21

Graph Layouts

Author: Zevi Miller, Department of Mathematics and Statistics, Miami University.

Prerequisites: The prerequisites for this chapter are big-O notation and basic concepts of graphs and trees. See Sections 2.2, 9.1, 9.2, 9.4, 10.1, and 10.4 of *Discrete Mathematics and Its Applications*.

Introduction

In this chapter we will discuss a set of graph problems having a common theme. These problems deal with the question of how well a given graph G "fits" inside some other given graph H. Why might we be interested in such problems? The answer is that solutions to such problems tell us much about the best design for interconnection networks of computers and for layouts of circuits on computer chips. The solution to a special case of one of these problems also sheds light on how to represent matrices in a compact way that makes it convenient to manipulate them when executing standard matrix operations.

Dilation

First consider a computer network based on a graph G. This network consists of a different computer located at each vertex of G, with two computers joined by a direct link when the corresponding vertices are joined by an edge in G. (We therefore continue to denote the network by G, its underlying graph.) We can imagine programming these computers to run in parallel, each computer receiving part or all of the original input data and then proceeding to perform its own private computations as specified by the master program controlling all the computers. At each time unit each computer is also allowed to pass the results of the computation just completed to one of its neighboring computers (i.e., the ones joined to it by an edge), and these neighbors will use these results as inputs into their own computations later. We call such a network G an **interconnection network** (or sometimes a **parallel computation network**). By dividing up the work between its different computers, we can expect that an interconnection network can solve at least some problems much faster than a single computer could.

We now take H to be an interconnection network different from G such that H has at least as many vertices as G; that is, $|V(G)| \leq |V(H)|$. Suppose that we have a program P for G to solve some problem as described in the last paragraph, but that G is not available while H is available. (In fact, even if G were available, we might still prefer to use H for standardization reasons.) The question then becomes how to "simulate" the program for G by a program for H which solves the same problem. Informally speaking, a simulation of G by H is a way of describing how, using the program P as a guide, H can accomplish the same task as G by assigning its computers the tasks assigned to those of G. One way of doing this is to make a *mapping* (or *correspondence*) between the vertices of G and the vertices of H so that corresponding vertices carry out the same tasks. The obvious question is how to define such a mapping, and the answer to this question in turn depends on how "effective" a given mapping is in doing the simulation.

Clearly, we need to be a bit more precise about how to evaluate the effectiveness of a mapping. We write

$$f : V(G) \to V(H)$$

to indicate that f is a one-to-one (though not necessarily onto) mapping from the vertices of G to the vertices of H. We will sometimes refer to such a mapping f as an **embedding** of G in H. In our simulation each computation of the program P at a vertex (i.e. computer) x in G will be replaced by the same computation at the corresponding vertex $f(x)$ in H. Also, each communication of P between adjacent vertices x_1 and x_2 of G will be replaced by the same communication between the corresponding vertices $f(x_1)$ and $f(x_2)$ in H.

Here is the place where we can measure the "effectiveness" of the map f. Notice that whereas x_1 and x_2 could communicate in some unit time t because

they were adjacent, the vertices $f(x_1)$ and $f(x_2)$ require communication time $d \cdot t$ where d is the distance in H between $f(x_1)$ and $f(x_2)$. The simulation thus introduces a time delay factor d in the simulation of P, and the delay could even be worse if there is another adjacent pair z_1 and z_2 of G for which the distance in H between $f(z_1)$ and $f(z_2)$ is bigger than the distance between $f(x_1)$ and $f(x_2)$. We see therefore that one criterion for the effectiveness of f is that the worst possible delay factor (as measured over all possible adjacent pairs of vertices x_1 and x_2 in G) is minimized.

We now put these ideas into mathematical form. First we discuss distance. Throughout this chapter, if x and y are two vertices of G, then a **path** between x and y is a succession of vertices starting from x and ending at y in which every two successive vertices are joined by an edge, *and no vertex is repeated.* The **length** of a path is the number of edges on it (which is one less than the number of vertices on it). Now there may be many different paths of differing lengths between two given vertices in a graph. We define the **distance** between x and y in G, written $\text{dist}_G(x, y)$, to be the length of the shortest path in G from x to y.

Example 1 Find paths of lengths $2, 3, 4, 5$, and 6 between x and y in the graph of Figure 1.

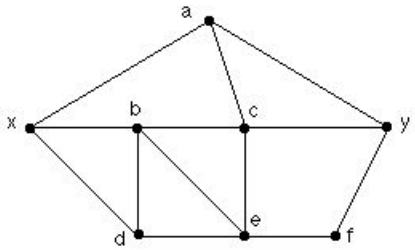


Figure 1. Paths of various lengths between x and y.

Solution: Examples of these paths, in order of length, are

$$x, a, y \qquad x, b, c, y \qquad x, b, e, c, y \qquad x, b, e, c, a, y \qquad x, d, b, e, c, a, y.$$

There are other paths not listed here. Since the shortest path between x and y has length 2, we have $\text{dist}_G(x, y) = 2$. □

We can now describe precisely the worst possible time delay of a map $f : V(G) \to V(H)$. For any two adjacent vertices x and y of G, the **delay** at the edge $\{x, y\}$ caused by f is the distance between $f(x)$ and $f(y)$ in H; that is, $\text{dist}_H(f(x), f(y))$. The worst possible delay, over all edges $\{x, y\}$ in G, caused by f is called the **dilation** of f and is denoted $\text{dil}(f)$. Put more precisely,

$$\text{dil}(f) = \max\{\text{dist}_H(f(x), f(y)) : \{x, y\} \in E(G)\}.$$

Example 2 Suppose that G is the graph obtained from K_4 by removing an edge (this graph is denoted $K_4 - e$) and H is the path on 4 vertices. In Figure 2 we illustrate two different maps from the vertices of G to the vertices of H, one having dilation 2 and the other having dilation 3. We have $\text{dil}(f) = 3$ because the edge $\{a, c\}$ of G is sent to the pair of vertices $f(a)$ and $f(c)$ in H which are distance 3 apart in H, and all other edges of G are sent to pairs that are at distance 2 or less apart. But we have $\text{dil}(g) = 2$, since any edge of G is sent by g to a pair of vertices in H that are distance 2 or less in H. □

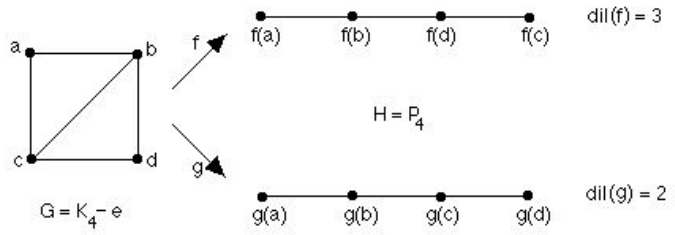


Figure 2. Two different maps $f, g : V(G) \to V(H)$ where $G = K_4 - e$ and H is the path on four vertices. The second map is optimal, so $B(G, H) = 2$.

Finally, we want to find a map f having minimum worst possible time delay; that is, a map with **minimum possible dilation**. We call this minimum $B(G, H)$; or more precisely,

$$B(G, H) = \min\{\text{dil}(f) : f \text{ a one}-\text{to}-\text{one map from } V(G) \text{ to } V(H)\}.$$

We see then that $B(G, H)$ measures the minimum possible worst case time delay when H simulates G, and so in some intuitive sense it measures how "compatible" G and H are. We call the map f **optimal** if $\text{dil}(f) = B(G, H)$, that is, if it has minimum possible dilation over all possible one-to-one maps from $V(G)$ to $V(H)$.

Example 3 What is $B(G, H)$ for the graphs G and H in Figure 2?

Solution: We could try all 4! maps from the vertices of G to the vertices of H, recording the dilation of each map as we go, and at the end we could scan our list to find the minimum dilation. This minimum is $B(G, H)$. We will not carry out this procedure here, but instead note the following. In Figure 2 we found a map $g : V(G) \to V(H)$ satisfying $\text{dil}(g) = 2$. This shows that $B(G, H) \leq 2$. In fact, we can also show that $B(G, H) \geq 2$ by proving that any map $M : V(G) \to V(H)$ must satisfy $\text{dil}(M) \geq 2$. We leave this to the reader as Exercise 2. Thus, $B(G, H) = 2$. □

In the next two sections we will study the function $B(G, H)$ for certain types of graphs G and H that arise in various applications.

Bandwidth

The first of our minimum dilation problems arose originally in the context of sparse matrices. We call a matrix A **sparse** if the number of nonzero entries it has is small in comparison with its total number of entries. Such matrices arise frequently as coefficient matrices of systems of equations or systems of differential equations in numerical analysis and physics. When performing various operations on these matrices, such as matrix multiplication and matrix inversion, we notice that a large number of our computations involve multiplying 0s together. Even storing these matrices in a computer by recording every entry means storing a large number of 0s. In fact, such an $n \times n$ matrix would require storage of all its n^2 entries, most of them being 0. This suggests that we could perform our operations and our storage more efficiently if we concentrated on the nonzero entries, "filling in" the others in a predictably simple way depending on the operation at hand.

The focus on the nonzero entries of a matrix A is made easier if all these entries are concentrated in a "band" consisting of a small number of diagonals of A above and below the main diagonal. Such a concentration would speed up matrix multiplication and Gaussian elimination by making it possible to carry out only a small fraction of all the computations involved and still get the desired result, since the remaining computations (the ones involving entries outside the band) involve all 0 entries and thus have predictable results. A small band for A also allows us to use relatively little memory in storing A since we could keep track of each nonzero entry by simply recording the diagonal to which it belongs and where along that diagonal it can be found.

All this sounds promising, but what do we do if the nonzero entries of the matrix A do not all lie in a small band about the main diagonal of A? The key idea here is to permute the rows and columns of A, hoping to obtain a matrix that does have the required small band. That is, we perform a permutation (i.e. a renumbering) of the columns of A, and also the *same* permutation of the rows of A. Such an identical permutation of rows and columns is called a **symmetric permutation**. (In the case when A is the coefficient matrix of a system of equations, this permutation of the rows is the same as reordering the equations, and the permutation of the columns is the same as reindexing the variables in the system.) If the resulting matrix A' has a smaller band enclosing all its nonzero entries than does A, then we would prefer to work with A' rather than with A. The results of our work on A' can be easily translated back to results on A. (In the context of coefficient matrices, the translation just amounts to reindexing the variables again so that they have their original names.)

Example 4 In the 4×4 matrix at the top of Figure 3, only three super- and subdiagonals are needed to enclose the 1s. Now, by interchanging columns 3 and 4, and also interchanging rows 3 and 4, we obtain the bottom matrix in which only two super- and subdiagonals are needed to enclose all the 1s. This interchange of rows and columns amounts to interchanging the names (or subscripts) of variables 3 and 4 in the corresponding system of equations. □

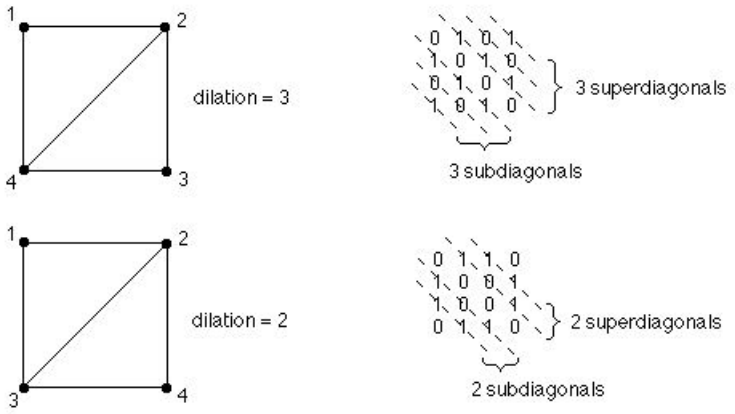


Figure 3. Two numberings of a graph and the corresponding matrices.

What does all this have to do with dilation? The intriguing answer is that the smallest possible band achievable (over all row and column permutations) for a given matrix A is equal to $B(G, H)$ for certain graphs G and H which are defined in terms of A.

To understand this connection between bands of matrices and dilation, we start by defining the path graph on n vertices.

Definition 1 The *path graph* P_n is the graph whose vertices are the integers $1, 2, \ldots, n$ and whose edges are the successive pairs $\{i, i+1\}$, $1 \leq i \leq n-1$. □

Figure 2 shows the path graph P_4.

Suppose G is a graph on n vertices. Then any one-to-one map $f : V(G) \to V(P_n)$ may be viewed as a numbering (or labeling) of the vertices of G by the integers $1, 2, \ldots, n$, and $\mathrm{dil}(f)$ is then the maximum distance between any two integers $f(x)$ and $f(y)$ for which $\{x, y\}$ is an edge of G. Then $B(G, P_n)$ is just the minimum possible $\mathrm{dil}(f)$ over all numberings of G with the integers $\{1, 2, \ldots, n\}$. The function $B(G, P_n)$ (of a graph G) has been extensively studied (see [5] or [3] for surveys), where it is called the *bandwidth* of G. We will abbreviate $B(G, P_n)$ by $B(G)$. We summarize the meaning of bandwidth in the following definition.

Definition 2 The *bandwidth* $B(G)$ of a graph G on n vertices is the minimum possible dilation of any numbering of G with the integers $1, 2, \ldots, n$. □

Example 5 Example 3 shows that $B(K_4 - e) = 2$. □

Now suppose we are given an $n \times n$ symmetric 0-1 matrix $A = [a_{ij}]$. Define a graph $G(A)$ by letting $V(G(A)) = \{1, 2, \ldots, n\}$ and saying that $\{i, j\}$ is an edge if and only if $a_{ij} = 1$. In Figure 3, referred to earlier, we see the 4×4 matrix A and the corresponding graph $G(A) = K_4 - e$. Notice that each possible numbering f of $G(A)$ for a matrix A corresponds to a symmetric permutation $P(f)$ of the rows and columns of A. Also, the dilation of any numbering f of $G(A)$ corresponds to the number of superdiagonals above and subdiagonals below the main diagonal containing all the 1s in the matrix resulting from the permutation $P(f)$. Hence $B(G(A))$ is the smallest possible band, over all symmetric permutations of rows and columns of A, of super- and subdiagonals of A which contain all the 1s of A. Finding this smallest band for A, or equivalently finding $B(G(A))$, is (as we said before) important in being able to efficiently store A, and in performing various operations on A such as Gaussian elimination and inversion.

Example 6 Figure 3 also shows the correspondence between the dilation of a numbering and the band of the corresponding matrix. □

Calculating and Bounding Bandwidth

Now that we have defined bandwidth of graphs and understand its relation to matrices, we will calculate the bandwidth of some familiar graphs and find upper and lower bounds for the bandwidth of arbitrary graphs.

We can easily calculate the bandwidths, $B(K_n)$ and $B(C_n)$, of K_n and C_n.

Example 7 What is $B(K_n)$ and $B(C_n)$?

Solution: Clearly $B(K_n) = n - 1$ since no matter what map $f : V(K_n) \to V(P_n)$ we consider, the two vertices mapped to 1 and n are joined by an edge, showing that $\text{dil}(f) = n - 1$. Since this is true for any map f, the smallest possible dilation is $n - 1$ and thus $B(K_n) = n - 1$.

To find $B(C_n)$, we can first show that $B(C_n) > 1$ (this is left to the reader). We can also show that $B(C_n) \leq 2$ by finding a map $f : V(C_n) \to V(P_n)$ with $\text{dil}(f) = 2$. This can be done by numbering the "left" half of C_n with the odd

integers from 1 to n or $n-1$ (depending on whether n is odd or even respectively) in increasing order as we proceed counterclockwise around C_n, and numbering the "right" half with the even integers from 2 to n or $n-1$ (depending on whether n is even or odd respectively) in increasing order proceeding clockwise, starting from the vertex at clockwise distance one from the vertex numbered 1. This is illustrated in Figure 4. Since $1 < B(C_n) \leq 2$, it follows that $B(C_n) = 2$. □

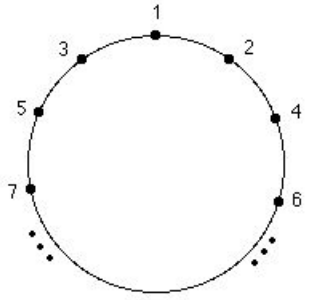


Figure 4. An optimal numbering of C_n showing that $B(C_n) = 2$.

Example 8 What is is the bandwidth $B(K_{m,n})$ of $K_{m,n}$ where $m \leq n$?

Solution: We will show that $B(K_{m,n}) = m - 1 + \lceil n/2 \rceil$, where $m \leq n$. We first show the upper bound $B(K_{m,n}) \leq m-1+\lceil n/2 \rceil$ by constructing a map f for which $\mathrm{dil}(f) = m - 1 + \lceil n/2 \rceil$. Let A and B be the disjoint sets of sizes m and n respectively defining the partition of the vertices of $K_{m,n}$ into two sets. We map $\lfloor n/2 \rfloor$ of the vertices in B to the integers $1, 2, \ldots, \lfloor n/2 \rfloor$, we then map all vertices in A to the integers $\lfloor n/2 \rfloor + 1, \ldots, \lfloor n/2 \rfloor + m$, and finally we map the remaining $\lceil n/2 \rceil$ vertices of B to the integers $\lfloor n/2 \rfloor + m + 1, \ldots, m + n$. With this map we see that the edge of $K_{m,n}$ that is stretched the longest is the one joining the vertex of A mapped to $\lfloor n/2 \rfloor + 1$ to the vertex of B mapped to $m + n$. This stretch is by definition the dilation of this map, and it has length $m - 1 + \lceil n/2 \rceil$.

We now prove the lower bound $B(K_{m,n}) \geq m - 1 + \lceil n/2 \rceil$. Let f be any numbering of $V(K_{m,n})$ by the integers $1, 2, \ldots, m + n$. Let m_1 and M_1 be the minimum and maximum of the set $f(A)$, and let m_2 and M_2 be the minimum and maximum of the set $f(B)$. Now at least one of the inequalities $m_2 \leq M_1$ or $m_1 \leq M_2$ holds, so that at least one of the intervals $[m_2, M_1]$ or $[m_1, M_2]$ is well defined. Then among either the one or the two intervals that are in this way defined, at least one must contain at least half the vertices in the set $f(B)$. This interval by its definition must also contain all the integers in the set $f(A)$. Since the first and last integers of this interval correspond to points on opposite sides of $K_{m,n}$, it follows that $\mathrm{dil}(f)$ is at least the length of this interval. But this length is clearly at least $\lceil |f(B)|/2 \rceil + |f(A)| - 1 = \lceil n/2 \rceil + m - 1$. □

Example 9 Figure 5 shows the optimal numbering of $K_{4,4}$ given by the example. □

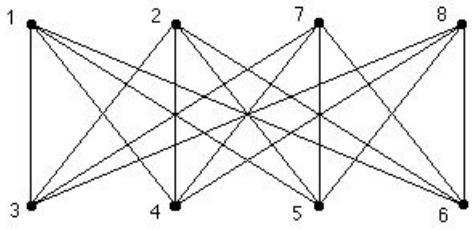


Figure 5. An optimal numbering of $K_{4,4}$.

We will now derive some bounds on $B(G)$ in terms of other graph parameters. Before giving these bounds, we might well ask why we should be interested in such bounds. The reason is that up to now we have not stated a general procedure for computing $B(G)$ for an arbitrary graph G on n vertices. This is because there is no known procedure for doing this other than attempting all $n!$ possible maps $f : V(G) \to V(P_n)$, which is not a very appetizing prospect. In this light we can see that bounds for $B(G)$ which are easy to compute would be very welcome. For many graphs certain graph parameters can be easily computed, and in such cases the bounds on $B(G)$ which we will find are then also easy to compute.

The graph parameters we will use for our bounds are the number of vertices n, the number of edges q, the connectivity κ, the independence number β, and the diameter D.

Definition 3 We define the *connectivity* $\kappa(G)$, for a connected graph G, to be the smallest number of vertices whose removal from G (together with the removal of all edges incident on these vertices) leaves a disconnected graph. (By convention we take $\kappa(K_n) = n - 1$ and $\kappa(G) = 0$ for a disconnected graph G.)

The *independence number* $\beta(G)$ is the maximum number of vertices in any set S of vertices of G with the property that no two vertices in S are joined by an edge. (A set S with this property is called an *independent* set in G.)

The *diameter* $D(G)$ is the maximum distance between any two vertices of G. □

Example 10 For the graph of Figure 1 we have $n = 8$, $q = 13$, $\kappa = 2$, $\beta = 3$, and $D = 3$. We have $\kappa = 2$ because the removal of the set $\{e, y\}$ of size 2 leaves a disconnected graph while the removal of any single vertex leaves the graph still connected.

We have $\beta = 3$ since there is an independent set $\{x, e, y\}$ of size 3 but no independent set of size 4 or greater.

Finally, we have $D = 3$ since the maximum distance between any two vertices is 3 (for example, $\text{dist}(x, y) = 3$). □

The bounds on bandwidth we are looking for are based on two simple observations we will make in the next two lemmas.

Lemma 1 If G is a subgraph of H with the same number of vertices as H, i.e. $|V(G)| = |V(H)|$, then $q(G) \leq q(H)$, $\kappa(G) \leq \kappa(H)$, and $\beta(G) \geq \beta(H)$. ■

The reader should verify this. To see that the hypothesis $|V(G)| = |V(H)|$ is necessary, the reader is invited to find an example of a subgraph G of some graph H with $|V(G)| < |V(H)|$ for which $\kappa(G) > \kappa(H)$. The second observation requires some notation. For any graph H let H^k be the graph having the same vertex set as H in which two vertices are joined by an edge if and only if they are at distance at most k in H.

Example 11 Draw the graph P_5^2.

Solution: The graph P_5^2 is illustrated in Figure 6. We obtain it by starting with P_5 and then joining by an edge every pair of vertices x and y separated by a distance of 2; in other words, joining "every other" vertex on P_5. □

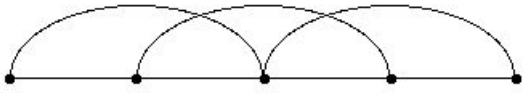


Figure 6. The graph P_5^2.

Lemma 2 Let G be a graph on n vertices. Then $B(G) \leq k$ if and only if G is a subgraph of P_n^k.

Proof: Recall that if the bandwidth of G satisfies $B(G) \leq k$, then the vertices of G can be numbered with the integers 1 through n in such a way that for any edge $\{x, y\}$ the numbers given to x and y differ by at most k. Therefore this numbering is an embedding of G as a subgraph of P_n^k.

To establish the converse, we observe that an embedding f of G as a subgraph of P_n^k can also be viewed as a numbering satisfying $\text{dil}(f) \leq k$. Hence $B(G) \leq k$. ■

We are now ready to state our bounds on bandwidth.

Theorem 1 Let G be a graph having n vertices, q edges, connectivity κ, independence number β, and diameter D. Then

(i) $B(G) \geq n - \frac{1}{2}(1 + ((2n-1)^2 - 8q)^{1/2})$.

(ii) $B(G) \geq \kappa$.

(iii) $B(G) \geq \frac{n}{\beta} - 1$.

(iv) $B(G) \geq \frac{n-1}{D}$.

Proof: We first note that $\kappa(P_n^k) = k$, $\beta(P_n^k) = \lceil n/k \rceil$, and $|E(P_n^k)| = \frac{1}{2}k(2n - k - 1)$. We leave proofs of these facts as Exercise 4. Now suppose $B(G) = k$.

To prove (i) we use Lemmas 1 and 2, obtaining

$$q \leq |E(P_n^k)| = \frac{1}{2}k(2n - k - 1).$$

Solving the quadratic inequality in k we obtain (i).

To prove (ii) we have similarly

$$\kappa(G) \leq \kappa(P_n^k) = k = B(G).$$

To prove (iii) we use Lemma 1 (as applied to β, with P_n^k playing the role of H) to get $\beta(G) \geq \beta(P_n^k)$. Exercise 4 asks the reader to prove that $\beta(P_n^k) = \lceil n/k \rceil$ and the right hand side is at least n/k. Hence (iii) follows.

To prove (iv), consider any numbering f of G with the integers $1, 2, \ldots, n$. Let $x = f^{-1}(1)$ and $y = f^{-1}(n)$. Clearly there is a path $x = x_0, x_1, x_2, \ldots, x_t = y$ in G from x to y of length $t \leq D$. The image of this path under f starts at 1 and ends at n. By the pigeonhole principle there must be an i such that

$$|f(x_i) - f(x_{i-1})| \geq \frac{n-1}{t} \geq \frac{n-1}{D}.$$

Thus $\mathrm{dil}(f) \geq (n-1)/D$, and part (iv) follows. ■

We remark that the simple lower bound in part (iv) of the previous theorem actually gives us the exact bandwidth in some important classes of graphs, as described in Theorem 2, which follows. Let T_k be the complete binary tree of k levels. Thus, T_k has a root at level 1, and 2^{i-1} vertices at level i for $i \leq k$.

Theorem 2 The bandwidth of T_k $(k \geq 1)$ is given by

$$B(T_k) = \left\lceil \frac{2^{k-1} - 1}{k-1} \right\rceil.$$

Proof: Observe first that T_k has $2^k - 1$ vertices, and has diameter $2k - 2$. Hence $B(T_k) \geq \left\lceil \frac{2^{k-1}-1}{k-1} \right\rceil$ by part (iv) of Theorem 1. To prove that equality holds we can construct a numbering of T_k with dilation $\left\lceil \frac{2^{k-1}-1}{k-1} \right\rceil$. We leave this construction as a challenging problem for the reader. ■

Example 12 Figure 7 shows a numbering of T_4 with dilation 3. This numbering is optimal by Theorem 2. □

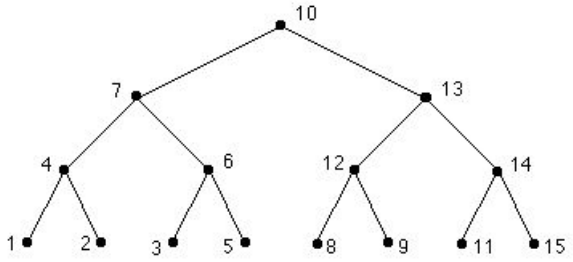


Figure 7. An optimal numbering of T_4.

A little bit of thought gives a stronger lower bound than (iv) which is nonetheless based on the same idea. Define the **density** of a connected graph H to be

$$\text{den}(H) = \max\left\{ \left\lceil \frac{|V(G)| - 1}{D(G)} \right\rceil : G \text{ a connected subgraph of } H \right\}$$

where $D(G)$ denotes the diameter of a graph G.

Example 13 Calculate the density of the graph C shown in Figure 8.

Solution: Consider the subgraph G of C consisting of vertex 7 together with all its neighbors. For this G we have $\left\lceil \frac{|V(G)|-1}{D(G)} \right\rceil = 4$, and no other subgraph has a larger such ratio. Therefore $\text{den}(C) = 4$. □

To get a feeling for $\text{den}(H)$, notice first that for any subgraph G of H the ratio $\left\lceil \frac{|V(G)|-1}{D(G)} \right\rceil$ measures how tightly packed or "dense" G is in the sense of packing in a number of vertices within a given diameter. Thus, although H may be dense in some parts and less dense in others, $\text{den}(H)$ measures the densest that H can be anywhere. Notice that when G is a subgraph of H we have $B(G) \leq B(H)$, while $B(G)$ is itself bounded below as in part (iv) of Theorem 1. We can in fact show the following.

Lemma 3 For any graph H, we have $B(H) \geq \text{den}(H)$. ■

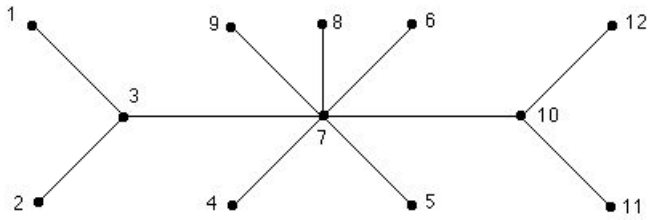


Figure 8. A caterpillar C, and an optimal numbering of it.

It is now interesting to see that just as the trivial lower bound (d) is the actual value of the bandwidth for a natural class of graphs, the more general lower bound of Lemma 3 is the actual value of bandwidth for an additional (and also natural) class of graphs. Recall that a tree T is called a **caterpillar** if it contains a path such that every vertex not on the path is adjacent to some vertex on the path.

Theorem 3 For any caterpillar C we have $B(C) = \text{den}(C)$. ■

We omit the proof here, though one can be found in [11] or [1] as a consequence of an algorithm for computing the bandwidth of any caterpillar.

Example 14 A caterpillar C, together with an optimal numbering of C, is shown in Figure 8. We saw in Example 13 that $\text{den}(C) = 4$. By Lemma 3 we have $B(C) \geq 4$, while the numbering given in the figure has dilation 4. It follows that $B(C) = 4 = \text{den}(C)$ (as claimed by Theorem 3). □

Before leaving bandwidth, we mention a widely used algorithm for approximating $B(G)$ for arbitrary graphs G. It is called a "level algorithm" since it numbers the vertices of G by levels. Specifically, we begin by choosing a root v of G. Now let S_i be the set of vertices in G at distance i from v (we think of S_i as being "level i" of G). Now the algorithm lets $f(v) = 1$. Then it numbers the neighbors of v consecutively using the numbers 2 through $|S_1| + 1$, and the vertices at distance 2 from v consecutively using $|S_1| + 2$ through $|S_1| + |S_2| + 1$, etc. In general, the algorithm numbers the vertices in any given level with consecutive integers, with level i coming before level j if $i < j$. In other words, the algorithm maps S_i to the integers $|\cup_{t \leq i-1} S_t| + 2$ through $|\cup_{t \leq i} S_t| + 1$. Note that the resulting dilation is at most twice the size of the largest level S_i since edges of G can only run between successive levels.

Example 15 Figure 9 illustrates a numbering of a graph produced by the level algorithm. We see that there are five levels, and the vertices within each level are numbered consecutively. □

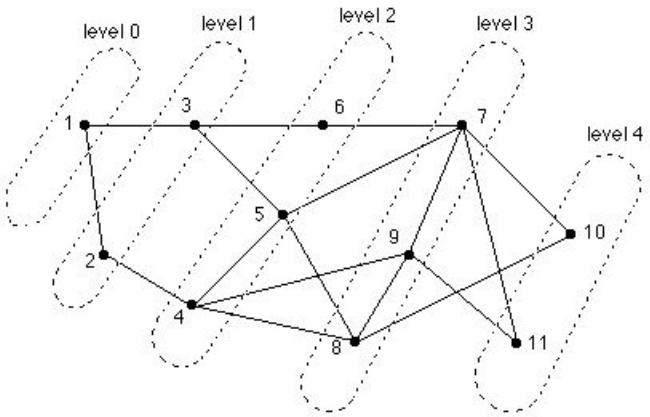


Figure 9. A numbering by levels.

How good is this algorithm ? That is, how close to the actual bandwidth is the dilation of the numbering produced by the level algorithm ? The following example (taken from [10]) shows that this algorithm performs badly on at least some examples. First we need some notation (similar to the "big O") on growth of functions.

Definition 4 For two functions $f(n)$ and $g(n)$, we write

$$f(n) = \Omega(g(n))$$

if $f(n) \geq Kg(n)$ for some constant K when n is sufficiently large. We also write

$$f(n) = \Theta(g(n))$$

to signify that $f(n) = \Omega(g(n))$ *and* $f(n) = O(g(n))$. □

We see then that while $f = O(g(n))$ means that f is bounded above by a constant times g for n sufficiently large, the notation $f(n) = \Omega(g(n))$ means that f is bounded below by a constant times g for n sufficiently large and $f(n) = \Theta(g(n))$ means that f is bounded both above and below by constants times g (the constant for the upper bound being usually different than the constant for the lower bound).

Example 16 For each integer $n \geq 1$ we construct a tree $L(n)$ as follows. Start with a path P_{2^n} on 2^n vertices. Now attach a path on 2^{n-1} vertices to the 2^{n-1}st vertex of P from the left. Then attach a path of 2^{n-2} vertices to the $2^{n-1} + 2^{n-2}$st vertex of P from the left, a path of 2^{n-3} vertices to the $2^{n-1} + 2^{n-2} + 2^{n-3}$st vertex of P from the left, etc. The tree $L(4)$ is illustrated in Figure 10. Now choose the leftmost vertex of P, call it v, as the root of $L(n)$ for the level algorithm. Observe that there are $n+1$ vertices at distance $2^n - 1$ from v, and n vertices at distance $2^n - 2$ from v; that is, $|S_{2^n-1}| = n + 1$ and $|S_{2^n-2}| = n$. Also, every vertex of S_{2^n-1} is joined to some vertex of S_{2^n-2} by an edge. Hence any level algorithm with v as root will produce a numbering f of $L(n)$ such that

$$\text{dil}(f) \geq |S_{2^n-1}| = n + 1 = \Omega(\log|L(n)|).$$

On the other hand it can be shown that $B(L(n)) = 2$ for all n. Thus there is a gap between $B(L(n)) = 2$ and the estimate $\Omega(\log|L(n)|)$ for $B(L(n))$ proposed by the level algorithm that grows without bound as n approaches infinity. □

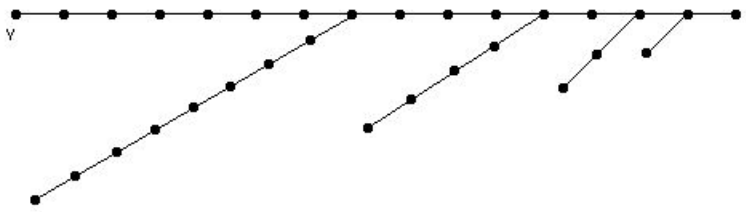


Figure 10. The tree $L(4)$. A level algorithm with v as root performs poorly.

Dilation Involving Other Pairs of Graphs

Let us now consider briefly the parameter $B(G, H)$ for graphs H other than just the path. Because cycles occur so frequently in graph theory we will first look at $B(C_n, H)$ where H is any connected graph on n vertices. Along the way we will find an interesting way of computing an upper bound on $B(C_n, T)$ where T is any tree on n vertices. Also because of its importance in circuit layout design on computer chips, we will study $B(G, G_2)$ where G is any graph and G_2 is the two-dimensional grid graph (to be defined later). Our first result is perhaps surprising because of its constant upper bound applicable to a large class of graphs H.

Theorem 4 Let H be any connected graph on n vertices. Then $B(C_n, H) \leq 3$ where C_n denotes the cycle on n vertices. ■

This result is found in [12]. We will sketch here the construction behind the proof because of its interesting algorithmic nature. First, however, we need to understand a connection between $B(Q, G_1)$ and $B(Q, G_2)$ when G_1 and G_2 are two graphs related in a special way, and Q is some third graph.

Lemma 4 Suppose that G_1 is a subgraph of a graph G_2 having the same number of vertices as G_2. If Q is a third graph with $|V(Q)| \leq |V(G_1)|$, then $B(Q, G_2) \leq B(Q, G_1)$.

Proof: Any map $f : Q \to G_1$ can also be viewed as a map $f' : Q \to G_2$, and we have $\text{dil}(f') \leq \text{dil}(f)$ since G_2 has all the edges which G_1 has, and possibly more. Now if we take $f : Q \to G_1$ to be a map with smallest possible dilation $B(Q, G_1)$, then since $B(Q, G_2)$ is the smallest possible dilation of any map from Q to G_2 (possibly even smaller than $\text{dil}(f')$) we get

$$B(Q, G_2) \leq \text{dil}(f') \leq \text{dil}(f) = B(Q, G_1).$$ ■

Now we know that any connected graph H on n vertices has a spanning tree; that is, a subgraph T which is a tree on n vertices. If we apply Lemma 4 with C_n, H, and T playing the roles of Q, G_2, and G_1 respectively, then we get $B(C_n, H) \leq B(C_n, T)$. Therefore, if we could show that $B(C_n, T) \leq 3$ for any tree on n vertices, then Theorem 4 would be proved. This amounts to showing that there is a map $f : C_n \to T$ such that $\text{dil}(f) \leq 3$.

The following is an algorithm for constructing an embedding $f : C_n \to T$ into any tree T on n vertices satisfying $\text{dil}(f) \leq 3$. We denote the vertices of C_n by $1, 2, \ldots, n$ indexed in cyclic order. Given a root vertex $r \in T$ we let

$$\text{level}(x) = \text{dist}_T(r, x) \text{ for any } x \in T.$$

We will also refer to a depth first search of T as a DFS of T (see Section 10.4 of *Discrete Mathematics and Its Applications*).

We first describe this algorithm informally. We imagine walking through the vertices of T in the order of a DFS, and constructing our map f as we go. Suppose that we have so far mapped the first $i-1$ vertices $\{1, 2, \ldots, i-1\}$ of C_n to T, and we are now located at the vertex $f(i-1)$ of T. Our task is to decide how to map vertex i; that is, what vertex of T should be chosen as $f(i)$. We call a vertex of T "used" if it is $f(t)$ for some t where $1 \leq t \leq i-1$; that is, if it has been used as an image of the partially constructed map f. The algorithm moves to the first vertex of T following $f(i-1)$ (in the DFS) which is unused — call this vertex v. We must now decide whether to use v as $f(i)$. How do we make this decision? The answer depends on whether $\text{level}(v)$ is even or odd. If $\text{level}(v)$ is even then we use v; that is, we let $f(i) = v$. If $\text{level}(v)$ is odd, skip v and continue walking on the DFS to an unused child of v, if such a child exists, and repeat the procedure. If all the children of v have been used, then use v by letting $f(i) = v$.

ALGORITHM 1. Cycle-Tree.

procedure *cycle-tree*(T: a tree on n vertices)
choose a vertex $v_1 \in T$ as a root of T
$f(1) := v_1$
$S_1 := \{v_1\}$
order the vertices of T according to a DFS of T starting at the root v_1
for $i := 2$ **to** n
begin
{We assume that $f(1), f(2), \ldots, f(i-1)$ have been defined, with values $v_1, v_2, \ldots, v_{i-1}$, respectively, and that $S_{i-1} = \{v_1, v_2, \ldots, v_{i-1}\}$.}
$z :=$ the first vertex following v_{i-1} in the DFS such that $z \notin S_{i-1}$
if level(z) is even **then** $v_i := z$
else {level(z) is odd}
if all children of z lie in S_{i-1} **then** $v_i := z$
else $v_i :=$ the first child of z in the order of the DFS that is not in S_{i-1}
$f(i) := v_i$
$S_i := S_{i-1} \cup \{v_i\}$
end
{The output is $f(1) = v_1, f(2) = v_2, \ldots, f(n) = v_n$.}

Algorithm 1 gives this algorithm in pseudocode. The input will be a tree T on n vertices and its output will be a one-to-one map $f : C_n \to T$ satisfying $\text{dil}(f) \leq 3$.

Example 17 Figure 11 shows an embedding $f : C_{15} \to T_4$ of C_{15} into the level 4 complete binary tree produced by Algorithm Cycle-Tree such that $\text{dil}(f) = 3$. It is instructive to go through the steps the algorithm follows to produce this embedding. □

We omit the proof that the embedding $f : C_n \to T$ produced by Algorithm Cycle-Tree indeed satisfies $\text{dil}(f) \leq 3$. The details can be found in [12].

We now pass to a brief look at $B(T_r, G_2)$, where T_r is the level r complete binary tree, and G_2 is the infinite two-dimensional grid.

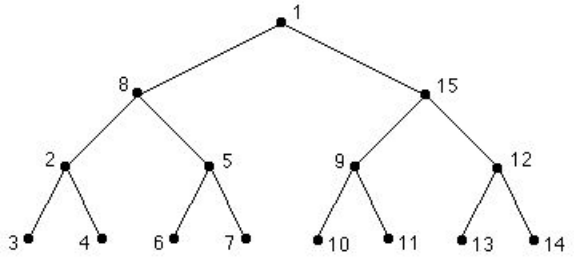


Figure 11. An embedding $F : C_{15} \to T_4$ produced by algorithm Cycle-Tree.

Definition 5 The *two dimensional grid* G_2 is the graph with vertices

$$V(G_2) = \{(x, y) : x \text{ and } y \text{ are integers}\}$$

and edges

$$E(G_2) = \{\{(x_1, y_1), (x_2, y_2)\} : |x_1 - x_2| + |y_1 - y_2| = 1\}. \qquad \square$$

Thus, the vertices of G_2 are the lattice points in the plane (i.e., points with integer coefficients), and two lattice points are joined by an edge in G_2 when they are unit distance apart.

The grid graph G_2 is an especially important graph in applications. One of the commonly used interconnection networks, called the *mesh*, has the structure of a finite rectangle in G_2. (That is, the mesh is just an $m \times n$ subgrid of G_2 for some m and n.) Also, G_2 is useful in analyzing circuit layout on computer chips — but more on this later. For these reasons it is important to estimate $B(H, G_2)$ for various graphs H as a way of finding the communication delay when G_2 simulates H. Although this problem is difficult, we at least have a good lower bound for $B(T_r, G_2)$. We include the proof of this bound here because it is based on a very simple geometrical idea.

Theorem 5 The minimum possible dilation of any mapping of a complete binary tree on n vertices into the two-dimensional grid is bounded below by a constant times $\sqrt{n}/\log n$ when n is sufficiently large.

That is, we have

$$B(T_r, G_2) = \Omega(\sqrt{n}/\log n)$$

where $n = |T_r|$.

Proof: Let $f : T_r \to G_2$ be a one-to-one map, and let $d = \text{dil}(f)$. Also, let z be the root of T_r, that is, the vertex of degree 2 in T_r. Since every vertex of T_r is within distance $r - 1$ of z, it follows that every vertex in $f(T_r)$ can be found

within distance (as measured in G_2) $d(r-1)$ of $f(z)$, and hence within a circle or radius $d(r-1)$ in the plane centered at $f(z)$. That is, we are saying that the entire image $f(T_r)$ must be contained in a circle C of radius $d(r-1)$ in the plane centered at $f(z)$. Now this image consists of $|V(T_r)| = 2^r - 1$ lattice points, so C must contain at least $2^r - 1$ lattice points since it contains this image. But the number of lattice points contained within a circle in the plane is proportional to the area of that circle. Since the area of C is $\pi d^2(r-1)^2$, it follows that $\pi d^2(r-1)^2 \geq K(2^r - 1)$, for some constant K. Solving for d we get

$$d \geq \left(\frac{K(2^r-1}{\pi(r-1)^2}\right)^{1/2} \geq L\frac{2^{r/2}}{r},$$

where L is some other constant (for example $L = K/(2\pi)$ will do). Now since $n = 2^r - 1$, we see that $2^{r/2}$ and r are proportional to $\sqrt{n}$ and $\log n$ (to the base 2) respectively. Therefore, we get

$$d = \Omega\left(\frac{2^{r/2}}{r}\right) = \Omega\left(\frac{\sqrt{n}}{\log n}\right),$$

and the theorem follows. ■

It is remarkable that in fact this lower bound is, up to a constant factor, also an upper bound! That is, one can prove that $B(T_r, G_2) = O(\frac{\sqrt{n}}{\log n})$. Thus, the function $\sqrt{n}/\log n$ is, up to a constant factor, a correct estimate for $B(T_r, G_2)$. The proof of this, though requiring no special knowledge beyond the elements of graph theory, is still more complicated than the proof of Theorem 5. The interested reader may find it in [14], pp. 89–91.

Remembering that in applications we would only use a finite $m \times n$ subgrid of G_2 as an interconnection network, the reader may well wonder if one could get as good an upper bound for $B(T_r, H)$ as for $B(T_r, G_2)$ if H were a particularly small finite subgrid of G_2, say with just enough points to accommodate T_r. Intuitively, one would expect that being "hemmed in" by H would make it harder to find embeddings with small dilation than when we had as much of G_2 to work with as we wanted. We will return to this subject when we discuss the "area" of an embedding later. For now we try to be precise about relating the size of the target graph to the size of the domain graph in a graph embedding.

Definition 6 Consider a graph embedding $f : V(G) \to V(H)$. The *expansion* of f is $\exp(f) = |H|/|G|$. □

Example 18 In Figure 12 the graph G on the left has 4 vertices while the graph H on the right has 7 vertices. Hence the map $f : V(G) \to V(H)$ satisfies $\exp(f) = 7/4$. □

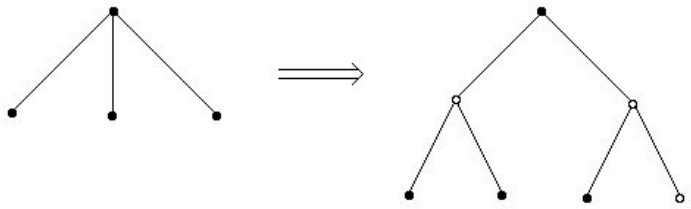


Figure 12. Embedding complete ternary trees into complete binary trees.

Example 19 An example in which one extreme of this tradeoff can be easily seen is the embedding of complete ternary trees (i.e. 3-ary trees) into complete binary trees. Let $T(3)_r$ be the complete ternary tree with r levels, and thus $|T(3)_r| = (3^r - 1)/2$. The basic idea leading to an embedding $f : T(3)_r \to T_{2r}$ with $\mathrm{dil}(f) = 2$ is indicated in Figure 12. The intended pattern is to embed level i of $T(3)_r$ into level $2i - 1$ of T_{2r} in the indicated manner for all $i \geq 1$. The fact that $\mathrm{dil}(f) = 2$ is then easy to see. But what is $\exp(f)$?

Since T_{2r} has $2^{2r} - 1$ vertices, when we form the required ratio we get

$$\exp(f) = 2\left(\frac{2^{2r}-1}{3^r-1}\right) = \Theta\left(\frac{2^{2r}-1}{3^r-1}\right) = \Theta\left(\left(\frac{4}{3}\right)^r\right).$$

Now, letting $n = |T(3)_r|$, we should try to express the last quantity on the right as a function of n. One way to do this is to ask ourselves what power, call it α, we must raise n to in order to get this last quantity. We can answer this by working backwards. Writing $n^\alpha = \Theta((\frac{4}{3})^r)$ and substituting $(3^r - 1)/2$ for n, we get $(3^r)^\alpha = \Theta((\frac{4}{3})^r)$ (having absorbed the factor of 2 in our Θ). Now taking the rth root of both sides, we can solve for α by taking logarithms and the result is $\alpha = \log_3(4/3)$. In conclusion, we have

$$\exp(f) = \Theta(n^\alpha),$$

where $\alpha = \log^3(4/3)$. □

Thus, in order to achieve a constant dilation of 2, we have paid the heavy price of expanding by a factor that grows as a fixed positive power of n. On the other hand, what price in dilation would we pay if we insisted on constant expansion? The fascinating answer is provided in [8] where it is shown that any embedding of a complete ternary tree into a complete binary tree with expansion less than 2 must have dilation $\Omega(\log\log\log n)$. The proof of this is beyond the scope of this chapter, but the interested reader is referred to [8] to see what is involved.

Min Sum and Cutwidth

In applications where a circuit (or graph) G is for automation purposes mapped on a path, the wires joining vertices of G must be placed on channels or tracks running parallel to the path. No two wires are allowed to overlap on the same track. Since the number of different tracks needed for the mapping essentially determines the area of the circuit layout (where the area is the product of the number of tracks and the length of the path), we would like to minimize the number of tracks. One way of doing this is to use each track "as much as possible". This amounts to constructing our map f in such a way that the edges of G, when drawn between points on the image path, overlap as little as possible.

We now express these ideas more precisely. Start with a one-to-one map $f : V(G) \to V(P_n)$ from G to the path on n vertices. For each interval $(i, i+1)$ of the "host" graph P_n we let $\text{cut}(i)$ be the number of "guest" edges from G which pass over that interval; that is, we let

$$\text{cut}(i) = |\{\{f(x), f(y)\} : \{x, y\} \in E(G), f(x) \leq i \text{ and } f(y) \geq i + 1\}|.$$

Now we let $\text{value}(f)$ be the maximum of $\text{cut}(i)$ over all $1 \leq i \leq n-1$. Thus, $\text{value}(f)$ is the biggest overlap of edges that we have over any interval when we use the map f. Finally, we define the **cutwidth** of G, which is denoted $c(G)$, to be the smallest possible biggest overlap, taken over all possible maps f; that is,

$$c(G) = \min\{\text{value}(f) : f : V(G) \to V(P_n) \text{ a one-to-one map}\}.$$

Thus, $c(G)$ is proportional to the smallest area possible in a linear layout of G subject to the constraint that no two wires overlap in the same channel.

Example 20 In Figure 13 we illustrate a graph G with five vertices and two different maps from G to P_5. The second map has the smaller value; in fact it is easy to see that no map can have a smaller value; that is, $c(G) = 3$. □

Another mapping problem which we will discuss follows immediately from the idea of dilation. When studying the dilation of a map $f : V(G) \to V(H)$ we are finding the worst possible time delay, over all edges in G, caused by f; that is, we are finding $\text{dil}(f)$. It is also natural to consider the *average* time delay caused by f, since although $\text{dil}(f)$ might be large it could happen that f might still be a "good" map if it has small time delay on a large fraction of all the edges of G. We would naturally calculate the average by adding up the individual delays and dividing by the total number of edges in G, and then we could ask for the smallest possible average over all possible maps f.

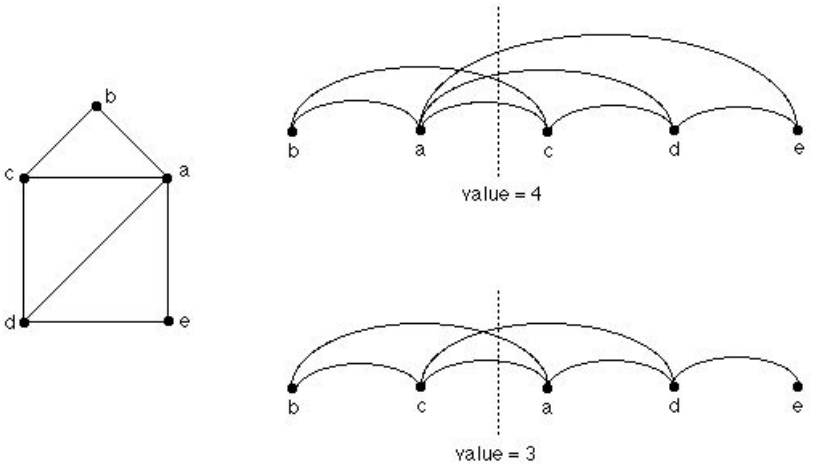


Figure 13. Two embeddings; the second is cutwidth optimal.

We now state all this precisely. Again we start with a one-to-one map $f : V(G) \to V(H)$. Define

$$\text{sum}(f) = \sum_{\{x,y\} \in E(G)} \text{dist}_H(f(x), f(y)),$$

which is the sum of the individual delays. We won't bother here to divide by the number of edges in G to get the average delay, since this number does not depend on the map f and we could do the division later if we care to. Now the minimum average we are looking for (apart from delaying the division until later) is

$$S(G, H) = \min\{\text{sum}(f) : f{:}\, V(G) \to V(H) \text{ a one-to-one map}\}.$$

Thus we may think of $S(G, H)$ as being the best average dilation (or time delay) obtainable over all embeddings $f : G \to H$.

Example 21 Find sum(f) and sum(g) where f and g are the dilation 3 and dilation 2 numberings respectively of $K_4 - e$ shown in Figure 3.

Solution: The number on a vertex of $K_4 - e$ indicates the vertex of P_4 to which it is mapped. Thus, for each edge $\{x, y\}$ of $K_4 - e$ the term $\text{dist}_H(f(x), f(y))$, with $H = P_4$, is just the difference between the numbers given to x and y. So, to calculate sum(f) or sum(g) we just need to add these differences over all edges in $K_4 - e$. Carrying out this addition we get sum(f) = 8 and sum(g) = 7. □

Some elementary results on the min sum and cutwidth problems are summarized in the following examples. We abbreviate $S(G, P_n)$ by $S(G)$.

Example 22 What are $S(P_n)$, $S(C_n)$, $S(K_{1,n})$, and $S(K_n)$?

Solution: We have $S(P_n) = n - 1$, $S(C_n) = 2(n - 1)$, $S(K_{1,n}) = \lfloor n^2/4 \rfloor$, and $S(K_n) = n(n^2 - 1)$. (See Exercise 9.) □

Example 23 What are $c(P_n)$, $c(C_n)$, $c(K_{1,n})$, and $c(K_n)$?

Solution: We have $c(P_n) = 1$, $c(C_n) = 2$, $c(K_{1,n}) = \lfloor n/2 \rfloor$, and $c(K_n) = \lfloor n^2/4 \rfloor$. (See Exercise 11.) □

Some values for S and c are difficult to prove. Values of $S(T_k)$ and $c(T_{r,k})$ (where $T_{r,k}$ is the complete k-level r-ary tree) have been computed. See [4] and [9]. Although the computation of both $c(G)$ and $S(G)$ is in general difficult, polynomial time algorithms have been developed for computing $c(T)$ and $S(T)$ when T is a tree (see [15], [7], [6]). These results are well outside the scope of this chapter, but the reader is encouraged to study them in order to gain an appreciation for algorithms in graph theory.

The analogue of Theorem 4 for min sum is the following.

Theorem 6 [12] Let H be any connected graph on n vertices. Then

$$\frac{S(C_n, H)}{|E(H)|} \leq 2 - \frac{2}{n}.$$

The map $f : C_n \to H$ which provides the upper bound for $S(C_n, H)$ of Theorem 6 is the one produced in the algorithm Cycle-Tree (given earlier) applied to any spanning tree of H.

Area

In this section we discuss the area of a graph embedding into G_2, the 2-dimensional grid. Results in this subject have obvious applications to the layout of circuits on computer chips, these chips being wafers with vertical and horizontal tracks etched into them along which connections between circuit elements must run. Thus, we let H be an arbitrary graph, and we consider a one-to-one map $f : H \to G_2$. To make our analysis realistic, we apply generally accepted assumptions on how wires run along the tracks of the chip. These assumptions

constitute the so called "Thompson grid model" [13]. Specifically, we view f not only as a map of vertices, but also as a map of the edges of H to paths in G_2, these paths of course running along the vertical and horizontal tracks of G_2. We also require that distinct edges e_1 and e_2 of H have images $f(e_1)$ and $f(e_2)$ which are not allowed to run along the same track (vertical or horizontal) of G_2 for any distance, although they may cross at a point when one image is running horizontally while the other is running vertically. A map satisfying these conditions is often called a *circuit layout* (with H being the electronic circuit).

Definition 7 The *area* $A(f)$ of an embedding $f : H \to G_2$ is the product of the number of rows and the number of columns of G_2 which contain any part of the layout $f(H)$. □

Example 24 In Figure 14 we illustrate an embedding $f : K_4 - e \to G_2$ with area 6 (because we use 2 rows and 3 columns) and dilation 2, and an embedding $f : K_4 \to G_2$ with area 15 (because we use 3 rows and 5 columns) and dilation 8. Each unit segment in G_2 that is part of the layout has been labeled with the edge of the graph that runs along it. Notice that no two edges in the layout run along the same segment for any distance, though they may cross at the intersection points of the segments.

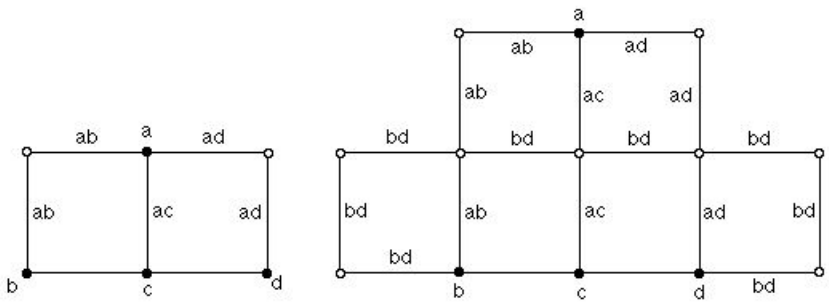


Figure 14. Embeddings of $K_4 - e$ and K_4 in the grid G_2.

In chip manufacturing we are concerned with producing chips having as small an area as possible. Therefore, given a graph H, we are interested in finding a map $f : H \to G_2$ for which $A(f)$ is as small as possible.

In order to get a feeling for what is involved in minimizing area, we will consider here in some detail the relatively simple case when H is a complete binary tree T_{2k+1} on $2k + 1$ levels and height $2k$. We use an embedding called the *H-tree layout* because it follows a recursive pattern based on the letter H. We illustrate H-tree layouts of T_3 and T_5 in Figure 15. In general our goal is to embed T_{2k+1} into a square S in G_2 of dimensions $(2^{k+1} - 1) \times (2^{k+1} - 1)$.

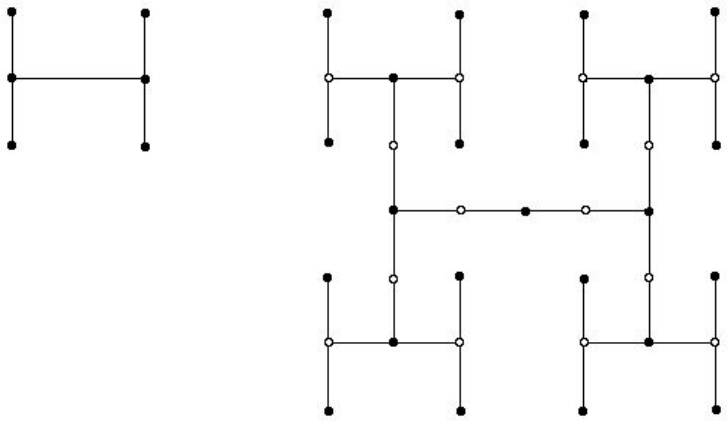


Figure 15. H**-tree layouts of** T_3 **and** T_5**.**

An informal description of how this is done is as follows. We map the degree 2 node z of T_{2k+1} to the middle of S. We now use the horizontal and vertical tracks of G_2 which meet at z to separate S into four quadrants, each quadrant being a square of dimensions $(2^k-1)\times(2^k-1)$. Now recursively map the four subtrees isomorphic to T_{2k-1} and rooted at the grandchildren of z to these four quadrants. Finally use one of the two tracks separating the quadrants as part of the T_3 which interconnects z, its children, and its grandchildren.

From this recursive description of the H-tree layout, and with Figure 15 as an aid, we can establish the following which gives a good embedding of T_{2k+1} into G_2 from the standpoint of both area and dilation.

Theorem 7 There is an H-tree layout $f : T_{2k+1} \to G_2$ with the following properties. Let $n = |T_{2k+1}|$.

(i) $f(T_{2k+1})$ is contained in a square of dimensions $(2^{k+1}-1)\times(2^{k+1}-1)$.

(ii) $A(f) = O(n)$.

(iii) $\mathrm{dil}(f) = O(\sqrt{n}\,)$.

Proof: For (i) we observe that a side of S must have length one more than twice a side of the square containing the H-tree layout of T_{2k-1}. Since the latter has side length 2^k-1 by induction, it follows that S has side length $2(2^k-1)+1 = 2^{k+1}-1$, as claimed. For (ii) we note that the area is $A(f) = O(2^{2k})$ while $n = 2^{2k+1}-1$. Hence it is easy to see that $A(f) = O(n)$. For (iii), we can check that the longest image of an edge is the one joining z to one of its children. The path in S to which this edge is mapped has length one more than half the side length of one of the quadrants. Since this side length is $2^{k+1}-1$, we get $\mathrm{dil}(f) = 1 + O(2^k) = O(\sqrt{n}\,)$. ■

It is natural to ask how well the H-tree layout does, using area and dilation as yardsticks. Clearly the area of $O(n)$ achieved is optimal up to a constant factor since any n-vertex graph must use $\Omega(n)$ area. As for dilation, we already know from Theorem 5 that any embedding $f : T_{2k+1} \rightarrow G_2$ (even without the constraints of the Thompson model) satisfies $\mathrm{dil}(f) = \Omega(\sqrt{n}/\log n)$. Hence the dilation $O(\sqrt{n}\,)$ achieved by the H-tree layout is at most a factor of $\log n$ from the smallest possible that the dilation could be.

Finally, what about graphs other than complete binary trees? It can be shown that any n-vertex binary tree can be laid out in area $O(n)$, and any n-vertex planar graph can be laid out in area $O(n \log^2 n)$. These results, and more general ones applying to any class of graphs having "$f(n)$ separators" for some suitable function f, are outside the scope of this chapter. The reader is referred to [14] for a good exposition, and to [2] for some recent results.

Suggested Readings

1. S. Assman, G. Peck, M.Syslo, and M.Zak, "The bandwidth of caterpillars with hairs of length 1 and 2", *SIAM J. Alg. Discrete Methods*, 1981, pp. 387–391.

2. S. Bhatt and F. Leighton, "A framework for solving VLSI graph layout problems", *J. of Computer and System Sciences*, Vol. 28, No. 2, 1984, pp. 300–343.

3. P. Chinn, J. Chvatalova, A. Dewdney, and N. Gibbs, "The bandwidth problem for graphs and matrices — a survey", *J. of Graph Theory*, Vol. 6, 1982, pp. 223–254.

4. F. Chung, "A conjectured minimum valuation tree", Problems and Solutions in *SIAM Review*, Vol. 20, 1978, pp. 601–604.

5. F. Chung, "Labelings of Graphs", in *Selected Topics in Graph Theory 3*, eds. L. Beineke and R. Wilson, Academic Press Ltd., London, 1988, pp. 151–168.

6. F. Chung, "On optimal linear arrangements of trees", *Computers and Mathematics with Applications*, Vol. 10, 1984, pp. 43–60.

7. M. Goldberg and I. Klipker, "Minimal placing of a line" (in Russian), Technical Report, Phsico-Technical Institute of Low Temperatures, Academy of Sciences of Ukrainian SSR, USSR, 1976.

8. J. Hong, K. Melhorn, and A. Rosenberg, "Cost trade-offs in graph embeddings, with applications", *Journal of the ACM*, Vol. 30, No. 4, 1983, pp.

709–728.

9. T. Lengauer, "Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees", *SIAM J. Alg. Discrete Methods*, Vol. 3, 1982, pp. 99–113.

10. F. Makedon and I. Sudborough, "Graph layout problems", *Surveys in Computer Science*, ed. H. Maurer, Bibliographisches Institut, Zurich, 1984, pp. 145–192.

11. Z. Miller, "The bandwidth of caterpillar graphs", *Congressus Numerantium*, Vol. 33, 1981, pp. 235–252.

12. A. Rosenberg and L. Snyder, "Bounds on the cost of data encodings", *Mathematical Systems Theory*, Vol. 12, 1978, pp. 9–39.

13. C. Thompson, "Area-time complexity for VLSI", *Eleventh Annual ACM Symposium on Theory of Computing*, 1979.

14. J. Ullman, *Computational Aspects of VLSI*, Computer Science Press, Rockville, Md., 1984.

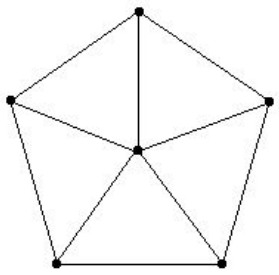
15. M. Yannakakis, "A polynomial algorithm for the min cut linear arrangement of trees", *Journal of the ACM*, Vol. 32, 1985, pp. 950–959.

Exercises

1. Show that if a graph G has a vertex of degree k, then $B(G) \geq \lceil k/2 \rceil$.

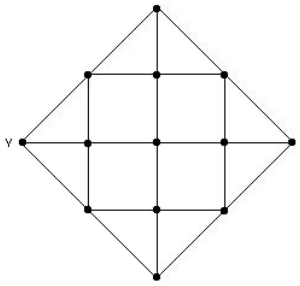
2. Show that $B(K_4 - e) \geq 2$.

3. Find the bandwidth of the following graph.



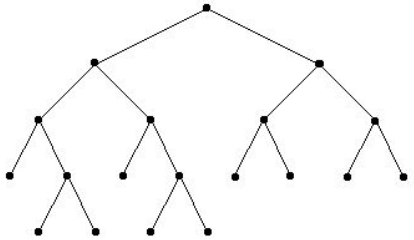
4. Show that
a) $\kappa(P_n^k) = k$. b) $\beta(P_n^k) = \lceil n/k \rceil$.
c) $|E(P_n^k)| = \frac{1}{2}k(2n - k - 1)$.

5. Consider the following graph G. If f is a numbering of G obtained by a level algorithm with the indicated vertex v as the root, then what is the smallest that $\mathrm{dil}(f)$ can be? What is the largest that $\mathrm{dil}(f)$ can be for such an f?



⋆**6.** Construct a sequence of graphs $\{H(n): n = 1, 2, 3, \ldots\}$ for which the estimate for $B(H(n))$ produced by the level algorithm differs from the true value of $B(H(n))$ by an amount that grows without bound as n approaches infinity no matter which vertex in $H(n)$ is chosen as a root. *Hint*: Use the graphs $L(n)$ described in the text by pasting them together somehow.

7. Use Algorithm Cycle-Tree to construct a dilation 3 map $f : C_{19} \to T$, where T is the tree in the following figure.



8. Construct layouts of K_5 and K_6 into G_2 (obeying the assumptions of the Thompson grid model) with the smallest area you can manage.

9. Show that
a) $S(P_n) = n - 1$. b) $S(C_n) = 2(n - 1)$.
c) $S(K_{1,n}) = \lfloor n^2/4 \rfloor$. d) $S(K_n) = n(n^2 - 1)$.

10. Show that if a graph G has a vertex of degree k, then $c(G) \geq \lceil k/2 \rceil$.

11. Show that
a) $c(P_n) = 1$. b) $c(C_n) = 2$.
c) $c(K_{1,n}) = \lfloor n/2 \rfloor$. d) $c(K_n) = \lfloor n^2/4 \rfloor$.

12. Consider the matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix},$$

which has a band of 5 diagonals above and below the main diagonal which enclose all the 1s.

a) Find a row and column permutation (the same permutation for both rows and columns) of A which results in a matrix A' requiring a band of only 4 such diagonals.

b) Find the graph $G(A)$, and show that $B(G(A)) = 4$.

13. Show that if a graph G on n vertices has a vertex of degree k with k even, then $S(G) \geq \frac{k}{2}(k+2)$.

14. Show that if the level algorithm is applied to T_k (the complete binary tree on k levels with $2^k - 1$ vertices) with the degree 2 vertex as the root, then the biggest possible dilation that could result is $2^k - 2^{k-2} - 1$. What is the smallest value?

Computer Projects

1. Write a program which takes a tree T and produces a numbering of T with as small a dilation as you can manage. *Note*: Do not try to write a program which will calculate the exact smallest possible dilation numbering. There are theoretical reasons why such a program is likely to take much too long to run when the number of vertices in T is large. Instead of trying to write such a program, just develop some sensible heuristic idea and make it the foundation of your program.

2. Write a program which takes a tree T and produces a numbering f of T for which $\text{sum}(f)$ is as small as you can manage.

3. Write a program which takes a tree T and produces a numbering f of T for which $\text{value}(f)$ is as small as you can manage.

Note: In Projects 2 and 3, there are programs to calculate the smallest possible $\text{sum}(f)$ and $\text{value}(f)$ which are reasonably time efficient, but developing such a program is quite ambitious. Again, try to develop some sensible heuristic.