extension is complete.

7. Output **True**

Theorem. (4.5) *One can determine with complexity $O(kn)$ whether a tree T with n vertices has a kBVT representation.*

Proof. If T has ℓ leaves with $\ell \le 2k$, by Lemma (4.4), T can be represented as a kBVT with complexity $O(n)$. Now suppose $\ell \ge 2k+1$ so that one can apply Algorithm 3 to T. Let $p_1 p_2 \cdots p_t$ be the spine determined by Algorithm 3. The complexity of Case 3 in Step 5 is $O(k \cdot d(p_i))$ when extending the spine at each vertex p_i, and the complexity of all other cases and steps altogether is $O(n)$. Thus the complexity of the algorithm is

$$\sum_{i=1}^{t} O(k \cdot d(p_i)) + O(n) = O\left(k \sum_{i=1}^{t} d(p_i)\right) + O(n) = O(kn).$$

□

Acknowledgement: The authors thank the anonymous referees for their suggestions, in particular, for the new proof of Claim 1 of Lemma 4.4.

References

[1] http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK4/NODE145.HTM

[2] N. Alon, G. Gutin, and M. Krivelevich, Algorithms with large domination ratio, *J. Algorithm*, **50**(1):118-131 (2004).

[3] P. Bose, A. Dean, J. Hutchinson, and T. Shermer, On rectangle visibility graphs: in *Lecture Notes in Computer Science #1190*, Springer-Verlag, Berline, 25-44, 1997.

[4] A. Dean, E. Gethner, and J. Hutchinson, Unit bar-visibility layouts of triangulated polygons: extended abstract, in *Lecture Notes in Computer Science #3383*: Graph Drawing 2004, J. Pach(ed.), Springer-Verlag, Berlin (2004), 111-121.

[5] A. Dean and J. Hutchinson, Rectangle-visibility representations of bipartite graphs, *Discrete Applied Math.* **75**:9-25 (1997).

[6] A. Dean and N. Veytsel, Unit bar-visibility graphs, *Congressus Numerantium* **160**:161-175 (2003).

[7] G. Gutin, A. Vainshtein, and A. Yeo, Domination analysis of combinatorial optimization problems, *Discrete Appl. Math.* **129**(2-3):513–520 (2003).

[8] S. Martello and P. Toth, Knapsack problems: algorithms and computer implementations, *Wiley Interscience Series in Discrete Mathematics and Optimization*, John Wiley & Sons, Ltd., Chichester, 1990.

[9] R. Tamassia and I. Tollis, A unified approach to visibility representations of planar graphs, *Discrete and Computational Geometry*, **1**:321-341 (1986).

[10] D. West, Introduction to graph theory, *Prentice Hall, Inc., Upper Saddle River, NJ,* 1996.

[11] S. Wismath, Characterizing bar line-of-sight graphs, *Proceedings of 1st ACM Symposium on Computational Geometry*, 147-152, Baltimore, MD, 1985.