Sum graphs from trees

M. N. Ellingham

Department of Mathematics
Vanderbilt University
Nashville, TN 37240, U. S. A.

Abstract. A graph G is a *sum graph* if there is a labelling σ of its vertices with distinct positive integers, so that for any two distinct vertices u and v, uv is an edge of G if and only if $\sigma(u) + \sigma(v) = \sigma(w)$ for some other vertex w. Every sum graph has at least one isolated vertex (the vertex with the largest label). Harary has conjectured that any tree can be made into a sum graph with the addition of a single isolated vertex. We prove this conjecture.

1. Introduction

All graphs in this paper are finite and have no loops or multiple edges. The idea of sum graphs was introduced by Harary [2]. A sum graph is a graph whose vertices can be given a labelling which determines the graph in a particularly simple way. A *sum labelling* of a graph G is a labelling σ of the vertices of G with distinct positive integers, so that for any two distinct vertices u and v, uv is an edge of G if and only if $\sigma(u) + \sigma(v) = \sigma(w)$ for some other vertex w. A sum labelling is *strong* if the above definition holds with "distinct vertices u and v" replaced by "possibly equal vertices u and v." If G has a (strong) sum labelling it is a (*strong*) *sum graph.* The quantity $\sigma(u) + \sigma(v)$ is called the *sum* of the edge uv.

Every sum graph must have at least one isolated vertex, the vertex with the largest label. Also, any graph with m edges can be made into a strong sum graph by the addition of at most m isolated vertices: label the original vertices with distinct powers of 3, and the m new isolated vertices with the m distinct sums corresponding to the edges. Thus, it is natural to ask for the minimum number of isolated vertices which must be added to a given graph G to form a sum graph. Let this number be denoted $s(G)$.

The trivial upper bound $s(G) \leq m$ can be improved substantially for "dense" graph by the following theorem.

Theorem 1.1 (Gould and Rödl [1]). *Let G be a graph with m vertices and n edges. Suppose that G has p vertex-disjoint complete subgraphs having $n_1, n_2, \ldots, n_p$ vertices, where $n_i \geq 3$ for each i, $1 \leq i \leq p$. Then*

$$s(G) \leq m + 2(n_1 + n_2 + \cdots + n_p) - 3p - \sum_{i=1}^{p} \binom{n_i}{2}.$$

The proof of this theorem uses the fact that $s(K_n) \leq 2n - 3$ for any complete graph K_n: label the vertices $1, 3, 5, \ldots, 2n - 1$. Gould and Rödl also have some lower bounds on $s(G)$.

For sparse graphs Theorem 1.1 seems not to provide a very good upper bound on $s(G)$; in fact for triangle-free graphs it just gives the trivial bound $s(G) \leq m$. The sparsest connected graphs are the trees, and Harary [2] has conjectured that $s(T) = 1$ for any tree T. The purpose of this paper is to prove this conjecture.

2. Caterpillars

To prove that $s(T) = 1$ for any tree $T \neq K_1$, we shall give an algorithm which finds a strong sum labelling of $T \cup \{z\}$, where z is an isolated vertex. The algorithm has two stages. Stage 1 is reasonably straightforward, and in some cases gives the complete sum labelling of $T \cup \{z\}$. However, in other cases Stage 1 gives only a partial labelling, and further action must be taken by Stage 2 to complete the labelling. In this section we lay the groundwork for Stage 1 by examining a special class of trees, the caterpillars.

When many people (including the author) first attempt to give a sum labelling σ to $T \cup \{z\}$, the following is what they suggest.

Naïve Algorithm.. *Label some vertex v_0 of T arbitrarily with a positive integer α. Choose a neighbour v_1 of v_0, and label v_1 with β, where $\beta > \alpha$ and $\beta \neq 2\alpha$ (so that $\sigma(v_0) + \sigma(v_0) \neq \sigma(v_1)$). Now, since $v_0 v_1$ is an edge, $\sigma(v_0) + \sigma(v_1)$ must appear as the label of some vertex. Choose an unlabelled vertex v_2 adjacent to an already labelled vertex u_2 ($u_2 = v_0$ or v_1), and label v_2 with $\sigma(v_0) + \sigma(v_1)$. Now, $\sigma(v_2) + \sigma(u_2)$ must appear on some vertex, so choose an unlabelled vertex v_3 adjacent to an already labelled vertex u_3, and label it with $\sigma(v_2) + \sigma(u_2)$. Continue in this fashion: at each step the vertex v_{i+1} receives the label $\sigma(v_i) + \sigma(u_i)$, where u_i is the only labelled neighbour of v_i. When the last vertex v_{n-1} of T is labelled, the sum $\sigma(v_{n-1}) + \sigma(u_{n-1})$ is allocated to the isolated vertex z.* ∎

The problem with this algorithm is that the labelling it produces may not, in general, be a sum labelling. It certainly ensures that if uv is an edge of G, then $\sigma(u) + \sigma(v) = \sigma(w)$ for some vertex w: in fact, every edge of T can be written as $u_i v_i$ for some i, $1 \leq i \leq n-1$ (letting $u_1 = v_0$), and the sum of this edge appears on $v_{i+1} - i + 1$ (letting $v_n = z$). However, the converse may not be true: there may be vertices u, v and w with $\sigma(u) + \sigma(v) = \sigma(w)$, where uv is not an edge of T. This is not allowed by our definition of a sum labelling. For example, the labelling in Figure 2.1 was produced by the Naïve Algorithm (with $\alpha = 1$ and $\beta = 3$), and has $\sigma(v_2) + \sigma(v_3) = 11 = \sigma(v_5)$, even though $v_2 v_3$ is not an edge of the tree. Therefore it is not a sum labelling.

While the Naïve Algorithm does not produce a sum labelling when applied indiscriminately, one may wonder whether it could produce a sum labelling if the order in which the vertices are labelled was carefully chosen. R.J. Gould (private communication) discovered that if T is a caterpillar and its vertices are labelled in

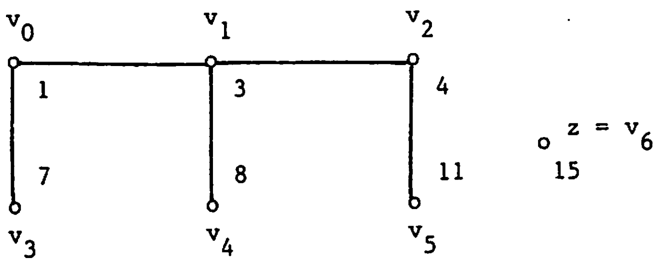


Figure 2.1: Non-sum labelling found by Naïve Algorithm

a particular order, then the Naïve Algorithm does give a sum labelling of $T \cup \{z\}$. His method forms the basis of our Stage 1.

For our purposes, a *caterpillar* C consists of a path $s_0 s_1 \ldots s_l$, called the *spine*, with leaves (degree 1 vertices) known as *feet* attached to the internal vertices of the spine by edges known as *legs*. We shall always think of the spine of a caterpillar as oriented in a particular direction, from the *tail* s_0 to the *head* s_l.

Caterpillar Algorithm (Gould).. *Suppose that each spine vertex s_i of the caterpillar C has feet t_{ij}, $1 \leq j \leq f_i$, adjacent to it. We have $f_i \geq 0$ for $1 \leq i \leq l-1$, and $f_0 = f_l = 0$ - the tail and head of a caterpillar are leaves. Use the Naïve Algorithm to label the vertices in the order:*

$$s_0, s_1, t_{11}, t_{12}, \ldots, t_{1f_1}, s_2, t_{21}, t_{22}, \ldots, t_{2f_2}, s_3, \ldots, s_l$$

(followed by z). ∎

In other words, beginning at the tail, label each spine vertex, followed by any feet adjacent to it, before moving to the next spine vertex. An example of a labelling produced by the Caterpillar Algorithm is given in Figure 2.2 (where $\alpha = 2$ and $\beta = 5$). As the Caterpillar Algorithm is a special case of the Stage 1 algorithm discussed in the next section, we shall not prove here that it actually gives a sum labelling. Notice that

$$\sigma(v_{i+1}) = \sigma(v_i) + \sigma(u_i),$$

where u_i is the unique labelled spine vertex adjacent to v_i at the time that v_i receives its label.

3. Stage 1

Stage 1 of our algorithm proceeds by trying to divide the tree T into caterpillars to which the Caterpillar Algorithm can be applied. The division of T into caterpillars must be done carefully to ensure that the resulting labelling is in fact a strong sum labelling. Some terminology is necessary.

First, if C is a caterpillar with spine $s_0 s_1 \ldots s_l$ the vertex s_{l-2} will be called the *heart* of C, and the vertex s_{l-1} will be called the *neck* of C.

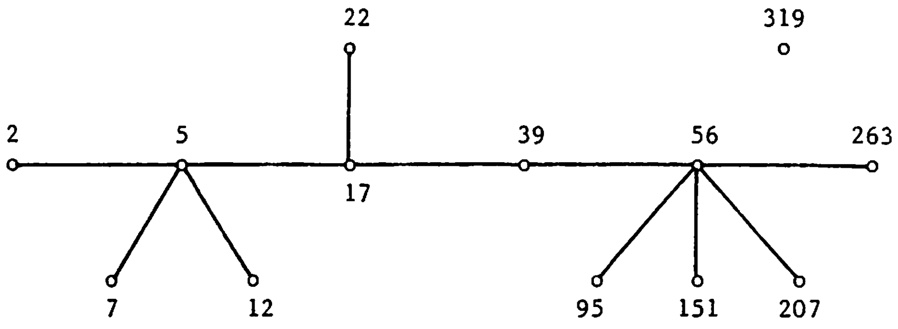


Figure 2.2: Sum labelling produced by Caterpillar Algorithm

Second, the vertices of T will be partitioned into three types: leaves, near-leaves and inner vertices. A leaf, as already mentioned, is a vertex of degree 1. A *near-leaf* is a non-leaf which has at most one non-leaf neighbour. An *inner vertex* is a vertex which has at least two non-leaf neighbours.

Now we state Stage 1 of our algorithm. Since it is easy to see that $s(K_1) = 0$ and $s(K_2) = 1$, we henceforth assume that T has at least three vertices.

Stage 1 Algorithm.. *Let T be a tree, and z an isolated vertex. To construct the first caterpillar C_1, choose any leaf whose neighbour is a near-leaf to be its tail s_0^1. Such a leaf always exists, since near-leaves occur in any tree other than K_1 or K_2. Construct the spine $S_1 = s_0^1 s_1^1 \dots s_{l_1}^1$ of C_1 as follows. Given $s_0^1 s_1^1 \dots s_j^1$, choose s_{j+1}^1 to be a currently unused vertex adjacent to s_j^1, preferring an inner vertex over a near-leaf and a near-leaf over a leaf. Stop when the path cannot be continued because the last vertex chosen, $s_{l_1}^1$, is a leaf. To form C_1, add to S_1 all leaves of T adjacent to internal vertices of S_1. All the vertices of C_1 are now to be considered 'used.'*

Given caterpillars $C_1, C_2, \dots, C_{i-1}$, we now construct C_i as follows. As the tail s_0^i of C_i choose any vertex used in $C_1, C_2, \dots, C_{i-1}$ which has an unused neighbour, with the restriction that the heart of C_{i-1} may not be chosen - in other words, $s_0^i \neq s_{l_{i-1}-2}^{i-1}$. To construct the spine S_i, grow a path by starting at s_0^i and adding unused inner vertices until the only possible vertices to add are near-leaves or leaves. Complete the path with an unused near-leaf $s_{l_i-1}^i$ and finally a leaf $s_{l_i}^i$. To obtain C_i from S_i, add any leaves of T adjacent to internal vertices of S_i. Note that any leaves of T adjacent to s_0^i do not form part of C_i - they are already part of some earlier caterpillar. Also, the leaves added to S_i to form C_i have not been used by any earlier caterpillar.

The process continues until we have formed caterpillars $C_1, C_2, \dots, C_k$, and either all the vertices of T have been used, or the only used vertex with an unused neighbour is the heart of C_k. Let $T' = C_1 \cup C_2 \cup \dots \cup C_k$ be the subtree of T induced by the used vertices. We label $T' \cup \{z\}$ by applying the Naïve Algorithm *to the vertices in the following order:*

$$s_0^1, s_1^1, \ldots (\textit{vertices of } C_1) \ldots, s_{l_1}^1, s_1^2, \ldots (\textit{vertices of } C_2) \ldots,$$
$$s_{l_2}^2, \ldots, s_1^k, \ldots (\textit{vertices of } C_k) \ldots, s_{l_k}^k$$

(followed by z), where the vertices in each C_i are ordered as in the Caterpillar Algorithm. Note that for $i \geq 2$, the tail s_0^i of C_i is not labelled with the vertices of C_i becaue it is equal to some s_p^j with $j < i$, and has already been labelled with the vertices of C_j. ∎

Thus, T is (completely or partially) decomposed into caterpillars, which are labelled in succession using Gould's Caterpillar Algorithm from Section 2. An example of a Stage 1 labelling appears as part of the example of a complete strong sum labelling in Figure 5.1. Notice the following four properties of Stage 1.

(A) Using the notation of the Naïve Algorithm, we have that if v_j is labelled as part of C_i, then
$$\sigma(v_{j+1}) = \sigma(v_j) + \sigma(u_j),$$
where u_j is the unique spine vertex of C_i which is both adjacent to v_j and already labelled when v_j is to be labelled.

(B) As in any application of the Naïve Algorithm, we have
$$\sigma(v_i) < \sigma(v_j) \text{ when } i < j.$$

(C) In each individual caterpillar C, with spine length l, if spine vertex s_i has f_i adjacent feet t_{ij} then
$$\sigma(t_{ij}) = \sigma(s_{i-1}) + j\sigma(s_i), \quad 1 \leq j \leq f_i,$$
$$\sigma(s_{i+1}) = \sigma(s_{i-1}) + (f_i + 1)\sigma(s_i), \quad 1 \leq i \leq l-1.$$

(D) For any $i \geq 2$, the tail s_0^i of C_i is an inner vertex used in the spine of some C_j, where $j \leq i-1$. It cannot be a leaf or near-leaf, since then all of its neighbours would have been used by C_j, eliminating it as a possible choice for the tail of C_i. Thus, s_0^i cannot be a vertex of C_{i-1} labelled after the heart of C_{i-1}, for all such vertices are leaves or near-leaves. It also cannot be the heart of C_{i-1} itself, because this is explicitly forbidden. Therefore, the tail of C_i has a label smaller than the heart of C_{i-1}.

We must now prove that the Stage 1 Algorithm gives a strong sum labelling of $T' \cup \{z\}$. Since it is an implementation of the Naïve Algorithm, we know that if uv is an edge of T' then $\sigma(u) + \sigma(v) = \sigma(w)$ for some w. So we just need to show that if $\sigma(u) + \sigma(v) = \sigma(w)$, then uv is an edge of T'. We may assume without loss of generality that $\sigma(u) \leq \sigma(v)$, which implies that $\sigma(v) \geq \sigma(w)/2$.

If $w = v_0$ then no u and v exist with $\sigma(u) + \sigma(v) = \sigma(w)$. If $w = v_i$ then the only possibility is that $u = v = v_0$, but by the requirement that $\beta \neq 2\alpha$, in the initialization of the Naïve Algorithm, $\sigma(v_0) + \sigma(v_0) \neq \sigma(v_1)$. It is also clear that if $w = v_2$ then $u = v_0$ and $v = v_1$, and thus uv is an edge. Therefore, we may now assume that $w = v_{j+1}$, where $j \geq 2$. Hence, by (A) above,

$$\sigma(w) = \sigma(v_{j+1}) = \sigma(v_j) + \sigma(u_j),$$

where, supposing v_j was labelled as part of C_i, u_j is a spine vertex of C_i which is adjacent to v_j and which was labelled before v_j. To prove that uv is an edge it is therefore sufficient to show either that $u = u_j$, or that $v = v_j$.

There are two cases to examine, depending on whether u_j is the tail of C_i or not. First suppose that u_j is not the tail of C_i. Then $u_j = s^i_k$ with $k \geq 1$. Let $\gamma = \sigma(s^i_{k-1})$ and $\delta = \sigma(s^i_k)$, so that by (B) above $\gamma < \delta$. Now v_j must be s^i_{k+1} or some foot t^i_{kp} of C_i adjacent to s^i_k. In either case, by (C) above, $\sigma(v_j) = \sigma(s^i_{k-1}) + p\sigma(s^i_k)$ for some $p \geq 1$. Thus,

$$\begin{aligned}\sigma(u_j) &= \delta,\\ \sigma(v_j) &= \gamma + p\delta,\\ \sigma(w) &= \sigma(v_j) + \sigma(u_j) = \gamma + (p+1)\delta.\end{aligned}$$

Therefore, $\sigma(w) > 2\delta$ and $\sigma(v) \geq \sigma(w)/2 > \delta$, which by (B) above implies that v was labelled after s^i_k. Of course, v was also labelled before $w = v_{j+1}$. Thus, either $v = v_j$, in which case we are finished, or else $v = t^i_{kq}$ for some $q < p$. Assuming the latter,

$$\begin{aligned}\sigma(v) &= \gamma + q\delta \quad \text{by (C)},\\ \sigma(u) &= \sigma(w) - \sigma(v) = (p + 1 - q)\delta.\end{aligned}$$

Since $p > q$, $\sigma(u) > \delta$ and by the same reasoning as for v, $u = t^i_{kr}$ for some $r < p$, and

$$\sigma(u) = \gamma + r\delta \quad \text{by (C)}.$$

Equating the two values of $\sigma(u)$ gives $\gamma \equiv 0 \pmod{\delta}$ which is impossible since $0 < \gamma < \delta$.

Second, suppose that u_j is the tail of C_i. Since $j \neq 1$, we may assume that $i \geq 2$. Let $l = l_{i-1}$ be the length of the spine of C_{i-1}. Then we must have

$$v_j = s^i_1,\ u_j = s^i_0,\ v_{j-1} = s^{i-1}_l,\ u_{j-1} = s^{i-1}_{l-1}.$$

Let $\gamma = \sigma(s^{i-1}_{l-2})$, $\delta = \sigma(s^{i-1}_{l-1})$ and $\epsilon = \sigma(u_j)$. By (D) we know that $\epsilon < \gamma$ and by (B) $\gamma < \delta$. Let $p = f^{i-1}_{l-1} + 1$ be the number of leaves (including s^{i-1}_l) adjacent to s_{l-1}^{i-1}; then $p \geq 1$. We now have

$$\begin{aligned} \sigma(u_{j-1}) &= \delta, \\ \sigma(v_{j-1}) &= \gamma + p\delta \quad \text{by (C)}, \\ \sigma(u_j) &= \epsilon, \\ \sigma(v_j) &= \sigma(u_{j-1}) + \sigma(v_{j-1}) = \gamma + (p+1)\delta, \\ \sigma(w) &= \sigma(u_j) + \sigma(v_j) = \epsilon + \gamma + (p+1)\delta. \end{aligned}$$

Since $p \geq 1$, $\sigma(w) > 2\delta$ and thus $\sigma(v) \geq \sigma(w)/2 > \delta$. Hence, v was labelled after s_{l-1}^{i-1} indicating that either $v = v_j$, in which case we are finished, or v is a leaf of C_{i-1} adjacent to s_{l-1}^{i-1}. Assuming the latter,

$$\begin{aligned} \sigma(v) &= \gamma + q\delta \quad \text{for some } q \text{ with } 1 \leq q \leq p, \\ \sigma(u) &= \sigma(w) - \sigma(v) = \epsilon + (p + 1 - q)\delta. \end{aligned}$$

Therefore $\sigma(u) > \delta$, and by the same reasoning as for v, u is a leaf adjacent to s_{l-1}^{i-1} and

$$\sigma(u) = \gamma + r\delta \quad \text{for some } r \text{ with } 1 \leq r \leq p.$$

Equating the two values of $\sigma(u)$, we now see that $\epsilon \equiv \gamma \pmod{\delta}$, which is impossible since $0 < \epsilon < \gamma < \delta$.

Notice that our argument in this last case relies on the fact that $\epsilon < \gamma$, which follows from the restriction in the algorithm that the tail of C_i not be the heart of C_{i-1}. This is the reason for that restriction.

This completes the proof that Stage 1 does give a strong sum labelling of $T' \cup \{z\}$. Our task now is to complete that labelling when $T' \neq T$.

4. Shrubs

Just as Stage 1 of our algorithm depends on an algorithm for labelling caterpillars, so Stage 2 depends on an algorithm for labelling another special class of trees. A *shrub* is a tree with at most one inner vertex. We shall think of a shrub as a tree with a distinguished vertex, called the *root*, all of whose neighbour are leaves or near-leaves. If a shrub has an inner vertex, then the root must be this vertex.

Our approach to labelling shrubs is different in nature from the approach to labelling caterpillars, which employed the Naïve Algorithm. The Naïve Algorithm assigns a unique sum to each edge, and every vertex except the two with the lowest labels has a label which is the sum of some edge. In contrast, the labelling procedure for shrubs assigns nonunique sums to some edges, and there are some vertices whose label is not the sum of any edge.

Our labelling for shrubs departs from the strict definition of a sum labelling, in that some of the labels are not integers, but rational numbers. However, any rational sum labelling can be transformed into an integral one by multiplying all labels by a suitable constant integer. The reason for using rational numbers is merely to make it easier to combine the Stage 1 labelling with the labelling of a shrub in Stage 2.

The labelling procedure for shrubs is as follows.

Shrub Algorithm.. *Suppose S is a shrub with root r, and z is an isolated vertex. Let the degree 1 neighbours of r be $a_1, a_2, \ldots, a_m$, the degree 2 neighbours $b_1, b_2, \ldots, b_n$ and the remaining neighbours $c_1, c_2, \ldots, c_p$, where c_i has degree $d_i \geq 3$ in S. Each b_i has a leaf neighbour $x_i \neq r$, and each c_i has leaf neighbours $y_{ij} \neq r$, where $1 \leq j \leq d_i - 1$. Define $D_1 = 1$ and $D_i = d_1 d_2 \ldots d_{i-1}$ for $i = 2, \ldots, p+1$.*

Choose positive integers θ, λ with $\lambda > 2(m+1)\theta$. Label as follows:

$$\begin{aligned}\sigma(r) &= \theta,\\ \sigma(c_i) &= D_i\lambda,\\ \sigma(y_{ij}) &= jD_i\lambda + \theta.\end{aligned}$$

If $n = 0$ let

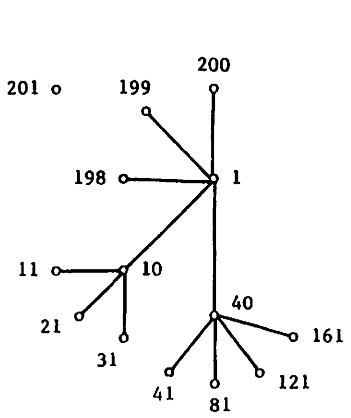
$$\begin{aligned}\sigma(a_i) &= D_{p+1}\lambda + (1-i)\theta,\\ \sigma(z) &= D_{p+1}\lambda + \theta,\end{aligned}$$

and if $n \neq 0$ choose an integer $\eta \geq n$ and let

$$\begin{aligned}\sigma(a_i) &= D_{p+1}\lambda - i\theta,\\ \sigma(b_i) &= D_{p+1}\lambda + (i-1)/\eta,\\ \sigma(x_i) &= D_{p+1}\lambda + \theta + (n-i)/\eta,\\ \sigma(z) &= 2D_{p+1}\lambda + \theta + (n-1)/\eta.\end{aligned}$$

∎

Examples of labellings produced by this algorithm are shown in Figure 4.1. To prove that this labelling produces a strong sum labelling, we must first show that the sum of every edge appears on some vertex. All edges in S have one of the



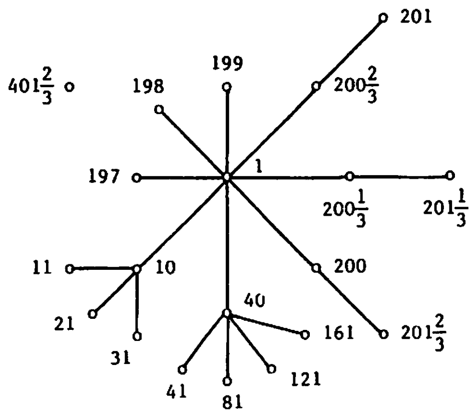


Figure 4.1: Strong sum labellings produced by Shrub Algorithm

following forms: ra_i, rb_i, b_ix_i, rc_i, or c_iy_{ij}. It is not difficult to verify that:

$$\sigma(r)+\sigma(a_i)=\begin{cases}\sigma(a_{i-1}) & \text{if } i\geq 2,\\ \sigma(z) & \text{if } i=1 \text{ and } n=0,\\ \sigma(b_1) & \text{if } i=1 \text{ and } n>0,\end{cases}$$

$$\sigma(r)+\sigma(b_i)=\sigma(x_{n+1-i});$$

$$\sigma(b_i)+\sigma(x_i)=\sigma(z)\ \ (\text{when } n>0);$$

$$\sigma(r)+\sigma(c_i)=\sigma(y_{i1});$$

$$\sigma(c_i)+\sigma(y_{ij})=\begin{cases}\sigma(y_{i,j+1}) & \text{if } j\leq d_i-2,\\ \sigma(y_{i+1,1}) & \text{if } i\leq p-1 \text{ and } j=d_i-1,\\ \sigma(z) & \text{if } i=p, j=d_p-1 \text{ and } n=0,\\ \sigma(x_n) & \text{if } i=p, j=d_p-1 \text{ and } n>0.\end{cases}$$

Thus, the sum of every edge appears as the label on some vertex.

We must also show that if $\sigma(u)+\sigma(v)=\sigma(w)$, then uv is an edge of S. There are two cases to examine, namely $n>0$ and $n=0$.

(1) **n** >0. Let $A=\{a_1,\ldots,a_m\}$, $B=\{b_1,\ldots,b_n\}$, $X=\{x_1,\ldots,x_n\}$, $C=\{c_1,\ldots,c_p\}$ and $Y=\{y_{11},\ldots,y_{p,d_p-1}\}$. For any integer μ, let $I(\mu)$ denote the set of vertices with integral label congruent to $\mu \pmod{\lambda}$. Consider the following subcases.

(1.1) **u** $\in$ **B** $\cup$ **X**. Then $\sigma(w)>\sigma(u)\geq D_{p+1}\lambda$, and so $w\in B\cup X\cup\{z\}$. Suppose first that $w=z$. From $\sigma(v)=\sigma(w)-\sigma(u)$ we find that if $u=b_i$ then $v=x_i$, and vice versa; therefore uv is an edge. Now suppose that $w\in X$. Since $\sigma(v)=\sigma(w)-\sigma(u)$, we have that $-1<\sigma(v)<1$ if $u\in X$, and $\theta-1<\sigma(v)<\theta+1$ if $u\in B$.

The only possibility is that $u \in B$, $\sigma(v) = \theta$ and $v = r$, implying that uv is an edge. Finally, suppose that $w \in B$; then $u \in B$ also. From $\sigma(v) = \sigma(w) - \sigma(u)$, we get that $-1 < \sigma(v) < 1$, which is impossible.

(1.2) **u = r**. By (1.1)we may assume that $v \notin B \cup X$, so that $v \in \{r\} \cup A \cup C \cup Y$. If $v \in A \cup C$ then uv is an edge. If $v \in \{r\} \cup Y$ then u, $v \in I(\theta)$, implying that $w \in I(2\theta)$. However, all the integral labels are congruent to one of $0, \theta, \lambda - \theta, \lambda - 2\theta, \ldots, \lambda - m\theta \pmod{\lambda}$, and none of these is equivalent to $2\theta \pmod{\lambda}$ because $\lambda > 2(m+1)\theta$.

(1.3) **u ∈ A**. By (1.1) and (1.2) we may assume that $v \in A \cup C \cup Y$. If $v \in A$ then $\sigma(v) \geq \sigma(a_m) = D_{p+1}\lambda - m\theta$, and if $v \in C \cup Y$, $\sigma(v) \geq \lambda$, in either case $\sigma(v) \geq \lambda - m\theta$. Therefore,

$$D_{p+1}\lambda - m\theta = \sigma(a_m) \leq \sigma(u) \leq \sigma(a_1) = D_{p+1}\lambda - \theta,$$
$$\lambda - m\theta \leq \sigma(v) < \sigma(b_1) = D_{p+1}\lambda.$$

Since $\sigma(w) = \sigma(u) + \sigma(v)$, and since $\lambda > 2(m+1)\theta$,

$$\sigma(x_1) = D_{p+1}\lambda + \theta + (n-1)/\eta < D_{p+1}\lambda + \lambda - 2m\theta$$
$$\leq \sigma(w) < 2D_{p+1}\lambda - \theta < \sigma(z).$$

But there are no vertices with label between $\sigma(x_1)$ and $\sigma(z)$.

(1.4) **u ∈ C**. By (1.1)-(1.3) we may assume that $v \in C \cup Y$. Suppose first that $v \in C$; without loss of generality we may assume that $\sigma(u) \leq \sigma(v)$. Then $w \in I(0)$ and $\sigma(v) < \sigma(w) \leq 2\sigma(v)$. However, for any $v = c_j \in C \subseteq I(0)$, the vertex with next smallest label in $I(0)$ has label $d_j\sigma(v) \geq 3\sigma(v)$, and so this is impossible.
Suppose therefore that $v \in Y$; write $u = c_i$ and $v = y_{jk}$. Then $\sigma(w) = D_i\lambda + kD_j\lambda + \theta$ and $w \in I(\theta)$. For any $y \in Y \subseteq I(\theta)$, let y^* denote the vertex with next smallest label in $I(\theta)$. If $i > j$ then $\sigma(y_{i1}) < \sigma(w) < \sigma(y_{i2})$, which cannot happen because $y_{i2} = y_{i1}^*$. If $j > i$ then $\sigma(y_{jk}) < \sigma(w) < \sigma(y_{jk}^*)$, which again cannot happen (y_{jk}^* is $y_{j,k+1}$, $y_{j+1,1}$ or x_n depending on j and k, but in all three cases $\sigma(y_{jk}^*) = (k+1)D_j\lambda + \theta$). Thus, $i = j$ and $uv = c_iy_{ik}$ is an edge.

(1.5) **u ∈ Y**. By (1.1)-(1.4) we may assume that $v \in Y$ also. But then u and v are both in $I(\theta)$, implying that $w \in I(2\theta)$, which is impossible, as we saw in (1.2).

(2) **n = 0**. This follows by a slightly modified and simplified version of the argument for (1). The relationship between the two cases is illustrated by Figure 4.1.

This completes the proof that the Shrub Algorithm gives a strong sum labelling of $S \cup \{x\}$. In the next section we use this to complete our strong sum labelling of $T \cup \{z\}$.

Notice that if we require only a sum labelling, not a strong sum labelling, then we can simplify the above procedure by labelling the degree 2 neighbours of r in the same way as the neighbours of degree at least 3: in other words, in the above algorithm we include the neigbours of degree 2 in the vertices $c_1, \ldots, c_p$ and do not have any vertices $b_1, \ldots, b_n$ The resulting labelling may not be strong because if c_i has degree 2 then we get $2\sigma(c_1) = \sigma(c_{i+1})$. However, all of the labels are integers; thus, we can obtain an integral labelling in which the smallest label is any given positive integer θ, including 1.

5. Stage 2

In this section we show how the Shrub Algorithm of Section 4 can be used to complete a strong sum labelling produced by the Stage 1 Algorithm. As this algorithm uses the Shrub Algorithm from Section 4, it can produce labels which are nonintegral rational numbers; however, as in Section 4, the labelling can easily be converted to an integral labelling if desired.

Stage 2 Algorithm.. *Suppose that T is a tree and z is an isolated vertex. Assume that the Stage 1 Algorithm of Section 3 has terminated without labelling all vertices of T, after producing caterpillars $C_1, C_2, \ldots, C_k$. Let $l = l_k$ denote the length of the spine of C_k. The only used vertex with unused neighbours is the heart of C_k, s^k_{l-2}. All the unused neighbours of s^k_{l-2} must be near-leaves: the leaf neighbours are used by some caterpillar C_j, where $j \le k-1$, and if there were an unused inner vertex adjacent to s^k_{l-2}, it would have been chosen rather than the near-leaf s^k_{l-1} when constructing the spine of C_k.*

There are various cases depending on the values of k and l.

(a) $k = 1, l \le 4$. Then T is a shrub whose root can be taken to be $r = s^k_{l-2}$. Use the Shrub Algorithm to give a sum labelling to $T \cup \{z\}$, replacing any labels already assigned by Stage 1.

(b) $k = 1, l \ge 5$. Let C_1' be the caterpillar obtained from C_1 by reversing its spine. Since $l \ge 5$, s^1_{l-2} is not the heart of C_1', and therefore we can return to the caterpillar construction phase of Stage 1, after discarding the labelling of C_1.

(c) $k \ge 2, l \ge 3$. In this case the tail s^k_0 of C_k is a spine vertex of some earlier caterpillar C_j, say $s^k_0 = s^j_i$. Define two new caterpillars C_j' and C_j'' as follows: C_j' has spine $S_j' = s^j_0 s^j_1 \ldots s^j_{i-1}(s^j_i = s^k_0)s^k_1 \ldots s^k_l$, and C_j'' has spine $S_j'' = s^j_i s^j_{i+1} \ldots s^j_{l_j}$; each has as feet the leaves of T adjacent to the internal vertices of the spine. Then the tail of C_j' is not the heart of C_{j-1}, the tail of C_j'' is not the heart of C_j' because $l \ge 3$, and the tail of C_{j+1} is not the heart of C_j''. Therefore, Stage 1 can be used to construct the following sequence of caterpillars:

$$C_1, C_2, \ldots, C_{j-1}, C_j', C_j'', C_{j+1}, \ldots, C_{k-1}.$$

The union of this sequence of caterpillars is the same subtree T' as before. The vertex s^k_{l-2} is still the only used vertex with unused neighbours, but is now not the heart of the last caterpillar in the sequence. Therefore we can return to Stage 1 and continue constructing caterpillars, after discarding the original Stage 1 labelling.

(d) $k \geq 2, l \leq 2$. Since C_k starts at an inner vertex of some earlier C_j (see (D) of Section 3), we must have $l = 2$. Let $r = s^k_0$ be the tail of C_k. Let S be the subtree of T induced by $V(C_k) \cup (V(T) - V(T'))$. Then the neighbours of r in S are s^k_1 and the unused neighbours of r, all of which are near-leaves. Therefore S is a shrub with root r. Let $\theta = \sigma(s^k_0)$ and $\lambda = \sigma(s^k_1)$, and label the vertices of $S \cup \{z\}$ using the Shrub Algorithm, replacing any labels already present from Stage 1.

∎

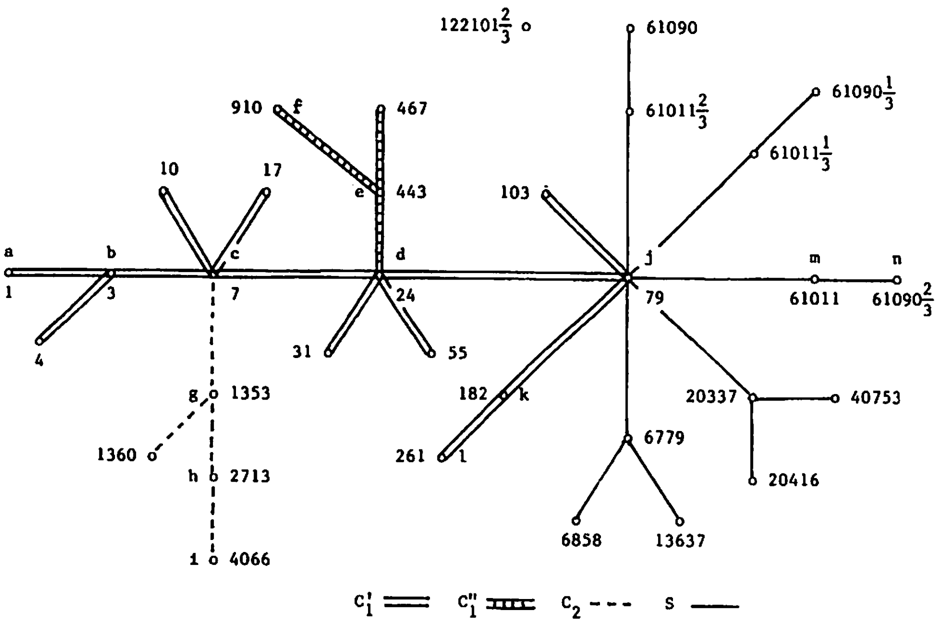


Figure 5.1: Strong sum labelling produced by Stages 1 and 2

Figure 5.1 shows an example of a strong sum labelling produced by the complete algorithm (Stages 1 and 2). The original Stage 1 labelling had caterpillars with spines $S_1 = abcdef$, $S_2 = cghi$ and $S_3 = djkl$. After applying (c) of Stage 2, the sequence of caterpillars C'_1, C''_1, C_2 as shown was formed, and a new caterpillar C^*_3 with spine jmn was constructed. On returning to (d) of Stage 2, C^*_3 became part of S.

We must now prove that the complete algorithm (Stages 1 and 2) produces a strong sum labelling of $T \cup \{z\}$. If we never use Stage 2, this follows from Section 3. If we arrive at (a) of Stage 2, then it follows from Section 4. If we reach (b) or (c), then we return to Stage 1 and construct exactly one more caterpillar, with tail s^k_{l-2} (since the unused part of T forms a shrub rooted at s^k_{l-2}). Now we either have a complete Stage 1 labelling of $T \cup \{z\}$, or we must apply (d) of Stage 2. Step (d) can also be reached without applying (b) or (c), and we henceforth asume that the algorithm has terminated in (d).

To prove that the labelling produced in (d) is a strong sum labelling, we need some notation. The sum labelling of $T' \cup \{z\}$ after we leave Stage 1 for the last time will be denoted σ_1. The final labelling of $T \cup \{z\}$ after the algorithm terminates will be denoted σ_2. The vertices of the subtree $T'' = C_1 \cup \cdots \cup C_{k-1}$ do not have their labels changed in any way by Stage 2, so that $\sigma_2(v) = \sigma_1(v)$ for all $v \in V(T'')$. Notice that $E(T) = E(T'') \cup E(S)$, $V(T) = V(T'') \cup V(S)$, and $V(T'') \cap V(S) = \{r\}$. Let $q = l_{k-1}$ denote the length of the spine of C_{k-1}.

Notice the following properties of (d).

(A) Since the root r of S is used in some C_j, where $j \leq k-1$, the leaves adjacent to r are used by C_j and are not part of S. In the notation of Section 4, this means that $m = 0$.

(B) By (D) and (B) of Section 3, $\theta = \sigma_1(s^k_0) < \sigma_1(s^{k-1}_{q-2}) < \sigma_1(s^{k-1}_{q-1})$. Thus, for any vertex v of T'',
$$\sigma_2(v) = \sigma_1(v) \leq \sigma_1(s^{k-1}_q) = \sigma_1(s^k_1) - \sigma_1(s^{k-1}_{q-1}) < \lambda - \theta.$$

(C) Letting $v = r = s^k_0$ in (B), we see that $\theta = \sigma_1(s^k_0) < \lambda - \theta$, or $\lambda > 2\theta$. Since $m = 0$, this implies that $\lambda > 2(m+1)\theta$. Therefore, from Section 4, the labelling of $S \cup \{z\}$ in (d) is a strong sum labelling. In other words, the restriction of σ_2 to $S \cup \{z\}$ is a strong sum labelling.

(D) Since $m = 0$, the four smallest integral labels on vertices of $S \cup \{z\}$ are θ, λ, $\lambda + \theta$, and $2\lambda + \theta$. If $p > 0$ these occur on r, c_1, y_{11} and y_{12}; if $p = 0$ these occur on r, b_1, x_n and z. Notice also that all nonintegral labels are greater than λ.

To prove that σ_2 is a strong sum labelling, we must first show that the sum of any edge appears as the label of some vertex. For any edge of S, the sum appears on a vertex of $S \cup \{z\}$, by (C) above. For any edge of T'', its sum appears on another vertex of T'', by Section 3, except for the edge $s^{k-1}_{q-1}s^{k-1}_q$ joining the last two spine vertices of C_{k-1}. The sum of this edge, from Section 3, is $\sigma(s^k_1) = \lambda$, which from (D) above appears on a vertex of S.

Finally, we must show that if $\sigma_2(u) + \sigma_2(v) = \sigma_2(w)$, then uv is an edge of T. There are three cases.

(1) $\mathbf{u}, \mathbf{v} \in \mathbf{V}(\mathbf{T}'')$. By (B), $\sigma_2(u) + \sigma_2(v) < 2\lambda - 2\theta$. Thus, by (D), $w \in V(T'')$, $\sigma_2(w) = \lambda$, or $\sigma_2(w) = \lambda + \theta$. Now $\lambda = \sigma_1(s^k_1)$ and $\lambda + \theta$ is equal

to either $\sigma_1(t_{11}^k)$ or $\sigma_1(s_2^k)$. Therefore, in any of the three cases there is some $w' \in V(T')$ with $\sigma_2(w) = \sigma_1(w')$. Then $\sigma_1(u) + \sigma_1(v) = \sigma_1(w')$, and since σ_1 is a sum labelling, uv is an edge of T' and hence an edge of T.

(2) $\mathbf{u}, \mathbf{v} \in \mathbf{V(S)} - \{\mathbf{r}\}$. Then by (B) and (D), $w \in V(S) \cup \{z\}$ and uv is an edge by (C).

(3) $\mathbf{u} \in \mathbf{V(S)} - \{\mathbf{r}\}, \mathbf{v} \in \mathbf{V(T'')}$. Then by (B) and (D), $w \in V(S) \cup \{z\}$. If $\sigma_2(u)$ is not integral then $u \in B \cup X$, and the arguments of Case (1.1) in Section 4 show that $v = r$ and $u \in B$, proving that uv is an edge. If $\sigma_2(u)$ is an integer, then $m = 0$ implies that $u,w \in I(0) \cup I(\theta)$, and hence $\sigma_2(v) = \sigma_2(w) - \sigma_2(v)$ is congruent to 0, θ or $\lambda - \theta \pmod{\lambda}$. By (B), $0 < \sigma_2(v) < \lambda - \theta$, so the only possibility is that $\sigma_2(v) = \theta$ and $v = r$. But then $u, v, w \in V(S) \cup \{z\}$ and uv is an edge by (C).

This concludes the proof that the algorithm gives a strong sum labelling of $T \cup \{z\}$. Thus, we have our main result.

Theorem 5.1. *If T is any tree other than K_1, then $T \cup \{z\}$ has a strong sum labelling, where z is an isolated vertex. Consequently, $s(T) = 1$.*

As F. Harary has pointed out, this result also gives the value of $s(F)$ for any forest F. We have $s(F) = 0$ when F has an isolated vertex, and $s(F) = 1$ otherwise.

If we require only a sum labelling, and not a strong sum labelling, then by modifying Stage 2 in the way described at the end of Section 4, we do not have to introduce nonintegral labels. Thus, we obtain the following result.

Theorem 5.2. *If T is any tree other than K_1, and α is any positive integer, then $T \cup \{z\}$ has a sum labelling with smallest label α, where z is an isolated vertex.*

6. Final comments

It would be nice to obtain bounds on $s(G)$ for other classes of sparse graphs. One now-disproved conjecture stated that for any connected graph G with n vertices and m edges, $s(G) \leq m - n + 2$. A counterexample is provided by the 4-cycle C_4, but it seems that this conjecture may actually hold for some restricted classes of graphs: this paper verifies that it is true for trees. One possible approach to extend this result to other graphs is to try to use a spanning tree T of a graph G. Assuming G has n vertices and m edges, one would take isolated vertices $z_1, \ldots, z_{m-n+2}$, label $T \cup \{z_1\}$, and then label $z_2, \ldots, z_{m-n+2}$ with the sums of edges of G not in T. However, this approach has the same problem as the Naïve Algorithm of Section 2: it can create situations where u and v are nonadjacent, but $\sigma(u) + \sigma(v)$ is the label of some vertex. It seems to be very difficult to choose T and the labelling of $T \cup \{z_1\}$ to avoid this problem.

Acknowledgements

The author would like to thank Ron Gould for introducing him to this problem, and Gerard J. Chang and Frank Harary for their helpful comments.

References

1. R. J. Gould and V. Rödl, *Bounds on the number of isolated vertices in sum graphs,*. to appear in *Graph Theory, Combinatorics and Applications*, Proceedings of the Sixth Quadrennial International Conference on the Theory and Applications of Graphs (Kalamazoo, Mich., 1988), ed. Y. Alavi et al. (1991).
2. F. Harary, *Sum graphs and difference graphs*, Proceedings of the Twentieth Southeastern International Conference on Combinatorics, Graph Theory and Computing, Boca Raton, 1989, *Congressus Numerantium* 72 (1990), 101–108.