The Unit Acquisition Number of a Graph

Frederick Johnson[*], Anna Raleigh[*],
Paul S. Wenger[*], and Douglas B. West[†]

November 5, 2018

Abstract

Let G be a graph with nonnegative integer weights. A *unit acquisition move* transfers one unit of weight from a vertex to a neighbor that has at least as much weight. The *unit acquisition number* of a graph G, denoted $a_u(G)$, is the minimum size that the set of vertices with positive weight can be reduced to via successive unit acquisition moves when starting from the configuration in which every vertex has weight 1.

For a graph G with n vertices and minimum degree k, we prove $a_u(G) \le (n-1)/k$, with equality for complete graphs and C_5. Also $a_u(G)$ is at most the minimum size of a maximal matching in G, with equality on an infinite family of graphs. Furthermore, $a_u(G)$ is bounded by the maximum degree and by $\sqrt{n-1}$ when G is an n-vertex tree with diameter at most 4. We also construct arbitrarily large trees with maximum degree 5 having unit acquisition number 1, obtain a linear-time algorithm to compute the unit acquisition number of a caterpillar, and show that $a_u(G) = 1$ when G has diameter 2, except that the value is 2 for C_5 and the Petersen graph.

Keywords: 05C22; acquisition number; unit acquisition; caterpillar; diameter 2, Petersen graph

1 Introduction

Consider a network of cities in which troops have been deployed. If the goal is to airlift the troops out of the cities, then a possible protocol for moving the troops is to first consolidate troops by having those in a city with many troops protect a smaller number arriving from a neighboring city. The goal is to minimize the number of cities where an airlift is then needed to remove the troops. In this setting, can all of the troops move to a single city?

[*]School of Mathematical Sciences, Rochester Institute of Technology, Rochester, NY; fwj9028@rit.edu, anr2291@rit.edu, pswsma@rit.edu.

[†]Departments of Mathematics, Zhejiang Normal University, China, and University of Illinois, USA , west@math.uiuc.edu. Research supported by National Natural Science Foundation of China grant NSFC-11871439.

We model this scenario with a graph in which every vertex initially has weight 1. A *unit acquisition move* transfers one unit of weight from a vertex u to a neighbor v, under the condition that before the move v has at least as much weight at u. The *unit acquisition number* of a graph G, denoted $a_u(G)$, is the minimum size that the set of vertices with positive weight can be reduced to via successive unit acquisition moves when starting from the configuration in which every vertex has weight 1. Because a legal move from u to v can be followed by another if u has additional weight, we can equivalently transfer any integer amount of weight when a transfer is allowed.

Acquisition numbers were introduced by Lampert and Slater [5]. They referred to unit acquisition moves (and the more general integer transfers) as *consolidations* and called a consolidation that transfers all the weight from a vertex an *acquisition move*. To unify the terminology across several models, we call such a move a *total acquisition move*; thus "unit" indicates that not all of the weight need move. Another model allows consolidations that move fractional (non-integer) amounts of weight. In each case the initial distribution has weight 1 at each vertex, and the aim is to make the set retaining positive weight as small as possible using the specified type of consolidation. In addition to the unit acquisition number $a_u(G)$, the corresponding parameters are the *total acquisition number* $a_t(G)$ and the *fractional acquisition number* $a_f(G)$.

Up to now, the study of acquisition has focused on total acquisition and fractional acquisition. Lampert and Slater [5] proved $a_t(G) \le \lfloor (n+1)/3 \rfloor$ and observed that no vertex v can reach weight greater than $2^{d(v)}$, where $d(v)$ is the degree of v. Slater and Wang [11] later proved that deciding whether a graph has total acquisition number 1 is NP-complete, and they gave a linear-time algorithm to determine $a_t(G)$ when G is a caterpillar. Among other results, LeSaulnier et al. [6] characterized the trees G such that $a_t(G) = 1$, obtained sharp bounds on $a_t(G)$ in terms of the diameter and number of vertices when G is a tree, gave bounds on $a_t(G)$ when G has diameter 2, and proved $\min\{a_t(G), a_t(\overline{G})\} = 1$, where $\overline{G}$ is the complement of G. LeSaulnier and West [7] characterized the n-vertex trees having the largest total acquisition number. MacDonald, Wenger, and Wright [9] determined the total acquisition numbers of most grids.

There has also been work on total acquisition involving random graphs and random initial distributions. Bal et al. [1] studied the total acquisition number of random graphs, determining that $p = \frac{\log_2 n}{n}$ is the threshold function for $a_t(G(n,p)) = 1$ in the usual binomial random graph model with independent edge probability p. Infeld, Mitsche, and Prałat [4] studied total acquisition on random geometric graphs with n vertices distributed randomly in

a $\sqrt{n}$-by-$\sqrt{n}$ square, adjacent when their distance is at most r; the result is asymptotically almost surely $\Theta(n/(r \lg r)^2)$ when $0 \le r \lg r \le \sqrt{n}$. Godbole et al. [2] instead consider a total acquisition protocol after a random initial distribution of n units to the vertices of the n-vertex path, showing that the expectation of the resulting total acquisition number tends to a value cn with $.242 \le c \le .375$.

Note that unit and then fractional acquisition provide more flexibility in choosing moves than total acquisition does; thus $a_f(G) \le a_u(G) \le a_t(G)$. Wenger [12] determined the fractional acquisition number of every graph: $a_f(G) = 1$ if G is connected and G is not a path or a cycle, but $a_f(P_n) = a_f(C_n) = \lceil \frac{n}{4} \rceil$ where P_n and C_n are the n-vertex path and cycle, respectively.

We begin the study of unit acquisition number. In Section 2, we obtain general bounds. Let $\delta(G)$ and $\Delta(G)$ denote the minimum and maximum vertex degrees in G, respectively. For a graph G with n vertices, we prove $a_u(G) \le (n-1)/\delta(G)$; equality holds if and only if $G \in \{K_n, C_5\}$, where K_n is the n-vertex complete graph. Also $a_u(G)$ is at most the minimum size of a maximal matching in G, and this bound is sharp on infinitely many graphs with maximum degree k when $k \ge 4$. Also $a_u(G) \le \Delta(G)$ and $a_u(G) \le \sqrt{n-1}$ when G is an n-vertex tree with diameter at most 4.

In Section 3, we construct arbitrarily large trees with maximum degree 5 having unit acquisition number 1. This is very different from total acquisition, where each vertex v can receive weight at most $2^{d(v)}$, and hence $a_t(G) \ge n/32$ when G is a tree with maximum degree at most 5.

A *caterpillar* is a tree in which all non-leaf vertices lie on one path. In Section 4, we characterize the caterpillars G such that $a_u(G) = 1$. This leads to a linear-time algorithm to determine $a_u(G)$ when G is a caterpillar; Slater and Wang [11] found such an algorithm for $a_t(G)$ on caterpillars.

In Section 5, we prove that $a_u(G) = 1$ whenever G has diameter 2, except that the value is 2 for C_5 and the Peterson graph. The question from [6] whether $a_t(G) \le 2$ whenever G has diameter 2 remains open.

We end this introduction with several open questions about unit acquisition. Recall that deciding $a_t(G) = 1$ is NP-complete, while "$a_f(G) = 1$?" can be answered in time linear in $|V(G)|$ by the results in [12].

Question 1. What is the complexity of deciding whether $a_u(G) = 1$?

Although we have determined which caterpillars have unit acquisition number 1, the question seems more difficult for general trees.

Question 2. Is there a simple characterization of the trees with unit acquisition number 1?

Our construction of arbitrarily large trees with maximum degree 5 having unit acquisiton number 1 suggests an obvious question.

Question 3. Is there a bound on the number of vertices in a graph G with maximum degree 4 such that $a_u(G) = 1$?

We suspect that the answer to this question is yes, and that the maximum number of vertices is somewhere around 250.

2 Basic results

We maintain the initial restriction that unit acquisition moves transfer one unit of weight. The additional flexibility of moving more weight in one move does not change any results, and the restriction simplifies the analysis in many places. We thus view the n units of weight on an n-vertex graph as *chips*; the initial unit of weight at a vertex v is the *chip* for v, denoted c_v. Finally, we call a sequence of unit acquisition moves a *protocol*, and a protocol on a graph G is *optimal* if it leaves only $a_u(G)$ vertices with positive weight.

Proposition 2.1. *Every protocol is finite.*

Proof. On an n-vertex graph G, the sum of the squares of the vertex weights is initially n, cannot exceed n^2, and increases by at least 2 with every unit acquisition move. It increases by at least 2 because $a \le b$ implies $(b+1)^2 + (a-1)^2 - a^2 - b^2 = 2(b-a) + 2 \ge 2$. □

Proposition 2.2. $a_u(P_n) = a_u(C_n) = \lceil n/4 \rceil$.

Proof. From [6] and [12], $a_t(C_n) = a_t(P_n) = \lceil n/4 \rceil$ and $a_f(C_n) = a_f(P_n) = \lceil n/4 \rceil$. In the introduction we observed $a_f(G) \le a_u(G) \le a_t(G)$ for every graph G. □

Example 2.3. In studying total acquisition or fractional acquisition numbers, it suffices to consider acyclic graphs. In total acquisition, weight can leave a vertex at most once, after which no weight can move to it; hence the set of edges used in a total acquisition protocol is acyclic. In fractional acquisition, any connected graph having maximum degree at least 3 has a spanning tree with maximum degree at least 3, which suffices for fractional acquisition number 1, while $a_f(G) = \lceil |V(G)|/4 \rceil$ when $G \in \{P_n, C_n\}$ [12]. However, optimal protocols for unit acquisition may need edge sets with cycles. For example, if G is the graph in Figure 1, then $a_u(G) = 1$, but $a_u(G - e) = 2$ when e is any edge in G.

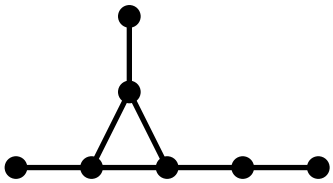

Figure 1: A minimal graph with unit acquisition number 1.

In a rooted tree, the *parent* of a non-root vertex v is its neighbor along the path from v to the root. An *ascending tree* is a rooted weighted tree such that the weight of every leaf is at most the weight of its parent, and the weight of every non-root non-leaf vertex is strictly less than the weight of its parent.

Observation 2.4. If a unit aquisition protocol on a tree T can turn it into an ascending tree, then $a_u(T) = 1$.

Proof. The weight on a leaf in T can be moved to the root one unit at a time. Repeating this moves all weight in the tree to the root. □

Let $N_H(v)$ denote the set of neighbors of a vertex v in a graph H.

Lemma 2.5. *If a graph G has a spanning tree with every vertex having distance at most 2 from the root, and some two vertices with distance 2 from the root are adjacent in G, then $a_u(G) \leq 2$.*

Proof. Let v be the root, with children $x_1, \ldots, x_k$. We may assume that a child z of x_1 has a neighbor w with distance 2 from v in the tree. First move c_w to z. Now in two unit moves per vertex, z can acquire the weight from each other child of x_1. After z acquires all these chips, move c_{x_1} from x_1 to v. Now all weight not on z lies in an increasing tree with root v. □

Proposition 2.6. *If G is an n-vertex graph, then $a_u(G) \leq (n-1)/\delta(G)$, with equality if and only if $G \in \{K_n, C_5\}$.*

Proof. Clearly $a_u(K_n) = 1$, and $a_u(C_5) = 2$ is easy to check, so these achieve equality.

For general G, let S be a largest set of vertices such that the distance between any two is at least 3. Let $S = \{v_1, \ldots, v_m\}$. Any vertex not in S has distance at most 2 from S. Partition $V(G)$ into sets $V_1, \ldots, V_m$ with $v_i \in V_i$ by assigning each vertex not in S to the set for any nearest vertex in S (there may be more than one choice). For each i, the set V_i is

the vertex set of a tree T_i contained in G such that $d_{T_i}(v_i) = d_G(v_i)$ and each vertex of T_i has distance at most 2 from v_i.

Let x_i be a vertex of $N_{T_i}(v_i)$ with least degree in T_i, and let $X_i = N_{T_i}(x_i) - \{v_i\}$. Move c_{x_i} from x_i to v_i, making $T_i - X_i$ an ascending tree in G rooted at v_i. All weight on $T_i - X_i$ can now move to v_i, so $a_u(T_i) \le d_{T_i}(x_i)$. By the choice of S, we have $N_G(v_i) \subseteq V_i$. Hence

$$a_u(T_i) \le d_{T_i}(x_i) \le \frac{|V(T_i)| - 1}{d_G(v_i)} \le \frac{|V(T_i)| - 1}{\delta(G)}. \tag{1}$$

Therefore,

$$a_u(G) \le \sum_{i=1}^{m} a_u(T_i) \le \sum_{i=1}^{m} \frac{|V(T_i)| - 1}{\delta(G)} \le \frac{n-1}{\delta(G)}. \tag{2}$$

Now suppose $a_u(G) = \frac{n-1}{\delta(G)}$; note that $\delta(G) > 1$ is needed unless $G = K_2$. Equality always holds for K_n, so assume $G \neq K_n$. Equality at the end of (2) requires $m = 1$, which by the definition of S requires diameter 2. Equality at the end of (1) requires $d_G(v_1) = \delta(G)$. Equality in the middle of (1) requires that all neighbors of v_1 have the same degree in T_1. If some vertex at distance 2 from v_1 has two neighbors in $N(v_1)$, then T_1 can be changed to reduce the bound. Hence each vertex at distance 2 from v has another neighbor at distance 2, since $\delta(G) > 1$. Now Lemma 2.5 implies $a_u(G) \le 2$.

Let $k = \delta(G)$. Equality in (2) now requires $(n-1)/k = 2$. This also equals $d_{T_1}(x_i)$, so each neighbor of v has one child in T_1. Recall also that each vertex at level 2 has no neighbor at level 1 other than its parent. Let $X = N_G(v)$. Since v has minimum degree in G, every vertex of X is thus adjacent to all except possibly one other vertex of X. If $k > 2$, then X can acquire all the weight from level 2, followed by accumulating all except c_v at one vertex of X, which then acquires c_v to reach $a_u(G) = 1$. Hence we may assume $k = 2$, and now avoiding $a_u(G) = 1$ requires $G = C_5$. □

With a bit more work, Lemma 2.5 can be used to show that $a_u(G) \le 2$ when G has diameter 2. We omit this proof, because in Section 5 we prove the stronger result that $a_u(G) = 1$ when G has diameter 2 and is not C_5 or the Petersen graph.

Our next upper bound is sharp more often. We begin with a lemma used to prove equality in the bound for many graphs. Two chips *meet* when they reach the same vertex.

Lemma 2.7. *Given vertices u and v in a graph G, let S be a minimal u,v-cut in G. If every vertex in S has degree 2 in G, and u and v have no neighbors in S, then the chips from u and v cannot meet via unit acquisition moves.*

Proof. Because S is a minimal u,v-cut and every vertex of S has degree 2, no edges are induced by S. If a protocol moves a chip along an edge with endpoint $x \in S$, then the first move along an edge incident to x reduces the weight of x to 0 or brings x the chip from a neighbor (which then has weight 0). Since $u, v \notin N(x)$, the chip transferred is not c_u or c_v. Also, once an endpoint of an edge has weight 0, the edge cannot be used again.

For each $x \in S$, delete the first edge at x used by the protocol, or x itself if no edge at x is used. This leaves u and v in distinct components, and no move can transfer c_u or c_v from one component to the other. Hence these two chips cannot meet. □

Proposition 2.8. *If G is a connected graph, then $a_u(G)$ is at most the minimum size of a maximal matching in G. For $m, k \in \mathbb{N}$ with $k \geq 4$, some graph $G_{m,k}$ with maximum degree k has a maximal matching of size m and $a_u(G_{m,k}) = m$, achieving equality in the bound.*

Proof. Let M be any maximal matching in G. The vertices of G can be partitioned into $|M|$ sets that induce trees with diameter at most 3, each of which has unit acquisition number 1. Thus $a_u(G) \leq |M|$.

To construct $G_{m,k}$, first let H_k be the tree with $2k$ vertices having two central vertices of degree k and $2k-2$ leaves. Form $G_{m,k}$ from m disjoint copies of H_k as follows. For $1 \leq i \leq k-1$, let x and y be leaf neighbors of the two central vertices in the ith copy of H_k, such that x and y still have degree 1 in the graph being formed. Also choose leaf neighbors x' and y' of the two central vertices in the $(i+1)$th copy of H_k. Merge x with x' and y with y', forming two vertices of degree 2. Note that the resulting graph $G_{m,k}$ decomposes into M copies of H_k, and the $m-1$ pairs of vertices formed in the merging steps are vertex cuts in which each vertex has degree 2. Figure 2 shows $G_{4,5}$.

The central vertices of the copies of H_k form a maximal matching of size m in $G_{m,k}$. Each vertex covered by this matching has a leaf neighbor in $G_{m,k}$. Let u_i be such a leaf in the ith copy of H_k. Vertices u_i and u_j are separated by a vertex cut containing no neighbor of u_i or u_j in which each vertex has degree 2. By Lemma 2.7, no two chips in $\{c_{u_1}, \ldots, c_{u_m}\}$ can meet under any protocol. Hence $a_u(G_{m,k}) \geq m$, and equality holds. □

LeSaulnier et al. [6] proved $a_t(T) \leq \sqrt{n \lg n}$ when T is an n-vertex tree with diameter 4 and constructed an n-vertex tree T_n with diameter 4 such that $a_t(T_n) \geq (1-o(1))\sqrt{(n/2)\lg n}$. For unit acquisition number, a stronger bound holds.

Proposition 2.9. *If T is a tree of diameter at most 4, then $a_u(T) \leq \sqrt{n-1} \leq \Delta(T)$. Equality holds in the first inequality only when $n-1$ is a square and T is the tree of diameter 4 such that the central vertex and all its neighbors have degree $\sqrt{n-1}$.*

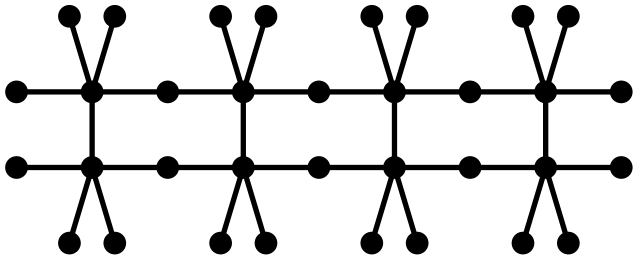

Figure 2: The graph $G_{4,5}$.

Proof. Since trees of diameter at most 3 have unit acquisition number 1, we may assume that T has diameter 4. Let v be the central vertex. Let $k = \sqrt{n-1}$. Note that when T has diameter at most 4, always $\Delta(T) \geq k$, since a tree of diameter at most 4 with maximum degree less than k has fewer than $1 + k^2$ vertices.

When $d_T(v) \leq k$, move all weight to $N_T(v)$ to obtain $a_u(T) \leq d_T(v) \leq k$. Otherwise, move weight 1 to v from the neighbor u of v having least degree. Now the tree formed by deleting u and its leaf neighbors is ascending, so $a_u(T) \leq 1 + d_T(u) - 1$. Since in this case $d_T(v) > k$, by the pigeonhole principle $d_T(u) < k$. Thus $a_u(T) \leq k$.

Avoiding $a_u(T) < k$ first requires $d_T(v) \geq k$. Since $n \geq 1 + d_T(v)d_T(u)$, having $d_T(v) \geq k$ requires $d_T(u) \leq k$. Since $a_u(T) \leq d_T(u)$, avoiding $a_u(T) < k$ requires $k = d_T(u) = d_T(v)$, and T is the tree specified in the theorem statement.

For this tree T, if no weight moves to the root, then weight remains in k disjoint subtrees. If a chip moves to the root, then the first such chip leaves $k-1$ isolated vertices with positive weight. Hence $a_u(T) \geq k$. □

3 Trees T with $\Delta(T) = 5$ and $a_u(T) = 1$

For total acquisition, Lampert and Slater [5] proved $a_t(G) \geq |V(G)|/2^{\Delta(G)}$ by observing that a vertex v cannot acquire weight more than $2^{d_G(v)}$ via total acquisition moves. For unit acquisition, no analogous result can be proved, since there is no bound on the amount of weight that a vertex in a tree of maximum degree 5 can acquire.

Theorem 3.1. *For $d \in \mathbb{N}$, there is a tree T_d with maximum degree 5 and at least d vertices such that $a_u(T_d) = 1$.*

Proof. We inductively construct a rooted tree T_d with maximum degree 5 in which we can move all the weight to an ascending tree. We use T'_d to denote the ascending version of T_d

after this partial protocol. Since T'_d is ascending, $a_u(T_d) = 1$.

We construct T_d from T_{d-1} by adding leaves. Since T_d contains T_{d-1}, the partial protocol can be followed on T_{d-1} to obtain T'_{d-1} within T_d. Further unit acquisition moves will then produce T'_d. The tree T_d will have d levels, with the root as the first. Thus T_1 consists of only the root. We also specify T_2 explicitly; it consists of the root plus five children. Let T'_2 be the ascending tree produced by moving the chip from one child to the root (see Figure 3).

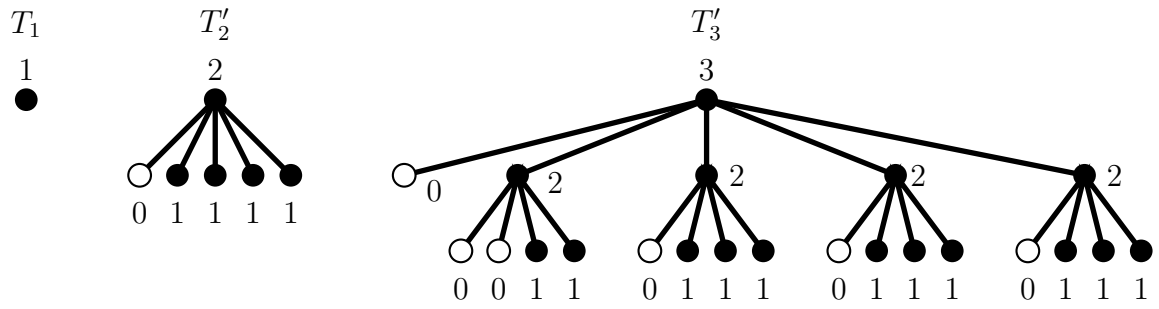

Figure 3: The first three trees for Theorem 3.1, converted to ascending trees.

We call a vertex with positive weight in T'_d *active*, except that some leaves retaining weight 1 may be designated inactive. The root vertex is at level 1; the leaves are at level d. In T'_d, the active vertices at level i have weight $d+1-i$. Let a_d denote the number of active vertices at level d in T'_d, so $a_2 = 4$.

For $d \geq 3$, we construct T_d from T_{d-1} by appending four leaves at each active vertex of T'_{d-1} on level $d-1$ and then viewing each vertex as starting with weight 1. To convert T_d to T'_d, first perform the protocol on the copy of T_{d-1} within T_d formed by levels 1 through $d-1$. By the induction hypothesis, this puts weight $d-i$ at each active vertex in level i, for $i < d$.

We now want to use chips from the leaves to increase the weight by 1 at (most) active non-leaf vertices. For i from 1 through $d-1$ successively, for each active vertex u at level i currently having an active leaf below it on level d, choose such a leaf x and move c_x up the path through the tree to reach u. This is possible, because the tree remains ascending throughout the process. See T_3 in Figure 3.

If there is no such leaf below u, then we instead choose some other remaining leaf x that has weight 1, arbitrarily, and designate x inactive. The tree remains ascending, because the parent of this leaf has weight at least 1. More importantly, when we grow T_{d+1} we will not add leaves below x, since x is inactive. Hence it causes no difficulty if the parent of x also ends the process with weight 1.

The process thus can be completed if the number of leaves added in forming T_d is at least the total number of active vertices in T'_{d-1}. The value a_i is the number of active vertices left at level i when T'_i is formed, and this always remains the number of active vertices at level i. For $d \geq 3$, we thus have

$$a_d = 4a_{d-1} - \sum_{i=1}^{d-1} a_i.$$

We can continue growing larger trees with the desired properties if $a_d > 0$ for $d \geq 1$.

With $a_1 = 1$ and $a_2 = 4$, writing the recurrence as $4a_{d-1} = \sum_{i=1}^{d} a_i$ for $d \geq 3$ yields $a_3 = 11$. The difference of two consecutive instances of the recurrence yields $a_d = 4a_{d-1} - 4a_{d-2}$ for $d \geq 4$, with $a_2 = 4$ and $a_3 = 11$. The solution $a_d = (3d+5)2^{d-2}$ for $d \geq 2$ is easily checked by induction. As desired, $a_d > 0$ for all d, which completes the proof. □

For graphs with maximum degree 1, 2, or 3, straightforward case analysis shows that a vertex can acquire weight at most 2, 4, or 10, respectively. With maximum degree 4, growing three leaves at active vertices, the corresponding recurrence in the method above is $a_d = 3(a_{d-1} - a_{d-2})$ for $d \geq 4$, with $a_2 = 3$ and $a_3 = 5$. Also $a_4 = 6$, $a_5 = 3$, and $a_6 = -9$, so this construction does not grow beyond depth 5, since the nine vertices generated at level 6 do not suffice to augment the higher active vertices. There are 56 vertices at this point, so a vertex an acquire weight 56. With more careful analysis it may be possible obtain a bound on the number of vertices in a tree with unit acquisition number 1 and maximum degree 4.

4 Unit Acquisition on Caterpillars

Toward the further understanding of unit acquisition on trees, in this section we characterize the caterpillars with $a_u(T) = 1$ and give a linear-time algorithm to compute unit acquisition number on caterpillars.

Definition 4.1. The path obtained by deleting the leaves of a caterpillar is called the *spine* of the caterpillar. Given a caterpillar T, let $v_0, \ldots, v_{k+1}$ denote the vertices in the spine of T, indexed in order. The vertices $v_1, \ldots, v_k$ are the *internal vertices* of the spine of T. In particular, note that v_0 and v_{k+1} are not leaves of T.

For $v \in V(T)$, let $d'(v)$ denote the number of leaf neighbors of v. For $s \in \mathbb{N}$, let $\ell(s) = \sum_{i=1}^{s} \lceil i/2 \rceil = \left\lceil \frac{s+1}{2} \right\rceil \left\lfloor \frac{s+1}{2} \right\rfloor$. During a protocol, let $\mathrm{wt}(v)$ be the current weight on v.

Let S be a set of s consecutive internal vertices on the spine of T, with $S = \{v_i, \ldots, v_{i+s-1}\}$. Let the *pyramid* of S be a set of cells arranged so that $\min\{j-i+1, i+s-j\}$ cells are stacked

above v_j (see Figure 4). Let $b_{j,1}, \ldots, b_{j,\min\{j-i+1,i+s-j\}}$ denote the cells above v_j. Counting columns moving in from both ends shows that the pyramid of S has $\ell(s)$ cells.

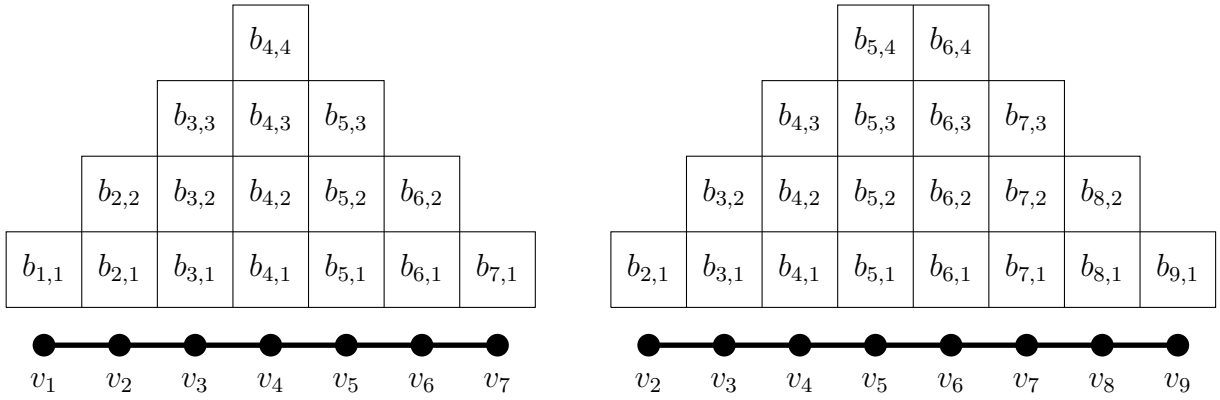

Figure 4: Pyramids of lengths 7 and 8 on the spine of a caterpillar.

Example 4.2. In P_5 and in the caterpillar J obtained from P_5 by appending a leaf at the central vertex, the spine has three vertices, which we label v_0, v_1, v_2 (so $k = 1$). The condition in the theorem below requires $d'(v) \geq 1$ for every internal spine vertex, so $a_u(P_5) > 1$. Indeed, applying Lemma 2.7 to leaf neighbors of the ends of the spine also requires each internal spine vertex to have at least one leaf neighbor when $a_u(T) = 1$. Since J has only one internal spine vertex, and $d'(v_1) = 1$, the condition below is satisfied, and $a_u(J) = 1$.

Theorem 4.3. *Let T be a caterpillar with spine vertices $v_0, v_1, \ldots, v_k, v_{k+1}$ in order. The unit acquisition number of T is 1 if and only if for $1 \leq s \leq k$, every set S of s consecutive internal vertices on the spine satisfies $\sum_{v \in S} d'(v) \geq \ell(s) = \lceil \frac{s+1}{2} \rceil \lfloor \frac{s+1}{2} \rfloor$.*

Necessity and sufficiency of the condition both take some work, so we separate these proofs into two items.

Theorem 4.4. *The condition on S in Theorem 4.3 is necessary for $a_u(T) = 1$.*

Proof. Let $S = \{v_j, \ldots, v_{j+s-1}\}$. To simplify notation, let $u_i = v_{i+j-1}$ for $0 \leq i \leq s+1$, so $S = \{u_1, \ldots, u_s\}$. Assume $a_u(T) = 1$. As noted in Example 4.2, $d'(u_i) \geq 1$ for all i, by Lemma 2.7. Hence u_0 has a leaf neighbor x, and u_{s+1} has a leaf neighbor y, and in some protocol $\mathcal{A}$ the chips c_x and c_y must meet. We prove $\sum_{v \in S} d'(v) \geq \ell(s)$.

Since chips in fact are indistinguishable, we may assume that at any point in $\mathcal{A}$ the chips on a vertex v are listed from bottom to top in their order of arrival at v. We may also

assume that the a chip moved off a vertex is the most recent chip that arrived there, again since chips are indistinguishable. In particular, the bottom chip on v is always c_v. Also, since $a_u(T) = 1$ requires all chips eventually to be on one vertex, it requires c_x and c_y to meet even under this restriction on the movement of chips.

Since a chip arriving at v will be placed on the top, by the weight rule for moves the height of a chip in its current stack strictly increases each time it moves. Since c_x and c_y start two steps from S, this implies that they can never occupy cells in the pyramid over S. Indeed, since they start two steps away from S, when on a vertex of S they must be at least one step above the pyramid (here we consider the chip on u_j to be at the point $(j, 0)$).

If a chip c_v ever moves from v, then thereafter $\mathrm{wt}(v) = 0$. If c_v moves from v when c_x and c_y are separated by v, then c_x and c_y cannot meet under $\mathcal{A}$. We will consider only protocols that move no such chips before c_x and c_y meet. This also requires that leaf neighbors of spine vertices between c_x and c_y always have weight at most 1.

For $1 \le i \le s$, let $\mu_i = \min\{i, s + 1 - i\}$; note that μ_i is the number of cells above u_i in the pyramid over S. At any point in $\mathcal{A}$, let $g(u_i) = \max\{\mu_i - \mathrm{wt}(u_i) + 1, 0\}$ for $u_i \in S$. Viewing chips at u_i as filling cells above u_i in the pyramid over S, $g(u_i)$ gives the number of empty cells above u_i (there may also be chips above the pyramid when there are no empty cells above a vertex). Recall that c_x and c_y cannot occupy cells inside the pyramid over S.

During $\mathcal{A}$, let $V_{x,y}$ denote the set of vertices in S internal to the path in T joining the current locations of c_x and c_y. Let l denote the number of chips on leaf neighbors of vertices in $V_{x,y}$. As noted earlier, such leaves have at most one chip, so l equals the number of leaf neighbors of vertices in $V_{x,y}$ having weight 1.

At a given time in $\mathcal{A}$, let $h = \sum_{u_i \in V_{x,y}} g(u_i) - l$. Under the assumption $\sum_{v \in S} d'(v) < \ell(s)$, we will show by induction on the number of moves in $\mathcal{A}$ that $h > 0$ and $V_{x,y} \neq \varnothing$ throughout $\mathcal{A}$. With $V_{x,y}$ remaining nonempty, c_x and c_y never meet. Initially, $V_{x,y} = S$ and $g(u_i) = \mu_i$ for $u_i \in S$; also $l = \sum_{v \in S} d'(v)$. Thus $h = \ell(s) - l > 0$ by assumption, and $V_{x,y} \neq \varnothing$.

Suppose that $h > 0$ and $V_{x,y} \neq \varnothing$ at some point in $\mathcal{A}$. Consider the next move in $\mathcal{A}$. To decrease h, either $g(u_i)$ must decrease for some $u_i \in V_{x,y}$, or l must increase. If $g(u_i)$ decreases, then some cell $b_{i,r}$ in the column over u_i (between c_x and c_y) becomes filled. Since no vertex of weight 0 can lie between c_x and c_y, we have $i \notin \{1, s\}$.

A chip that moves to $b_{i,r}$ must come from a leaf neighbor of u_i with weight 1 or from $b_{i\pm1,r'}$ with $r' < r$. If the chip moves to $b_{i,r}$ from a leaf neighbor of u_i, then $g(u_i)$ decreases by 1, and l decreases by 1, so h and $V_{x,y}$ are unchanged. If the chip moves to $b_{i,r}$ from $b_{i\pm1,r'}$, then let $u_{i'}$ be the vertex contributing the chip. By our restrictions on chip movement, the

weight on $u_{i'}$ prior to the move is r', and $b_{i',r'}$ is empty after the move. Also, since $r' \le \mu_i$, neither c_x nor c_y can be on $u_{i'}$. This yields $i, i' \in V_{x,y}$, so $\sum_{u_i \in V_{x,y}} g_{u_i} - l$ is unchanged, and $V_{x,y}$ and h are both unchanged by this move.

Therefore a decrease in h must come from l increasing. This requires moving c_x and c_y away from each other. If such movement does not enlarge $V_{x,y}$, then l does not increase and h does not change. Hence we may assume that c_y moves to the right from u_j to u_{j+1}, adding u_j to $V_{x,y}$, with u_j still having l_j leaf neighbors with weight 1.

Since y began to the right of S, there was a most recent time when c_y moved from u_{j+1} to u_j. At that point, h grew by at least l_j, since then $g(u_{j+1}) = g(u_i) = 0$ and there were at least l_j leaves of u_j with weight 1. This increase of l_j has not been counted in any of the moves analyzed above. Furthermore, between that move and when y moves back, no chip from a leaf neighbor of u_j can move into a cell in the pyramid over S, because it would have to land above c_y, which is already outside the pyramid. Hence the bonus contribution of l_j to h persists until c_y moves away, keeping h still positive after that move.

Hence h remains positive and $V_{x,y}$ remains nonempty, as desired. □

Theorem 4.5. *The condition specified in Theorem 4.3 is sufficient for $a_u(T) = 1$.*

Proof. Recall that the condition is the requirement that $\sum_{v \in S} d'(v) \ge \ell(s)$ whenever $1 \le s \le k$ and S is a set of s consecutive vertices among the internal vertices $v_1, \ldots, v_k$ along the spine, where $d'(v)$ is the number of leaf neighbors of v and $\ell(s) = \left\lceil \frac{s+1}{2} \right\rceil \left\lfloor \frac{s+1}{2} \right\rfloor$. Let $v_{i,m}$ denote the mth leaf neighbor of vertex v_i along the spine, for $1 \le m \le d'(v_i)$. Let A be the set of leaf neighbors of $v_1, \ldots, v_k$, so $|A| \ge \ell(k)$. Let B be the set of cells in the pyramid over $\{v_1, \ldots, v_k\}$, so $|B| = \ell(k)$.

We specify edge costs w for the complete bipartite graph with parts A and B, by

$$w(v_{j,m} b_{i,h}) = \begin{cases} |i-j| & \text{if } h > |i-j| \\ \infty & \text{if } h \le |i-j| \end{cases}.$$

By assumption, $|A| \ge |B|$. If $|A| > |B|$, vertices can be added to B (with zero cost on incident edges) until the parts have equal size. Let M be a perfect matching of minimum cost in the resulting graph. Such a matching can be obtained by the Hungarian Algorithm [8, 10].

Claim 1: *M has finite cost.* Let H be the subgraph formed by the edges of positive finite cost. We prove that H contains a matching that covers B. By Hall's Theorem [3], this holds if and only if $|N_H(X)| \ge |X|$ whenever $X \subseteq B$. If if fails, then let $X \subseteq B$ be a minimal set such that $|N_H(X)| < |X|$.

Because X is minimal, it follows that the subgraph of H induced by X and $N_H(X)$ is connected. For $b_{i,h} \in B$, by definition $N_H(b)$ is the set of leaf neighbors of the spine vertices v_j such that $|i-j| < h$; these spine vertices are consecutive along the spine. If the union of these segments for the vertices of X is not a single consecutive segment, then again X is not a minimal failure of Hall's Condition. Thus $N(X) = \sum_{j=a}^{b} d'(v_j)$ for some a and b.

By definition, all leaf neighbors of v_i are in $N_H(b_{i,h})$. Hence each element of X lies in the pyramid over $\{v_a, \ldots, v_b\}$. Thus $|X| \le \ell(b-a+1)$. We now have $|N(X)| = \sum_{j=a}^{b} d'(v_j) \ge \ell(b-a+1) \ge |X|$. Thus Hall's Condition is satisfied, and M has finite cost.

We use M to specify a protocol that converts T into an ascending tree. We fill all cells in the pyramid by moving to each cell the chip on the leaf vertex matched to it in M. This places weight $1+\mu_i$ on v_i for $1 \le i \le k$. Vertices v_0 and v_{k+1} retain their original chips and weight 1. Leaves not matched by edges with positive cost in M also retain weight 1 in T. Hence this new distribution T' is an ascending tree, with unit acquisition number 1.

The chip from the leaf $v_{j,m}$ matched to cell $b_{i,h}$ will move $|i-j|$ steps along the spine to reach the assigned destination v_i, but we still must determine a feasible order for these moves to occur in a protocal. We begin by representing the desired moves on a grid. Place a filled circle for $b_{i,h}$ at the lattice point (i,h). For an edge $v_{j,m}b_{i,h}$ in M, draw a line segment from $(j, h-|i-j|)$ upward along a diagonal to the circle at (i,h). When $i=j$, the segment has length 0 (see Figure 5). Each circle corresponding to a cell in the pyramid is the top end of one segment, and the slope of each segment is ± 1.

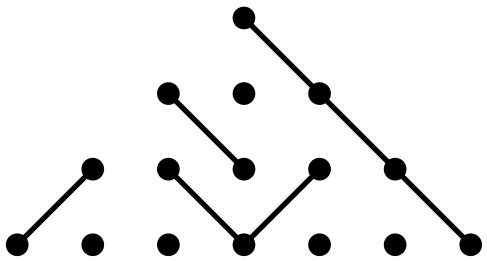

Figure 5: A matching being converted to a protocol.

These line segments may overlap (one may contain another), but they do not cross.

Claim 2: *Two non-collinear segments in the grid diagram cannot share a lattice point (a,b) if one extends above (a,b) and the other extends below (a,b).* If the segment L extending above contains $(a-1, b+1)$ and the segment L' extending below contains $(a-1, b-1)$, then M contains edges $v_{j,m}b_{i,h}$ and $v_{j',m'}b_{i',h'}$ such that $j > a \ge i$ and $j' < a \le i'$. These edges

have cost $i-j$ and $i'-j'$, but since $j' < j$ and $i \le i'$, we have $|i-j'|+|i'-j| < (i-j)+(i'-j')$. Hence replacing these edges in M with the edges $v_{j,m}b_{i',h'}$ and $v_{j',m'}b_{i,h}$ yields a matching M' with smaller cost, contradicting the minimality of M.

Similarly, if the segment L extending above contains $(a+1, b+a)$ and the segment L' extending below contains $(a+1, b-1)$, then M contains edges $v_{j,m}b_{i,h}$ and $v_{j',m'}b_{i',h'}$ such that $j < a \le i$ and $j' > a \ge i'$, and the edges $v_{j,m}b_{i',h'}$ and $v_{j',m'}b_{i,h}$ have smaller total cost.

Finally, we convert the grid diagram to a protocol. We have $\ell(s)$ segments reaching the points for cells in the pyramid; they may have length 0. A segment D is *below* a segment D' if D and D' contain points with the same horizontal coordinate such that the point in D is vertically below the point in D', or if D and D' are collinear and the destination cell of D has a smaller vertical coordinate than that of D'. By Claim 2, no two distinct segments can be below each other. Linearly order the cells of the pyramid by iteratively taking a remaining cell $b_{i,h}$ such that no remaining cell has its segment below that of $b_{i,h}$.

The resulting linear order on the points is the order in which we fill the cells to obtain the ascending tree described earlier. When $b_{i,h}$ is to be filled, and $v_{j,m}b_{i,h}$ is the edge incident to it in M, the chip from the mth leaf neighbor of v_j is moved to v_i. Since all segments below the edge for this segment have been processed, and this segment is not below any edge that has been processed, the current weights on the spine vertices from v_j to v_i permit this chip to move as desired. Hence we convert T to an ascending tree, and $a_u(T) = 1$. □

Theorem 4.3 yields a linear-time algorithm for $a_u(T)$ on caterpillars.

Corollary 4.6. *There is a $O(|V(T)|)$-time algorithm that determines the unit acquisition number of a caterpillar T.*

Proof. Let T be a caterpillar, drawn with the spine in order from left to right. The algorithm iteratively removes the largest caterpillar subtree having unit acquisition number 1 that contains the left end of the remaining spine. This caterpillar is determine by applying the condition in Theorem 4.3.

When removing a subtree T', it may be necessary to count a member of the spine of T belonging to T' as a leaf of T' in some cases: 1) the leftmost member of T_i has no leaf neighbors, 2) the rightmost member of T_i has no leaf neighbors, or 3) the two rightmost members of T_i have no leaf neighbors (see Figure 6).

The algorithm partitions the vertex set of T into set inducing caterpillars with unit acquisition number 1, so it provides an upper bound for $a_u(T)$. If $a_u(T)$ is less than the

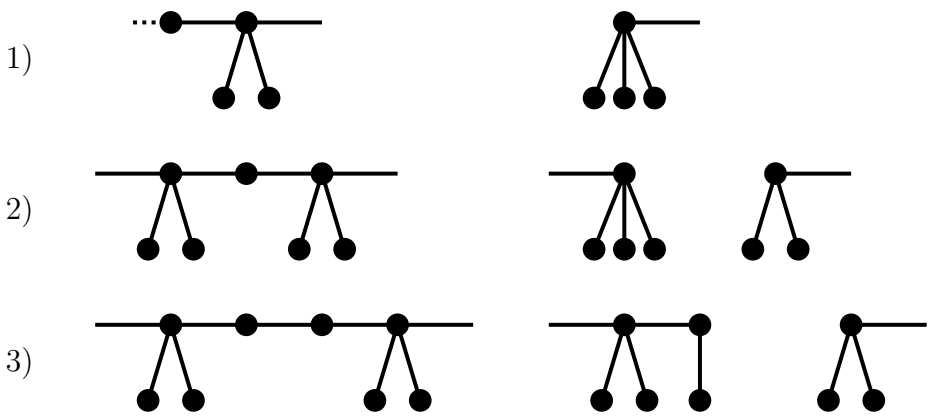

Figure 6: Situations where a spine vertex is treated as a leaf in a subtree.

bound provided by the algorithm, then an optimal protocol collects all the weight at $a_u(T)$ vertices. The weight on each one of these vertices comes from a subtree of T that is a caterpillar. Since this partitions $V(T)$ into fewer sets inducing caterpillars, and the spines of these caterpillars and those found by the algorithm, some caterpillar T^* used by the optimal protocol properly contains some caterpillar T' found by the algorithm, extending farther on *both* ends. Now the necessity of the condition in Theorem 4.3 for T^* implies that the condition also holds for the extension of T' to the right end of T^*, and then sufficiency of the condition contradicts the choice of T' to be a largest caterpillar from its left end.

To run the algorithm, we first compute the values $d'(v)$ for vertices along the spine, in time linear in $|V(T)|$. To understand the subsequent running time, consider the extraction of the first caterpillar. The condition of Theorem 4.3 must be checked. Since the condition is specified over all segments of internal vertices of the subtree, it also holds for all subsegments. That is, we grow the potential caterpillar from the left. If the condition holds for the first $i-1$ internal vertices of the spine, then we next test the segments ending at the ith internal vertex. At the point where the test first fails, the number of sums that have been tested is quadratic in the length of the spine examined so far, but the number of leaves adjacent to those spine vertices is also quadratic in that length. Thus, the number of sums performed is linear in the number of vertices that are collected by the first caterpillar. Since this holds for each subcaterpillar, the entire algorithm runs in time linear in $|V(T)|$. □

5 Diameter 2

In this section we prove $a_u(G) = 1$ for every graph G with diameter 2 except C_5 and the Petersen graph, where we show that the value is 2.

We begin with a lemma. A *solo-neighbor* of a vertex u in a clique Q with at least two vertices is a vertex outside Q whose only neighbor in Q is u.

Lemma 5.1. *Let G be a graph with diameter 2. If Q is a clique in G with at least two vertices, and there exists $u \in Q$ such that u has no solo-neighbor, then $a_u(G) = 1$.*

Proof. Given such a vertex u, let v be another vertex in Q. Since u has no solo-neighbor relative to Q, every vertex has a path of length at most 2 to v that does not pass through u. Thus moving c_u to v converts G to a graph with a spanning ascending tree. □

The *girth* of a graph G having a cycle is the minimum length of a cycle in G.

Theorem 5.2. *If G is a graph with diameter 2 that is not C_5 or the Petersen graph, then $a_u(G) = 1$.*

Proof. A graph with diameter 2 that is not a star has girth at most 5. We consider cases.

Case 1: *G has girth 3.* Let Q be a maximum clique in G, so $|Q| \geq 3$. Let S be the set of vertices outside Q having a neighbor in Q, and let $U = V(G) - Q - S$. If $U = \varnothing$, then $a_u(G) = 1$. If some vertex u in Q has no solo-neighbor, then Lemma 5.1 yields $a_u(G) = 1$.

In the remaining case, $U \neq \varnothing$, and every vertex of Q has a solo-neighbor. Choose $u, v, w \in Q$. Let S' be the set of solo neighbors of vertices in $Q - \{w\}$. If $N_G(z) \subseteq S'$ for some $z \in U$, then w has distance 3 from z. Hence U has no such *bad* vertex.

Begin by moving all weight from solo-neighbors of v to v. Next move all weight from solo-neighbors of u to v, two moves per chip; note that now $\mathrm{wt}(v) \geq 3$. Next move c_u to w, and let each vertex of $Q - \{u, v, w\}$ acquire the chip from one of its solo-neighbors. Now $\mathrm{wt}(v) \geq 3$, all vertices of $Q - \{u, v\}$ have weight 2, other vertices retaining positive weight have weight 1, and every vertex with weight 1 has a neighbor in $Q - \{u\}$ or a neighbor of weight 1 with a neighbor in $Q - \{u\}$ (because U has no bad vertex). Hence positive weight remains only on an ascending tree rooted at v, and $a_u(G) = 1$.

Case 2: *G has girth 4.* Let w, x, y, z be the vertices of a 4-cycle in order. Move c_w to x and c_z to y. We claim that all the weight is now contained in disjoint ascending trees T_x and T_y to x and y. If so, then x and y can acquire all the weight, after which it can be combined along the edge xy.

Let $X = N_G(x) - \{y, w\}$ and $Y = N_G(y) - \{x, z\}$. The sets X and Y are disjoint. Let u be a vertex with weight 1 not in $X \cup Y$. If u has no neighbor in X or Y, then distance at most 2 from both x and y requires $u \in N_G(w) \cap N_G(z)$. Now u, w, z form a 3-cycle, which is forbidden. Hence u has a neighbor in X or Y, and the two desired trees exist.

Case 3: *G has girth 5 and is not C_5 or the Petersen graph.* The only graphs having girth 5, diameter 2, and minimum degree at most 3 are C_5 and the Petersen graph. Hence we may assume $\delta(G) \geq 4$. Let v, w, x, y, z in order be the vertices of a 5-cycle C in G. Every vertex outside $V(C)$ has at most one neighbor on C, since G has girth 5. Let $V = N_G(v) - V(C)$, and similarly define W, X, Y, Z. Let N be the set of vertices with no neighbors on C.

Since $\delta(G) \geq 4$, each of V, W, X, Y, Z has size at least 2. Hence we can move chips from two vertices of X to x, giving x weight 3. Also, diameter 2 requires every vertex of Y to have a neighbor in W in order to reach w. Hence there is an edge $w'y'$ with $w' \in W$ and $y' \in Y$. Move $c_{w'}$ to w and $c_{y'}$ to y, giving weight 2 to w and y.

We claim that now all weight in G lies in an ascending tree rooted at x, so $a_u(G) = 1$. Since the paths to x along C strictly ascend in weight, the claim holds immediately for all vertices except those in N. To reach w and y in two steps, each vertex of N must be adjacent to one vertex in each of W and Y (exactly one, to avoid 4-cycles). However, avoiding 3-cycles requires that no vertex of N is adjacent to both w' and y'. Hence each vertex of N has a neighbor in W or Y with weight 1. Since w and y have weight 2, the claim holds. □

Since $a_u(C_5) = 2$ (Proposition 2.2), it remains only to analyze the Petersen graph.

Theorem 5.3. *The unit aquisition number of the Petersen graph is 2.*

Proof. For the upper bound, we can use the edges of a perfect matching to bring weight 2 to each vertex of a 5-cycle. With $a_u(C_5) = 2$, we can then acquire all weight onto two vertices.

Now suppose that some unit acquisition protocol brings all weight to a single vertex of the Petersen graph G. At any point in the protocol, let H denote the subgraph induced by the remaining vertices with positive weight. Note that H must always be connected.

Consider the first moment when some vertex v acquires weight 3. We may assume that the weight of v reaches 3 before any moves occur that are not used to bring weight 3 to v. Let $N(v) = \{x, y, z\}$. There are two ways that v can acquire weight 3, up to symmetry.

Case 1: *v acquires the original chips from x and z.* The chip from y cannot now move away from y in the next move involving y, because that would leave multiple components in H. Hence we may assume that y next acquires the original chip from w, leaving the situation on the left in Figure 7.

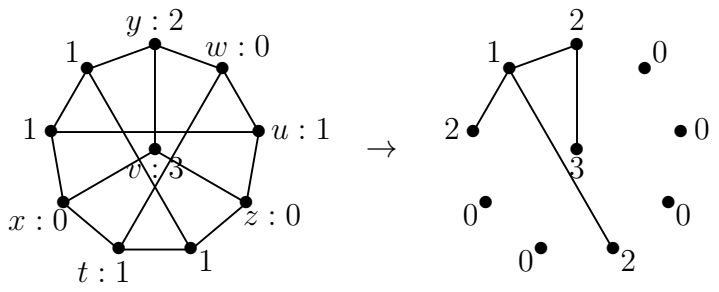

Figure 7: Case 1 for the Petersen graph.

This leaves weight 1 at a leaf u of H that is the common neighbor of w and z. Since acquiring weight from its neighbor would isolate u, the chip from u must eventually move to its remaining neighbor, so we may assume that happens now. Similarly, the common neighbor t of x and w is a leaf of H; by the same reasoning, we may assume that the chip from t now moves to its remaining neighbor. The situation is now as on the right in Figure 7. Because the cut-vertex of H has weight only 1, the protocol will end with three components of positive weight.

Case 2: *v acquires the original chip of x and then acquires the chip of a nonneighbor w after the chip from w is acquired by y.* In this case again the chip from the common neighbor t of x and w must be acquired by its remaining neighbor, leaving the situation in the middle of Figure 8.

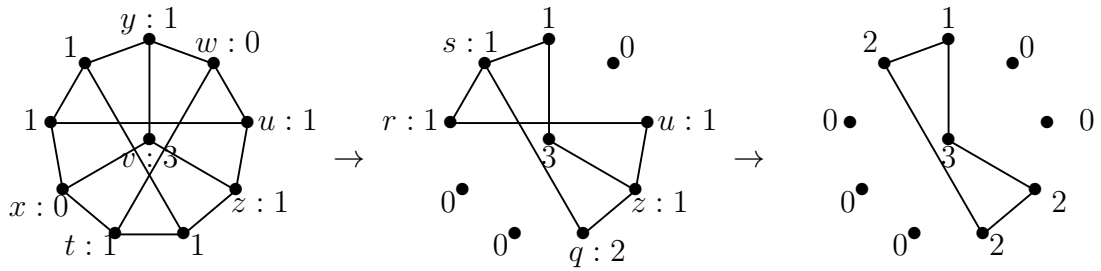

Figure 8: Case 2 for the Petersen graph.

If the first move involving z removes its chip, then we may assume that it is made now unless it moves to the common neighbor q of s and z after s acquires the chip from y or r and then also a chip from q. This will leave v and u in distinct components with positive weight.

In any other way the first move involving z removes its chip, we may assume it happens now. This leaves the remaining neighbor s of y as a cut-vertex of H with weight 1. As in

Case 1, moving the chip off s disconnects H, and either way for s to acquire a chip also disconnects H.

We may therefore assume that z acquires the chip from its neighbor u of weight 1. This leaves the common neighbor r of s and u as a leaf in H with weight 1, so its chip must be acquired by its remaining neighbor in H. The situation now is on the right in Figure 8.

What remains is a 5-cycle $[y, v, z, q, s]$ in which one unit move has been made (from y to v) starting from a distribution with two chips at each vertex. Indeed, this is what results if we use a perfect matching in the Petersen graph to move all the weight onto a 5-cycle.

Now, whether the unit weight from y moves eventually to v or to s, we will be left with a path along which vertices are separated by vertices with strictly smaller weight, and no chips can cross the cut. □

Acknowledgments

The authors would like to thank Noah Prince for helpful discussions on this topic.

References

[1] D. Bal, P. Bennett, A. Dudek, and P. Prałat, The total acquisition number of random graphs, *Electron. J. Combin.* 23 (2016), no. 2, Paper 2.55, 21 pp.

[2] A. Godbole, E. Kelley, E. Kurtz, P. Prałat, and Y. Zhang, The total acquisition number of the randomly weighted path, *Discuss. Math. Graph Theory* 37 (2017), 919–934.

[3] P. Hall, On representation of subsets, *J. London Math. Soc.* 10 (1935), 26–30.

[4] E. Infeld, D. Mitsche, and P. Prałat, The total acquisition number of random geometric graphs, *Electron. J. Combin.* 24 (2017), Paper 3.31, 18 pp.

[5] D. Lampert and P. Slater, The acquisition number of a graph, *Congr. Numer.* 109 (1995), 203-210.

[6] T.D. LeSaulnier, N. Prince, C. Stocker, P.S. Wenger, D.B. West, and P. Worah, Total acquisition in graphs, *SIAM J. Discrete Math.* 27 (2013), no. 4, 1800–1819.

[7] T.D. LeSaulnier and D.B. West, Acquisition-extremal graphs, *Discrete Appl. Math.* 313 (2013), no. 19, 2020–2025.

[8] H.W. Kuhn, The Hungarian method for the assignment problem, *Naval Res. Logist. Quart.* 2 (1955), 83–97.

[9] L. MacDonald, P.S. Wenger, and S. Wright, Total acquisition on grids, *Australas. J. Combin.* 58 (2014), 137–156.

[10] J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Indust. Appl. Math.* 5 (1957), 32–38.

[11] P. Slater and Y. Wang, Some results on acquisition numbers, *J. Combin. Math. Combin. Comput.* 64 (2008), 65–78.

[12] P.S. Wenger, Fractional acquisition in graphs, *Discrete Appl. Math.* 178 (2014), 142–148.