Theoretical Computer Science 58 (1988) 209-229
North-Holland

MIN CUT IS NP-COMPLETE FOR EDGE WEIGHTED TREES

B. MONIEN*
Fachbereich Mathematik und Informatik, Universität Paderborn, 4790 Paderborn, Fed. Rep. Germany

I. H. SUDBOROUGH**
Computer Science Department, University of Texas at Dallas, Richardson, TX 75083, U.S.A.

Abstract. We show that the Min Cut Linear Arrangement Problem (Min Cut) is NP-complete for trees with polynomial size edge weights and derive from this the NP-completeness of Min Cut for planar graphs with maximum vertex degree 3. This is used to show the NP-completeness of Search Number, Vertex Separation, Progressive Black/White Pebble Demand, and Topological Bandwidth for planar graphs with maximum vertex degree 3.

1. Introduction

The *Min Cut Linear Arrangement problem* (Min Cut) asks, for a given finite undirected graph G and positive integer k, if there exists an arrangement of G's vertices along a horizontal line so that, for any vertical line drawn between consecutive vertices, dividing the set of vertices into those to its left and those to its right, there are at most k edges connecting vertices on opposite sides. It is motivated by problems of circuit design in which one wants to minimize the number of channels needed for wires connecting distinct circuit boards or gates. (A *linear layout* of a graph G is a one-to-one mapping from the vertices of G to the natural numbers, describing how vertices are to be laid out along a horizontal line.) The *cutwidth of a linear layout L of a graph G*, denoted by $cw(G, L)$ is the maximum number of edges connecting vertices on opposite sides of any vertical line drawn between consecutive vertices. The *cutwidth of G*, denoted by $\mathrm{cw}(G)$, is minimum $\{\mathrm{cw}(G, L) \mid G$ is a linear layout of $G\}$. A related problem the *Weighted Min Cut problem*, asks, for a given finite undirected graph with integer edge weights and a positive integer k, whether there is a linear arrangement so that each vertical line drawn between consecutive vertices cuts edges (in the sense described above) whose weights sum to at most k. The weighted version may more closely model actual circuit design problems, as connections in practice may be of different sizes or may consist of bundles of wires from board to board or from gate to gate. Due to the importance

* Supported in part by NATO research grant 850183.
** Supported in part by a grant from Texas Instruments, Inc.

of the Min Cut problem there is interest in the question: For what classes of graphs can one construct a polynomial-time Min Cut algorithm?

In [25] Yannakakis described an $O(n \log n)$ algorithm for the Min Cut problem on trees. Yannakakis described a generalized version of the Min Cut problem in which nodes have variable heights indicated by integer weights. His algorithm works for this generalization, but not for the Weighted Min Cut problem on trees. Earlier an $O(n \log^{d-2} n)$ algorithm for the Min Cut problem on trees with maximum vertex degree d was described by Chung, Makedon, Sudborough, and Turner [3], who also described a characterization theorem for cutwidth in trees.

In this paper, we show:

(1) the Min Cut problem is NP-complete even when restricted to planar graphs with maximum vertex degree 3 and

(2) the weighted Min Cut problem is NP-complete even when restricted to trees with polynomial size weights.

We also solve an open problem about the complexity of the search number problem described in a recent paper by Megiddo, Hakimi, Garey, Johnson, and Papadimitriou [18] and several related problems.

Informally, searching an undirected graph G is the process of capturing alien intruders using searchers to clear all the edges and vertices. The intruders are mobile and can speed about the edges and vertices of the graph with complete knowledge of the whereabouts of their pursuers. A *search sequence* is a sequence of steps of the following types:

(1) placing a searcher on a vertex,

(2) moving a searcher from a vertex through an edge to a neighboring vertex, and

(3) removing a searcher from a vertex.

An edge of G is *cleared* when there is no possibility of an intruder on that edge. An edge $e = \{x, y\}$ becomes cleared through the process of either having a searcher on one end, say x, and moving another searcher through the edge to the other end, y, or, when all other edges incident to x are already cleared, moving the one searcher on vertex x through the edge e to y. An uncleared edge is *contaminated.* In the beginning all edges are assumed to be contaminated, as they are capable of holding an intruder. A cleared edge e can become *recontaminated* by the movement or deletion of a searcher which results in a path without any searchers from a contaminated edge to e. A search sequence that does not allow any recontamination is called *progressive.* LaPaugh [12] has shown that if there is a search sequence that clears all the edges of a graph with k searchers, then there is also a progressive search sequence that uses k searchers and clears all edges of the graph. That is, allowing recontamination does not help to reduce the number of searchers.

The *search Number problem* asks, for a given finite undirected graph G and positive integer k, whether k searchers are sufficient to clear all the edges and vertices of G [20, 21]. Megiddo, Hakimi, Garey, Johnson, and Papdimitriou [18] describe a proof that the Search Number problem is NP-hard. By showing that recontamination does not help, [12] Andrea LaPaugh showed that Search Number is, in fact, in NP. In [18] it is stated as open problems whether Search Number is NP-complete for planar graphs and graphs with maximum vertex degree 3. We show Search Number is NP-complete for planar graphs with maximum vertex degree 3.

In fact, some of our results rely on earlier results of Makedon and Sudborough [16], who show Search Number is identical to Min Cut for graphs with maximum vertex degree 3. Makedon, Papadimitriou, and Sudborough [17] also use this relationship to obtain a weaker result: Search Number is NP-complete for the class of graphs with maximum vertex degree 3.

The main subject of [17] is *topological bandwidth.* The *bandwidth of a linear layout L of a graph G*, denoted by $\mathrm{bw}(G, L)$, is the maximum length of any edge in the layout, i.e., $\max\{|L(x)-L(y)| \mid \{x, y\}$ is an edge of $G\}$. (It is to be understood that the vertices are always assigned to integer points in the line.) The *bandwidth of a graph G*, denoted by $\mathrm{b}(G)$, is minimum $\{\mathrm{b}(G, L) \mid L$ is a linear layout of $G\}$. The *Bandwidth Minimization Problem* (BMP) is the problem of deciding, for a given finite undirected graph G and positive integer k, whether the bandwidth of G is at most k. Papadimitriou [27] showed BMP to be NP-complete. Garey, Graham, Johnson, and Knuth [7] showed BMP to be NP-complete for trees with maximum vertex degree 3, and Monien [26] showed that BMP is NP-complete even for trees with maximum vertex degree 3 and with search number 2, i.e. the class of trees often called caterpillars. The *topological bandwidth of a graph G*, denoted by $\mathrm{tb}(G)$, is minimum$\{\mathrm{b}(G') \mid G'$ is a homeomorphic image of $G\}$. (G' is a *homeomorphic image of G* if it is obtained from G by adding some number of degree-2 vertices into edges of G.) The *Topological Bandwidth problem* asks, for a given finite undirected graph G and positive integer k, whether $\mathrm{tb}(G) \leq k$. Topological Bandwidth has also been considered recently by Miller [19], Chung [2] and others. We show that the Topological Bandwidth problem is NP-complete even for planar graphs with maximum vertex degree 3.

Another graph layout problem has been investigated recently by Lengauer [13], Kirousis and Papadimitriou [10], and Ellis, Sudborough, and Turner [5]. The *vertex separation of a linear layout L of a graph G*, denoted by $\mathrm{vs}(G, L)$, is the maximum, over all integers i ($1 \leq i \leq |G|$), of the number of vertices in $\mathrm{left}(i)=\{x$ in vertices$(G) \mid L(x) \leq i\}$ that need to be deleted from G in order to separate (disconnect) all vertices in $\mathrm{left}(i)$ from those in $\mathrm{right}(i)=\{x$ in vertices$(G) \mid L(x) > i\}$. The *vertex separation of a graph G*, denoted by $\mathrm{vs}(G)$, is minimum$\{\mathrm{vs}(G, L) \mid L$ is a linear layout of $G\}$. The problem is motivated by the general problem of finding good separators for graphs, which has been used in algorithm and VLSI layout design [14]. The *Vertex Separation problem* asks, for a given finite undirected graph G and positive integer k, whether the vertex separation of G is at most k. It has recently been shown to be NP-complete [13]. We show that the Vertex Separation problem is NP-complete even for planar graphs with maximum vertex degree 3.

The *modified cutwidth of a linear layout L of G*, denoted by $\mathrm{mcw}(G, L)$, is the maximum, over all integers i ($1 \leq i \leq |\text{vertices}(G)|$) of the number of edges connecting vertices in $\mathrm{left}(i)-\{L^{-1}(i)\}$ with those in $\mathrm{right}(i)$. (That is, it is the maximum number of edges cut by any vertical line drawn through a vertex dividing the remaining vertices into those to its left and those to its right.) The *modified cutwidth of a graph*

G, denoted by $\text{mcw}(G)$, is $\text{minimum}\{\text{mcw}(G, L) \mid L$ is a linear layout of $G\}$. The *Modified Min Cut problem* asks, for a given finite undirected graph G and positive integer k, whether $\text{mcw}(G) \leq k$. Lengauer [13] observed that the Modified Min Cut problem is also NP-complete and related it to the Vertex Separation problem. We show that the Modified Min Cut problem is NP-complete even for planar graphs with maximum vertex degree 3.

The number of registers needed to perform a computation, i.e., the space needed for a computational process, is often considered in papers through studies of various pebble games played on directed acyclic graphs (dag's). For example, Hopcroft, Paul, and Valiant [9] show that, for every positive integer k, dags with n vertices in which every vertex has at most k predecessors, can be pebbled with $O(n/\log n)$ pebbles. The rules of the pebble game are simple:

(1) a (black) pebble can be removed from a vertex at any time and
(2) a (black) pebble can be placed on a vertex provided that all of its predecessors have pebbles.

The placing of a (black) pebble on a vertex corresponds to computing the value represented by the vertex and storing it in memory. The predecessors represent values that are needed in the computation and must have been previously computed and stored in memory. Gilbert, Lengauer, and Tarjan [8] show that the *Pebble Demand problem* is PSPACE-complete. The Pebble Demand problem asks, for a given finite dag G and positive integer k, whether k pebbles are sufficient to pebble a sink of G. (A *sink* of G is a node in G with no successors.) One is permitted to save pebbles in the pebble game by *recomputation*, i.e., placing a pebble on a vertex, removing the pebble, and placing a pebble again on the same vertex. A sequence of pebble game steps is *progressive* if no vertex is pebbled more than once, i.e., there is no recomputation. The *Progressive Pebble Demand problem* asks, for a finite dag G and positive integer k, whether k pebbles are sufficient to pebble a sink of G by a progressive sequence of steps in the pebble game. Sethi [23] has shown the *Progressive Pebble Demand problem* to be NP-complete.

A nondeterministic version of the pebble game involving both black and white pebbles was introduced by Cook and Sethi [4]. Placing a white pebble on a node represents guessing the value corresponding to that node. The rules for the placement and removal of white pebbles are the duals of the rules for black pebbles:

(3) a white pebble can be placed on a vertex at any time, and
(4) a white pebble can be removed from a vertex provided that all predecessors have a pebble (either black or white).

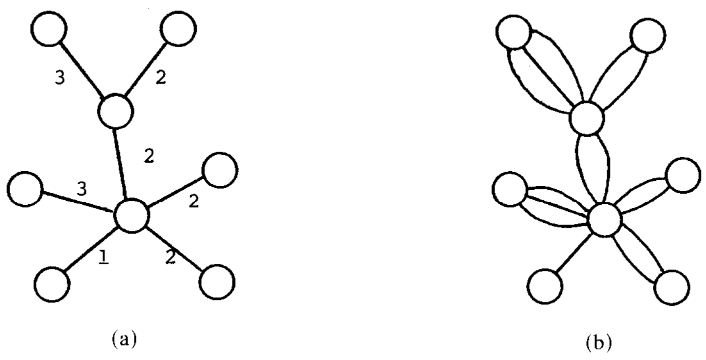
The rules for playing the black/white pebble game are all of the rules (1)-(4). The relationship between nondeterministic and deterministic pebble games has been the subject of many recent papers, for example, see Meyer auf der Heide [15], Wilber [24], Klawe [11] and Rosenberg and Sudborough [22]. The *Black/White Pebble Demand problem* asks, for a given finite dag G and positive integer k, whether k pebbles are sufficient to pebble a sink of G in the black/white pebble game. The complexity of the *Black/White Pebble Demand problem* is currently open. It is easily seen to be in PSPACE, but it not known to be PSPACE hard. The progressive version of the black/white pebble game is defined in the same way as for the regular (black) pebble game. Lengauer [13] showed that the Progressive Black/White Pebble demand problem is NP-complete. We extend this by showing that the progressive Black/White Pebble Demand problem is NP-complete even when restricted to planar dag's with maximum vertex degree 3. (The degree of a vertex in a dag is taken to be the sum of the number of its predecessors and successors.)

2. Description of results

We show first that the Weighted Min Cut problem restricted to trees with polynomial-size edge weights is reducible (by a polynomial-time algorithm) to the Min Cut problem for planar graphs. Given a weighted tree $T=(V, E)$, construct a planar graph $G(T)$, which has the same set of vertices as T, and whose edge set is obtained by replacing a weight-k edge $\{x, y\}$ of T with k parallel edges in $G(T)$ connecting x and y. Clearly, the cutwidth of T and $G(T)$ are the same. In fact, a linear layout of the one is also a linear layout of the other with the same cutwidth. $G(T)$ is not a simple graph since it has parallel edges, but it can easily be transformed into a simple graph with the same cutwidth. Namely, construct the simple graph, say $S(T)$, by placing a degree-2 vertex into each constructed parallel edge of $G(T)$. Since the addition or removal of degree-2 vertices does not change cutwidth, T and $S(T)$ have the same cutwidth. Moreover, $S(T)$ is planar since it was obtained from a tree by the introduction of parallel edges. Consequently, (T, k) is a positive instance of the Weighted Min Cut problem if and only if $(S(T), k)$ is a positive instance of the Min Cut problem for planar graphs. Notice it is necessary that the edge weights of T have polynomial size, for otherwise there would be more than a polynomial number of parallel edges in $G(T)$ and the transformation from T to $G(T)$, and hence also to $S(T)$, would not be possible in polynomial time. Thus, we have the following lemma.

Lemma 2.1. *The Weighted Min Cut Problem restricted to trees with polynomial-size edge weights* $\leqslant_{\text{poly}}$ *the Min Cut problem restricted to planar graphs.*

Observe that the planar graph $S(T)$, described in the last paragraph, is a very special kind of planar graph. It is a 2-outerplanar graph, where a *1-outerplanar graph* is simply an outerplanar graph and, for all $k>1$, a *k-outerplanar graph* is a planar graph that has embedding in the plane such that if one deletes all the vertices and incident edges on one face, say the external face, the result is a $(k-1)$-outerplanar graph [1]. In fact, $S(T)$ has a planar embedding such that if all vertices and incident edges on the external face are deleted, the result is a collection of isolated vertices. Moreover, $G(T)$ is outerplanar in the class of multigraphs, i.e., the graphs in which multiple parallel edges are allowed. An example of the reduction described in Lemma 2.1 is shown in Fig. 1.



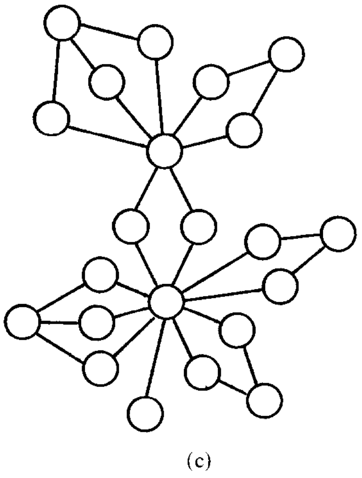


Fig. 1. (a) An edge weighted tree T. (b) The constructed planar graph $G(T)$. (c) The constructed simple planar graph $S(T)$.

Corollary 2.2. *The Weighted Min Cut Problem restricted to trees with polynomial size edge weights* $\leq_{\text{poly}}$ *the Min Cut problem restricted to 2-outerplanar graphs. (Moreover, the Weighted Min Cut problem restricted to trees with polynomial-size edge weights is polynomial-time reducible to the Min Cut problem for outerplanar multigraphs.)*

We will show that the Weighted Min Cut problem restricted to trees with polynomial-size edge weights is NP-complete in Lemma 2.4 by a polynomial-time reduction from a restricted version of the following so-called 0-1 solvability problem for linear integer equations.

0-1 Solvability Problem for Linear Integer Equations (denoted by 0, 1-LinEq).
Input: A rectangular matrix A and a column vector b, both of nonnegative integers.
Question: Does there exist a 0-1 column vector x such that $Ax = b$?

0, 1-LinEq is easily seen to be NP-complete even when all entries in the matrix A and all entries in b are from $\{0, 1\}$. For example, a straightforward reduction from ONE-IN-THREE 3SAT is given by the following. Given a well-formed formula w in 3CNF, create a column vector b of $m+n$ 1's and a matrix A with $m+n$ rows and $2m$ columns, where m is the number of variables and n is the number of clauses occurring in w. Intuitively, for all i $(1 \leqslant i \leqslant m)$, the $(2i-1)$st column corresponds to the ith variable in w and the $2i$th column corresponds to the negation of this variable. For each i $(1 \leqslant i \leqslant m)$, let row i of A have a 1 in columns $2i-1$ and $2i$ (and only in those two columns). Also, for each i $(1 \leqslant i \leqslant n)$, let row $m+i$ have a 1 in exactly those three columns corresponding to literals that occur in the ith clause of w. Then, due to the first m rows of A, it follows that, for all i $(1 \leqslant i \leqslant m)$, exactly one of the $(2i-1)$st or $2i$th entries of a solution vector x is 1, corresponding to assigning the value true to either the ith variable or its negation. And, due to the last n rows of A, it follows that this solution vector x must give each clause exactly one true literal. Thus it follows that w is in ONE-IN-THREE 3SAT if and only if there is a solution vector x over $\{0, 1\}$ such that $Ax = b$.

We use a restricted version of 0, 1-LinEq, where A is a matrix of positive integers with m rows and n columns and b is a column vector of length m such that

(1) A has an even number of columns, i.e., n is even,

(2) any 0-1 vector x such that $Ax = b$ must have $\frac{1}{2}n$ 0 values,

(3) the sum of the integers in each row of A is exactly twice the value of the integer in the corresponding row of the column vector b,

(4) the values in the first column and second column of each row of the matrix A are equal and greater than all other values in that row.

(5) for all i, (a): if the sum of r values from the ith row of A is b_i, then $r = \frac{1}{2}n$, and (b): the sum of the smallest value in the ith row of A and the sum of any $\frac{1}{2}n - 1$ values in the $(i+1)$st row of A is larger than b_{i+1} (thus, for example, all elements in the ith row are larger than all elements in the $(i+1)$st row),

(6) the first value of the vector b, namely b_1, is more than twice as big as the second value of the vector b, namely b_2.

Lemma 2.3. *The* 0-1 *solvability problem for linear integer equations restricted to instances which satisfy conditions* (1)-(6) *listed above is* NP-*complete in the strong sense, i.e., even when the integers in the matrix A have size bounded by a polynomial in the size of A.*

Proof. Note that the matrix A produced in the reduction from ONE-IN-THREE 3SAT, as described above, already satisfies conditions (1) and (2). We need only show that such an instance (A, b) of 0, 1-LinEq can be transformed into an instance (A', b') satisfying all six conditions such that (A, b) is a positive instance of 0, 1LinEq if and only if (A', b') is a positive instance of 0, 1-LinEq. We describe a sequence of transformations to accomplish this.

Let (A, b) be an instance of 0, 1-LinEq satisfying conditions (1) and (2). For each i, let k_i denote the sum of the elements in the ith row of A. Form A' by adding two new columns to A, where these two new columns have values $2k_i - b_i$ and $k_i + b_i$ respectively in the ith row for all i. Let b' be the column vector having $2k_i$ in its ith row. Observe that (A, b) is a positive instance of 0, 1-LinEq if and only if (A', b') is a positive instance of 0, 1-LinEq. (That is, if one can select elements in the ith row of A whose sum is b_i, then one can select the same elements in the ith row of A' together with the new element $2k_i - b_i$ to sum up to $2k_i$. Conversely, if there are elements in the ith row of A' whose sum is $2k_i$, then the remaining elements in the ith row of A' must also have the sum $2k_i$ since the sum of all ith row elements is $4k_i$. Since $2k_i - b_i$ must be included in one of these sums and $k_i + b_i$ in the other, it follows that there are elements from the ith row of A whose sum is b_i.) Notice that (A', b') satisfies conditions (1)–(3) above. It satisfies (3) since in each row the sum of all elements is $4k_i$ and b'_i is $2k_i$.

Let (A, b) be an instance of 0, 1-LinEq satisfying (1)–(3). Form the new matrix A' by adding two new columns to A, where the new columns have in the ith row, for all i, the value $2b_i$. That is, A' is the matrix with the new columns as columns 1 and 2 and the remaining columns exactly the same as in the matrix A. Let b' be the column vector with $3b_i$ in its ith row. Then, (A, b) is a positive instance of 0, 1-LinEq if and only if (A', b') is a positive instance of 0, 1-LinEq and (A', b') satisfies conditions (1)–(4).

Let (A, b) be an instance of 0, 1-LinEq satisfying conditions (1)–(4), where A has m rows and n columns. Let $M = 1 + \max\{b_i \mid 1 \leq i \leq m\}$. Define the matrix A' by $a'_{i,j} = a_{i,j} + (m - i + 1) \cdot M$, for all i. Define the column vector b' by $b'_i = b_i + \frac{1}{2}n \cdot (m - i + 1) \cdot M$. It is straightforward to verify that (A, b) is a positive instance of 0, 1-LinEq if and only if (A', b') is a positive instance of 0, 1-LinEq. For any i,

$$\begin{aligned} & a'_{i,j(1)} + a'_{i,j(2)} + \cdots + a'_{i,j(r)} = b'_i \\ \Rightarrow\ & r \cdot (m - i + 1) \cdot M + a_{i,j(1)} + a_{i,j(2)} + \cdots + a_{i,j(r)} \\ & \quad = b_i + \tfrac{1}{2}n \cdot (m - i + 1) \cdot M \\ \Rightarrow\ & a_{i,j(1)} + a_{i,j(2)} + \cdots + a_{i,j(r)} - b_i \\ & \quad = [\tfrac{1}{2}n - r] \cdot (m - i + 1) \cdot M \\ \Rightarrow\ & |a_{i,j(1)} + a_{i,j(2)} + \cdots + a_{i,j(r)} - b_i| \\ & \quad = |[\tfrac{1}{2}n - r] \cdot (m - i + 1) \cdot M|. \end{aligned}$$

By condition (3), the sum of all elements in the ith row is at most $2 \cdot b_i$, so the left side of the equation is at most b_i. Therefore, $b_i \geq |[(n/2) - r] \cdot (m - i + 1) \cdot M|$. Consequently, as $M > b_1$ and $m - i + 1 > 0$, $\frac{1}{2}n - r$ must be 0. So, $r = \frac{1}{2}n$. Furthermore, the sum of any $\frac{1}{2}n - 1$ elements from row i and an element of row $i - 1$ is greater than

$$\begin{aligned} [\tfrac{1}{2}n - 1] \cdot (m - i + 1) \cdot M + (m - i + 2) \cdot M &= \tfrac{1}{2}n \cdot (m - i + 1) \cdot M + M \\ &> (n/2) \cdot (m - i + 1) \cdot M + b_i = b'_i. \end{aligned}$$

So, (A', b') satisfies condition (5). Observe that (A', b') satisfies conditions (1)–(4)

as well. In fact, any solution vector x for the instance (A', b') is also a solution vector for the instance (A, b) and conversely. Note that (A', b') also satisfies conditions (3) and (4), as A' is created from A by adding a fixed value to each element of the ith row of A for all i, and b' is obtained in a similar and appropriate manner.

Let (A, b) be an instance of 0, 1-LinEq satisfying conditions (1)-(5). Create the matrix A' by multiplying each entry in row 1 by 2. Let b' be the column vector obtained from b by replacing the first entry b_1 by $2 \cdot b_1$. It easily follows that the result (A', b') satisfies (1)-(6) and is a positive instance of 0, 1-LinEq if and only if (A, b) is a positive instance of 0, 1-LinEq.

The reader should observe that the maximum size of any integer in the constructed matrix A is $38m+24$. That is, if we start with the initial 0-1 valued matrix described in the paragraphs directly before the statement of Lemma 2.3 and apply the sequence of transformations indicated, then the maximum size of any element in the resulting m by n matrix A is $38m+24$. So, 0, 1-LinEq restricted to instances that satisfy conditions (1)-(6) is NP-complete in the strong sense, i.e., it is NP-complete even when the integers in the matrix have size bounded by a polynomial in the size of the matrix. □

Lemma 2.4. 0, 1-LinEq *restricted to instances satisfying conditions* (1)-(6) $\leq_{\text{poly}}$ *the Weighted Min Cut problem restricted to trees with polynomial-size edge weights.*

Proof. Let A be the given rectangular matrix, say with m rows and $2n$ columns, and b be the given column vector such that together they form an instance of 0, 1-LinEq satisfying conditions (1)-(6). We construct a tree T with polynomial-size edge weights and an integer k such that there is a 0-1 column vector x such that $Ax = b$ if and only if T has cutwidth k. In fact, the cutwidth bound k is chosen to be the first value of the column vector, namely b_1. The tree T consists of a central vertex, say c, and $2n$ attached chains of $2m-1$ edges. The ith such chain, denoted by D_i, has vertices $d_{1,i}, d_{2,i}, \cdots, d_{2m-1,i}$. ($T$ is not just a tree, but is a homeomorphic image of a star.) Furthermore, for all j $(1 \leq j \leq n)$, the edge weights on the jth chain are as follows:

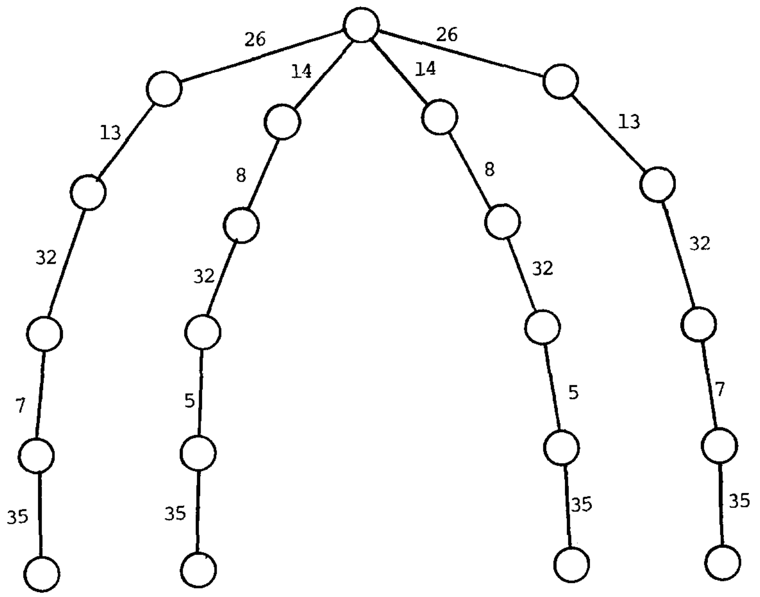
(1) the first edge in the chain has weight $a_{1,j}$, i.e., the value in the first row and jth column of A,
(2) for all i $(2 \leq i \leq m)$, the $2(i-1)$st edge of D_j has weight $a_{i,j}$, i.e., the value in the ith row and jth column of A, and
(3) for all i $(2 \leq i \leq m)$, the $(2i-1)$st edge of D_j has weight $E_i = b_1 - b_i + a_{i,1}$, i.e. the first value of the vector b minus the ith value of the vector b plus the value in the first column and ith row of A.

Figure 2 describes a tree T constructed from an instance of 0, 1-LinEq. We show that T has cutwidth b_1 if and only if (A, b) is a positive instance of 0, 1-LinEq.

($\Leftarrow$): Suppose that there exists a 0-1 valued column vector x such that $Ax = b$. Let $U = \{j \mid x_j = 1\}$ and $V = \{j \mid x_j = 0\}$. It follows that, for every i, the sum of the n elements in row i and a column listed in U is b_i and the sum of the n elements in

$$A = \begin{bmatrix} 26 & 26 & 14 & 14 \\ 13 & 13 & 8 & 8 \\ 7 & 7 & 5 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} 40 \\ 21 \\ 12 \end{bmatrix}$$

(a)



(b)

Fig. 2. (a) An instance of 0, 1-LinEq. (b) The constructed edge weighted tree T.

row i and a column listed in V is also b_i. We assume, without loss of generality, that 1 is in U. We describe a layout of T with cutwidth at most b_1.

Lay out those chains in T corresponding to columns whose index is in U to the right of the center node c and those chains corresponding to columns whose index is in V to the left of c. Let the chains laid out to the right of c be denoted by $D_1, D_2, \ldots, D_n$, where it is assumed that D_1 is the chain corresponding to column 1. Let the nodes of the ith chain D_i be denoted by $d_{1,i}, d_{2,i}, \ldots, d_{2m-1,i}$. The layout of the individual nodes is given by the function f described below. The center node is assigned to position 0. (The layout of the chains to the left of c is done analogously.)

$$f(d_{1,j}) = j,$$

$$\left.\begin{aligned} &f(d_{2i,j}) = (2i-1)\cdot n + 2\cdot j - 1, \text{and} \\ &f(d_{2i+1,j}) = f(d_{2i,j}) + 1. \end{aligned}\right\} \quad \text{for } 1 \leqslant i \leqslant m-1 \text{ and } 1 \leqslant j \leqslant n$$

Figure 3 shows the desired layout of the tree T.

The cut just to the left of a vertex $d_{i,j}$ contains all edges in the tree T that connect vertices assigned by the layout to a position to the left of $d_{i,j}$ with either $d_{i,j}$ itself

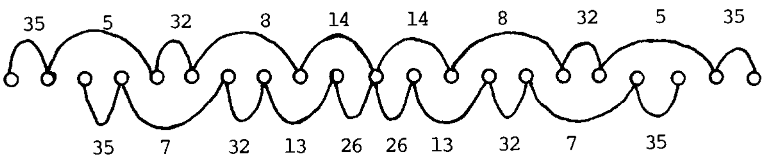


Fig. 3. A cutwidth 40 layout of the edge weighted tree T described in Fig. 2.

or a vertex assigned to a position to the right of $d_{i,j}$. We show that for every vertex $d_{i,j}$ placed to the right of the center vertex c that the cut to the left of $d_{i,j}$ satisfies the property that the sum of the weights on all its edges is at most b_1. As the layout to the left of the center vertex c is similar, it follows that the entire layout of T has cutwidth at most b_1.

(a) For all j $(1 \leq j \leq n)$, the edges in the cut just to the left of $d_{1,j}$ are those connecting $d_{1,i}$ with $d_{2,i}$ for all i $(1 \leq i < j)$ and edges connecting the center vertex c with $d_{1,i}$, for all i $(j \leq i \leq n)$. The weight of an edge connecting $d_{1,i}$ with $d_{2,i}$ is $a_{2,i}$ and the weight of an edge connecting c with $d_{1,i}$ is $a_{1,i}$. Since the second row elements are all smaller than the first row elements, the sum of all such weights is bounded above by $S = \text{sum}\{a_{1,j} \mid j \text{ is in } U\}$. Since x is a solution of the given instance of 0, 1-LinEq, it follows that $S = b_1$. So, all the cuts to the left of vertex $d_{1,j}$ $(1 \leq j \leq n)$ have edges whose weights sum up to at most b_1.

(b) For all i $(1 \leq i \leq m-1)$ and all j $(1 \leq j \leq n)$, the edges in the cut just to the left of $d_{2i,j}$ are those connecting $d_{2i+1,k}$ and $d_{2i+2,k}$, for all k $(1 \leq k < j)$ when there is a vertex $d_{2i+2,k}$, i.e., when $i < m-1$, and those connecting $d_{2i-1,k}$ and $d_{2i,k}$, for all k $(j \leq k \leq n)$. The weight of an edge connecting $d_{2i+1,k}$ and $d_{2i+2,k}$ is $a_{i+1,k}$ and the weight of an edge connecting $d_{2i-1,k}$ and $d_{2i,k}$ is $a_{i,k}$. Since the elements in row $i+1$ are all less than the elements in row i, the sum of all such weights is bounded by $S = \text{sum}\{a_{i,j} \mid j \text{ is in } U\}$. Since x is a solution of the given instance of 0, 1-LinEq, it follows that $S = b_i < b_1$.

(c) For all i $(1 \leq i \leq m-1)$ and all j $(1 \leq j \leq n)$, the edges in the cut just to the left of $d_{2i+1,j}$ are those connecting $d_{2i+1,k}$ and $d_{2i+2,k}$ for all k $(1 \leq k < j)$ when there is a vertex $d_{2i+2,k}$, i.e., when $i < m-1$; those connecting $d_{2i-1,k}$ and $d_{2i,k}$ for all k $(j < k \leq n)$, and the edge connecting $d_{2i,j}$ and $d_{2i+1,j}$. The weight of an edge connecting $d_{2i+1,k}$ and $d_{2i+2,k}$ is $a_{i+2,k}$, the weight of an edge connecting $d_{2i-1,k}$ and $d_{2i,k}$ is $a_{i+1,k}$, and the weight of an edge connecting $d_{2i,j}$ and $d_{2i+1,j}$ is E_{i+1}. The sum of all such weights is bounded above by $S = \text{sum}\{a_{i+2,k} \mid k < j \text{ and } k \text{ in } U\} + \text{sum}\{a_{i+1,k} \mid k > j \text{ and } k \text{ in } U\} + E_{i+1}$. As all elements in row $i+2$ are less than all elements in row $i+1$, it follows that S is bounded above by $\text{sum}\{a_{i+1,k} \mid k \text{ in } U\} - a_{i+1,1} + E_{i+1}$. That is, if $a_{i+2,1}$ is in $\{a_{i+2,k} \mid k < j \text{ and } k \text{ in } U\}$, replace it by $a_{i+1,j}$ and replace all other elements $a_{i+2,k}$ in the set by the corresponding $a_{i+1,k}$. Each number is replaced by a greater number and we have replaced $\text{sum}\{a_{i+2,k} \mid k < j \text{ and } k \text{ in } U\} + \text{sum}\{a_{i+1,k} \mid k > j \text{ and } k \text{ in } U\}$ by $\text{sum}\{a_{i+1,k} \mid k \text{ in } U\} - a_{i+1,1}$. Finally, as x is a solution of the given instance of 0, 1-LinEq, it follows that

$$S \leq b_{i+1} - a_{i+1,1} + E_{i+1} = b_{i+1} - a_{i+1,1} + b_1 - b_{i+1} + a_{i+1,1} = b_1.$$

($\Rightarrow$): Let f be a layout of T with cutwidth at most b_1. We need to show that there is a 0-1 solution vector x for the instance (A, b) of 0, 1-LinEq. Observe that the sum of the weights on the $2n$ edges incident to the center vertex c is $2b_1$. Consequently, the layout must place some of T's chains entirely to the left of c and the remaining chains entirely to the right of c. That is, the cuts just to the right and left of the center vertex c must have edges whose weights sum to b_1 and these weights come just from those edges incident to the center vertex c. Consequently, we get a partition of the columns of A into $R = \{i \mid \text{the chain of } T \text{ corresponding to column } i \text{ is placed to the right of } c\}$ and $L = \{i \mid \text{the chain of } T \text{ corresponding to column } i \text{ is placed to the left of } c\}$. It follows that

$$\text{sum}\{a_{i,1} \mid i \text{ is in } L\} = b_1 = \text{sum}\{a_{i,1} \mid i \text{ is in } R\}.$$

By condition (5) it follows that there are n elements in R and n elements in L. Let $\{D_i \mid i \text{ in } R\}$ be the set of n chains placed to the right of the center vertex c in the layout. For all i, let D_i contain the vertices $d_{1,i}, d_{2,i}, \ldots, d_{2m-1,i}$. Let x be the 0-1 vector such that $x_i = 1$ if and only if i is in R. We show that $Ax = b$. (We have already shown that the first entry in the vector Ax is b_1.)

We now show that the first n positions to the right of c in the layout f must contain the vertices in $\{d_{1,i} \mid i \text{ in } R\}$. Assume not. Then there is a j $(0 \le j < n)$ such that the first j vertices to the right of c form the set $\{d_{1,i} \mid i \text{ in } R(+)\}$ for some subset $R(+)$ of R, and the $(j+1)$st vertex to the right of c, say vertex x in some chain D_k, is not $d_{1,i}$ for any i. Let $R(-)$ denote the set of indices for chains placed to the right of c that do not have their first vertex placed between c and x. As x is not in $\{d_{1,i} \mid i \text{ in } R\}$, it follows from the definition of the tree T that it is incident to an edge with weight E_s, for some s $(1 < s \le m)$. Furthermore,

(1) there is an edge connecting each of the j vertices in the set $\{d_{1,i} \mid i \text{ in } R(+)\}$ placed between c and x to the corresponding vertices in the set $\{d_{2,i} \mid i \text{ in } R(+)\}$ and the sum of the weights of these edges is $\text{sum}\{a_{2,i} \mid i \text{ in } R(+)\}$ and

(2) there is an edge connecting the center vertex c to each of the $n-j$ vertices in the set $\{d_{1,i} \mid i \text{ in } R(-)\}$ and the sum of the weights on these edges is $\{a_{1,i} \mid i \text{ in } R(-)\}$.

It follows that there is a cut immediately to the right of x with edges whose weights sum to at least

$$\text{sum}\{a_{2,i} \mid i \text{ in } R(+) - \{j\} \text{ for some } j \text{ in } R(+)\} + \text{sum}\{a_{1,i} \mid i \text{ in } R(-)\} + E_s.$$

(The value $a_{2,j}$ is not included in this sum if $x = d_{2,j}$.) An example of the general situation described is shown in Fig. 4.

Substituting the defined value for E_s, we obtain

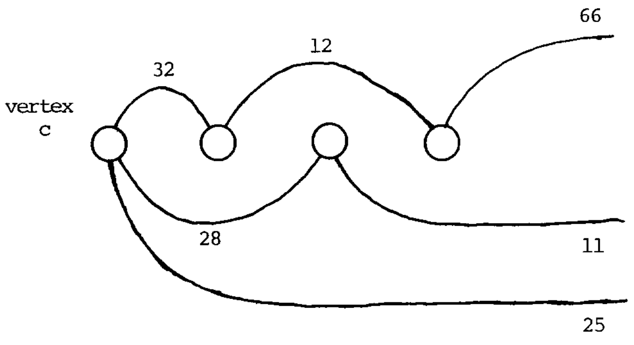
$$\text{sum}\{a_{2,i} \mid i \text{ in } R(+) - \{j\} \text{ for some } j \text{ in } R(+)\}$$
$$+ \text{sum}\{a_{1,i} \mid i \text{ in } R(-)\} + b_1 - b_s + a_{s,1}.$$

As the cutwidth is at most b_1, this sum must be at most b_1. So,

$$\text{sum}\{a_{2,i} \mid i \text{ in } R(+) - \{j\} \text{ for some } j \text{ in } R(+)\} + \text{sum}\{a_{1,i} \mid i \text{ in } R(-)\} + a_{s,1}$$
$$\le b_s.$$

$$A = \begin{bmatrix} 32 & 32 & 28 & 27 & 26 & 25 \\ 12 & 12 & 11 & 10 & 9 & 8 \\ 7 & 7 & 6 & 5 & 6 & 5 \\ 4 & 4 & 3 & 3 & 2 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 85 \\ 31 \\ 18 \\ 9 \end{bmatrix}$$

(a)



(b)

Fig. 4. (a) An instance (A, b) of 0, 1-LingEq. (b) A layout of the constructed edge weighted tree T that does not place all vertices adjacent to c as close to c as possible and hence has cutwidth larger than $b_1 = 85$.

However, as each element in the ith row of A is larger than any element in the $(i+1)$st row for all i, and the sum of any element in row s with $n-1$ elements from row $s-1$ is larger than b_s, it follows that

$$\text{sum}\{a_{2,i} \mid i \text{ in } R(+) - \{j\} \text{ for some } j \text{ in } R(+)\} + \text{sum}\{a_{1,i} \mid i \text{ in } R(-)\}$$
$$> b_s.$$

This contradicts the previous inequality. (An identical argument shows that the first n vertices placed to the left of c are those adjacent to c in the tree.) Let $R(0) = \{d_{1,i} \mid i \text{ in } R\}$ and, for all $j > 0$, let $R(j) = \{d_{2j,i}, d_{2j+1,i} \mid i \text{ in } R\}$. Similarly, let $L(0) = \{d_{1,i} \mid i \text{ in } L\}$ and, for all $j > 0$, let $L(j) = \{d_{2j,i}, d_{2j+1,i} \mid i \text{ in } L\}$.

We have seen that the layout f assigns all vertices in $R(0)$ to the integers $1, \ldots, n$ and all vertices in $L(0)$ to $-n, \ldots, -1$. We now show that f also places, for all $j > 0$, all vertices in $R(j)$ to the right of all vertices in $R(j-1)$. Suppose not. Consider the first occurrence of a vertex x in $U\{R(j) \mid j > r\}$ to the left of a vertex in $R(r)$. As this is the first occurrence of such a situation, the vertices to the left of x are in the indicated order. So, the cut just to the right of x contains $n-1$ edges of the following types:

(a) those connecting vertices in $R(r)$ with vertices in $R(r+1)$;

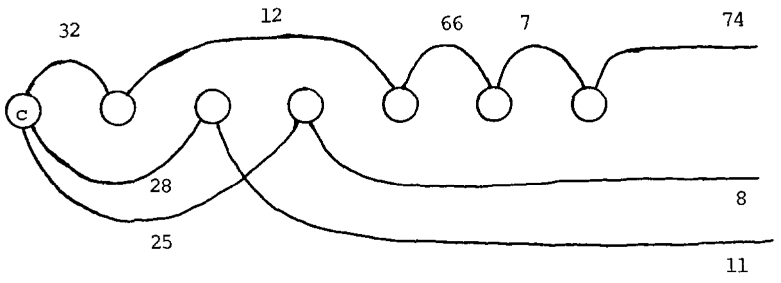
(b) those connecting vertices in $R(r-1)$ with $R(r)$, or

(c) those connecting a vertex in $R(r)$ with another vertex in $R(r)$.

Furthermore, there must be at least one edge connecting a vertex in $R(r-1)$ with one in $R(r)$. (The last condition follows from the assumption that there is at least one vertex in $R(r)$ to the right of the misplaced vertex x.) The sum of the weights on these $n-1$ edges is at least as large as the sum of one value from the rth row of A and $n-2$ values from the $(r+1)$st row. Also, there is an edge incident to x with weight E_s, for some $s>r$. Such a situation is shown in Fig. 5.

$$A = \begin{bmatrix} 32 & 32 & 28 & 27 & 26 & 25 \\ 12 & 12 & 11 & 10 & 9 & 8 \\ 7 & 7 & 6 & 5 & 6 & 5 \\ 4 & 4 & 3 & 3 & 2 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 85 \\ 31 \\ 18 \\ 9 \end{bmatrix}$$

(a)



(b)

Fig. 5. (a) An instance (A, b) of 0, 1-LinEq. (b) A layout of the constructed edge weighted tree T that does not place all the vertices in $R(2)$ to the right of the vertices in $R(1)$ and hence has cutwidth larger than $b_1 = 85$.

As $E_s = b_1 - b_s + a_{s,1}$ and all elements in the $(r+1)$st row are greater than or equal to any element in the sth row, the sum of all the weights on edges in the cut to the right of x must be at least as large as $b_1 - b_s + S$, where S is the sum of $n-1$ elements in the sth row of A plus one element from the rth row, where $s>r$. By condition (5), S is greater than b_s. So, $b_1 - b_s + S > b_1$. However, this contradicts the fact that the layout f has cutwidth b_1. By a similar argument it can be seen that f places, for all $j>0$, all vertices in $L(j)$ to the right of all vertices in $L(j+1)$.

Observe that $f(d_{2j+1,i}) = f(d_{2j,i}) + 1$ must be true, for all i in R. If not, there would be a cut between consecutive vertices in the layout with edges having weights E_j and E_t, $t>1$. (That is, the edge connecting $d_{2j,i}$ and $d_{2j+1,i}$ has weight E_j. When some other vertex is placed between these two vertices, as it is connected to an edge with weight E_t for some t, we have a cut with edges of weight E_j and E_t.) Observe that

$$\begin{aligned} E_j + E_t &= b_1 - b_j + a_{j,1} + b_1 - b_t + a_{t,1} = b_1 + (b_1 - b_j - b_t) + a_{j,1} + a_{t,1} \\ &\geqslant b_1 + a_{j,1} + a_{t,1} > b_1, \end{aligned}$$

as b_1 is more than twice as big as either b_j or b_t by condition (6). However, this contradicts the fact that the layout has cutwidth b_1. A similar argument shows that $f(d_{2j+1,i}) = f(d_{2j,i}) - 1$ is true for all i in L.

For any $j > 0$, consider the vertex in $R(j)$ closest to the center vertex c in the layout. Let it be $d_{2j,p}$. Then, the sum of the weights on the edges in the cut immediately to its right is

$$\begin{aligned} &\text{sum}\{a_{j,i} \mid i \text{ in } R\} - a_{j,p} + E_j \\ &\quad = \text{sum}\{a_{j,i} \mid i \text{ in } R\} - a_{j,p} + b_1 - b_j + a_{j,1}. \end{aligned}$$

As f is a layout with cutwidth at most b_1, it follows that

$$\text{sum}\{a_{j,i} \mid i \text{ in } R\} - a_{j,p} + a_{j,1} \leq b_j$$

and, in fact, as

$$a_{j,p} \leq a_{j,1}, \qquad \text{sum}\{a_{j,i} \mid i \text{ in } R\} \leq b_j.$$

A similar argument shows that $\text{sum}\{a_{j,i} \mid i \text{ in } L\} \leq b_j$. By condition (3) on the given instances of 0, 1-LinEq,

$$\text{sum}\{a_{j,i} \mid i \text{ in } R\} + \text{sum}\{a_{j,i} \mid i \text{ in } L\} = 2b_j,$$

it follows that $\text{sum}\{a_{j,i} \mid i \text{ in } R\} = b_j$.

Thus, we have $Ax = b$. It follows that $Ax = b$ if and only if the weighted tree T has cutwidth b_1. □

Combining Lemmas 2.3 and 2.4 we obtain the following theorem.

Theorem 2.5. *The Weighted Min Cut problem restricted to trees with polynomial edge weights is* NP-*complete.*

Combining Theorem 2.5 with Lemma 2.1 we obtain the next theorem.

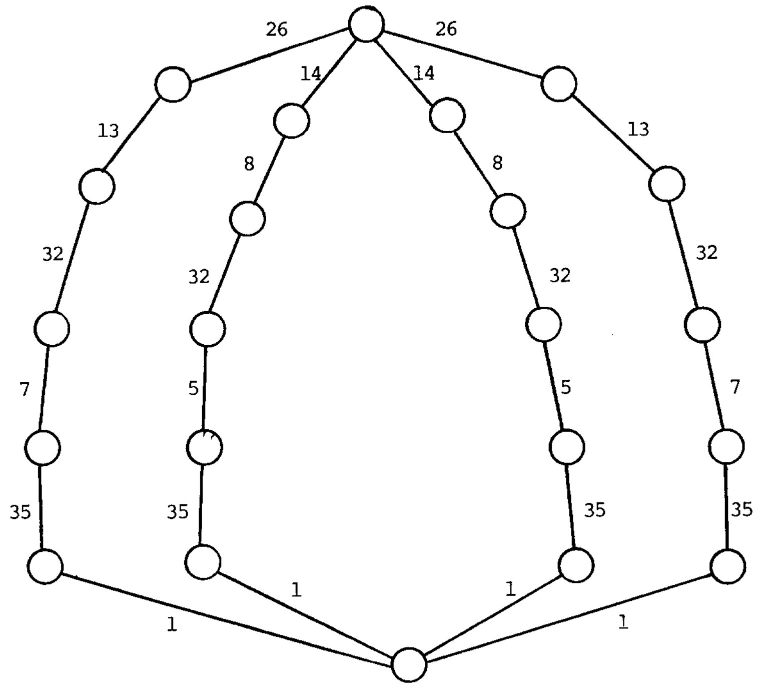
Theorem 2.6. *The Min Cut problem restricted to planar graphs is* NP-*complete.*

Consider the weighted tree T constructed in the proof of Lemma 2.4. We can ensure that every edge has weight $>2n$, by requiring each element of the matrix A to be greater than $2n$. This can easily be done by adding $2n$ to each element of the matrix and the value $\frac{1}{2}n \cdot 2n = n^2$ to each element of the column vector b. As observed, the weighted tree T constructed in the proof of Lemma 2.4 is a homemorphic image of a star. Consequently, one can form a series-parallel graph, say SP(T), by

(1) adding a new vertex x and an edge with weight 1 connecting each leaf of T to x and

(2) replacing each of the edges with a weight, say k, by k parallel edges (and adding a degree-2 vertex, if desired, to all resulting parallel edges).

It follows that SP(T) has cutwidth $b_1 + n$ if and only if T has cutwidth b_1. Note that, because each of the weights on edges of T are greater than $2n$ and the sum of the weights incident to the center vertex x is $2 \cdot b_1$, a cutwidth $b_1 + n$ layout of SP(T) must still place n chains incident to the center vertex c to the right of c and the other n chains to the left of c. In fact, a cutwidth b_1 layout of T can be transformed into a cutwidth $b_1 + n$ layout of SP(T) simply by inserting the vertex x in a position next to c and a cutwidth $b_1 + n$ layout of SP(T) can be transformed into a cutwidth b_1 layout of T simply by deleting x and its $2n$ incident edges. Figure 6 shows an example of such a weighted tree T, the corresponding series-parallel graph SP(T), and a layout of SP(T). Thus, we have the following corollary.



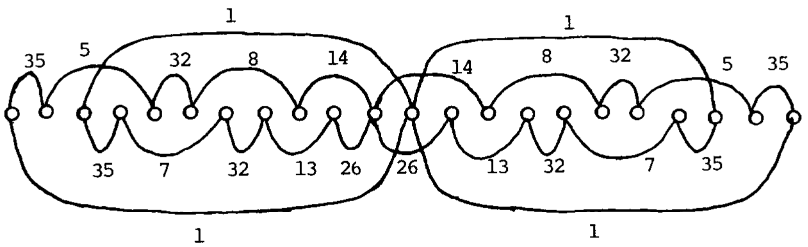


Fig. 6. (a) The series-parallel graph SP(T) constructed from the edge weighted tree T described in Fig. 2. (b) A cutwidth 42 layout of SP(T).

Corollary 2.7. *The Min Cut problem restricted to series-parallel graphs is* NP-*complete.*

Corollary 2.8. *The Min Cut problem restricted to* 2-*outerplanar graphs is* NP-*complete.* (*Moreover, the Min Cut problem for outerplanar multigraphs is* NP-*complete.*)

Lemma 2.9. *The Min Cut problem for planar graphs* $\leqslant_{\text{poly}}$ *the Min Cut problem for planar graphs with maximum vertex degree* 3.

Proof. Let G be a planar graph and k be a positive integer. We construct a new planar graph G' by substituting a planar component for each vertex of G. The components substituted for vertices are U-shaped walls. The *U-shaped wall* $U(m, n, p, q)$, where $n \geqslant 2q$, is the graph with the vertex set

$$V = \{(i,j) \mid 1 \leqslant i \leqslant m \text{ and } 1 \leqslant j \leqslant n\} \cup \{(i,j) \mid 1 \leqslant i \leqslant m+p \text{ and } 1 \leqslant j \leqslant q\}$$
$$\cup \{(i,j) \mid 1 \leqslant i \leqslant m+p \text{ and } n-q+1 \leqslant j \leqslant n\}$$

and the edge set

$\{((i,j), (i,j+1)) \mid$ for all i, j such that (i,j) and $(i,j+1)$ are in $V\}$

$\cup \{((i,j), (i+1,j)) \mid$ for all i,j such that (i,j), $(i+1,j)$ are in V and at least one of the following is true: (1): $i+j$ is even, (2): $j=1$, (3): $j=n$, (4): $m \leqslant i < m+p$ and $j=q$, or (5): $m \leqslant i < m+p$ and $j = n-q+1\}$.

The U-shaped wall $U(4, 11, 4, 4)$ is shown in Fig. 7. Observe that $U(m, n, p, q)$ has maximum vertex degree three if $m+q$ is even and n is odd. (The principal reason that $U(m, n, p, q)$ has maximum vertex degree 3 is that the vertical edges are staggered from row to row, as shown in Fig. 7. The conditions "$m+q$ is even" and "n is odd" ensure that the edges around the periphery of the U-shaped wall do not make peripheral vertices have degree 4.)

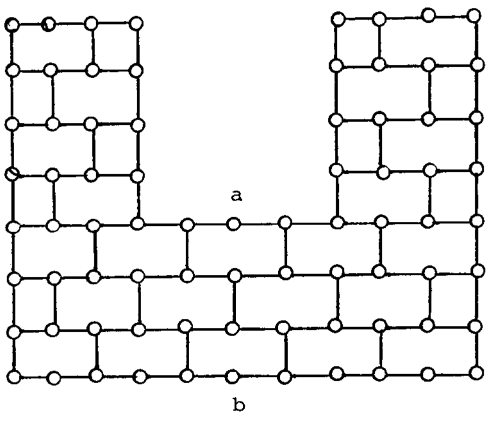


Fig. 7. The U-shaped wall $U(4, 11, 4, 4)$. The vertices a and b are examples of "connection points".

It is known that the cutwidth of a rectangular wall with m rows and n columns, when $n > 4m^2$ is $m+1$ [17]. (A rectangular wall with m rows and n columns is just $U(m, n, 0, 0)$.) In fact, an optimum cutwidth layout simply lays out the wall column by column, i.e., all of the vertices in column i are positioned before the vertices in column $i+1$. Moreover, the vertices within a column are laid out in the following order: for all j, the vertex in row j is assigned a position to the left of the vertex in row $j+1$.

We replace each vertex of G with the U-shaped wall $U(2k, 16k^2n, 2k, 16k^2)$. It has cutwidth $4k+1$. That is, it has cutwidth at least $4k+1$, as it has a rectangular wall with $4k$ rows and $16k^2$ columns as a subgraph, and it has cutwidth no larger than $4k+1$, as it can be laid out column by column in the manner described above. Let U_x denote the copy of $U(2k, 16k^2n, 2k, 16k^2)$ that is substituted for vertex x of G. Observe that there are vertices on the periphery in the short part of U_x that have only two incident edges. These are called "connection points". One such point is shown in Fig. 7. Each edge of G that is incident to vertex x connects to a distinct connection point of U_x in G'. Thus, for every edge $\{x, y\}$ of G, there is an edge in G' connecting a unique connection point of U_x to a unique connection point of U_y.

It follows that G' has maximum vertex degree 3, as the components U_x substituted for each vertex x in G have maximum vertex degree 3 and the edges connecting distinct components are made to distinct vertices in these components that have degree 2 (when considered as part of the component alone). We claim that G has cutwidth k if and only if G' has cutwidth $k' = 5k+1$.

($\Rightarrow$): Given a cutwidth k layout of G, we arrange the U-shaped components of G' in the same order as the vertices of G. That is, all of the vertices of U_x will lie to the left of all of the vertices of U_y if and only if x lies to the left of y in the layout of G. Furthermore, the U-shaped components are laid out in the manner described above so that they have cutwidth $4k+1$. Since the given layout of G has cutwidth k, there are never more than k edges crossing between consecutive vertices in the layout of G. Consequently, there will never be more than k edges of G' passing over the tall portion of any U-shaped grid. Since the grid itself has cutwidth $4k+1$, the total cutwidth of G' with these additional edges is at most $5k+1$. (Note that, since G has cutwidth k, there cannot be more than $2k$ edges incident to any vertex x of G. Consequently, there cannot be more than $2k$ edges incident to connection points of U_x. So, there can be at most $2k$ such edges adding to the total cutwidth in the short section of the U-shaped wall. Since the short section has cutwidth $2k$, the total cutwidth in this part of the layout is $4k < 5k+1$.)

($\Leftarrow$): Given a cutwidth $5k+1$ layout, say L', of G', we observe that it is not possible in the layout L' for U-shaped walls to overlap substantially. That is, each U-shaped wall $U(2k, 16k^2n, 2k, 16k^2)$ has cutwidth $4k+1$ and substantial overlapping of two such walls would combine their cutwidth yielding cutwidth at least $8k+2 > 5k+1$. Consequently, from the given layout L' of G', one obtains an induced layout, say L, of G. The vertices of G are arranged in L in the same order as the U-shaped walls appear in L'. This layout of G has cutwidth k since, for any vertex

x of G, there can be at most k edges incident to connection points of U_x passing over either of the tall segments of U_x. (Otherwise, the cutwidth of L' would not be $5k+1$.) Therefore, there are at most k edges cut by any line segment drawn between two consecutive vertices of L. □

Corollary 2.10. *The Min Cut problem is* NP*-complete even when restricted to planar graphs with maximum vertex degree* 3.

Corollary 2.11. *The Search Number problem is* NP*-complete even when restricted to planar graphs with maximum vertex degree* 3.

Proof. This follows immediately from Corollary 2.9 since the Search Number problem is identical to the Min Cut problem for graphs with maximum vertex degree 3 [16]. □

Corollary 2.12. *The vertex Separation problem is* NP*-complete even when restricted to planar graphs with maximum vertex degree* 3.

Proof. This follows immediately from Corollary 2.11 by a transformation described in [5], where it is shown that, for any graph G, the vertex separation of a transformed graph, denoted by $2e(G)$, is identical to the search number of G. (The graph $2e(G)$ is the homeomorphic image of G obtained by adding two degree-2 vertices into every edge of G.) If G is planar and has maximum vertex degree 3, then $2e(G)$ is planar and has maximum vertex degree 3. Thus, G has search number k if and only if $2e(G)$ has vertex separation k. So we have a polynomial-time reduction from the Search Number problem restricted to planar graphs with maximum vertex degree 3 to the Vertex Separation problem restricted to planar graphs with maximum vertex degree 3. □

Corollary 2.13. *The Progressive Black/White Pebble Demand problem is* NP*-complete even when restricted to planar dags with maximum vertex degree* 3.

Proof. This follows immediately from Corolloary 2.12 by a reduction described by Lengauer from the Vertex Separation problem to the Progressive Black/White Pebble Demand problem [13]. Since Lengauer's reduction preserves planarity and only adds vertices with two incoming edges, the result follows. □

Corollary 2.14. *The Modified Min Cut problem is* NP*-complete even when restricted to planar graphs with maximum vertex degree* 3.

Proof. The reduction described in the proof of Lemma 2.9 can be used for a reduction from the Min Cut problem restricted to planar graphs to the Modified Min Cut problem restricted to planar graphs with maximum vertex degree 3. That is, G has cutwidth k if and only if G', as described in the proof of Lemma 2.9, has modified cutwidth $5k$. □

Corollary 2.15. *The Topological Bandwidth problem is* NP*-complete even when restricted to planar graphs with maximum vertex degree* 3.

Proof. This follows immediately from Corollary 2.14 using the result that the topological bandwidth of a graph G with maximum vertex degree 3 is exactly 1 greater than the modified cutwidth of G [17]. □

References

[1] B. Baker, Approximation algorithms for NP-complete problems on planar graphs, in: *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science*, Tucson (1983) 265–273.
[2] F.R.K. Chung, On the cutwidth and the topological bandwidth of a tree, *SIAM J. Algebraic Discrete Methods* **6** (1985) 268.
[3] M.-J. Chung, F. Makedon, I.H. Sudborough and J. Turner, Polynomial algorithms for the Min-Cut linear arrangement problem on degree restricted trees, *SIAM J. Comput.* **14**(1) (1985) 158–177.
[4] S.A. Cook and R. Sethi, Storage requirements for deterministic polynomial time recognizable languages, *J. Comput. System Sci.* **13** (1976) 25–37.
[5] J.A. Ellis, I.H. Sudborough and J.S. Turner, Graph separation and search number, in: *Proc. 1983 Allerton Conf. on Communication, Control, and Computing*; also: *SIAM J. Comput.*, to appear.
[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).
[7] M.R. Garey, R.L. Graham, D.S. Johnson and D.E. Knuth, Complexity results for bandwidth minimization, *SIAM J. Applied Math.* **34** (1978) 477–495.
[8] J.R. Gilbert, T. Lengauer and R.E. Tarjan, The pebbling problem is complete in polynomial space, *SIAM J. Comput.* **9**(3) (1980) 513–524.
[9] J.E. Hopcroft, W. Paul and L. Valiant, On time versus space, *J. ACM* **24** (1977) 332–337.
[10] M. Kirousis and C.H. Papadimitriou, Searching and pebbling, *Theoret. Comput. Sci.* 47(2) (1986) 205–218.
[11] M. Klawe, A tight bound for black and white pebbles on the pyramid, *J. ACM* **32**(1) (1985) 218–228.
[12] S. LaPaugh, Recontamination does not help to search a graph, *J. ACM*, to appear.
[13] T. Lengauer, Black-White pebbles and graph separation, *Acta Informatica* **16** (1981) 465–475.
[14] R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs, *SIAM J. Appl. Math.* **36**(2) (1979) 177–189.
[15] F. Meyer auf der Heide, A comparison of two variations of a pebble game on graphs, *Theoret. Comput. Sci.* **13** (1981) 315–322.
[16] F. Makedon and I.H. Sudborough, Minimizing width in linear layouts, in: *Proc. 10th ICALP* (Barcelona) *Lecture Notes in Computer Science* **154** (Springer, Berlin, 1983) 478–490.
[17] F. Makedon, C.H. Papadimitriou and I.H. Sudborough, Topological bandwidth, *SIAM J. Algebraic Discrete Methods* **6**(3) (1985) 418–444.
[18] N. Megiddo, S.L. Hakimi, M.R. Garey, D.S. Johnson and C.H. Papadimitriou, The complexity of searching a graph (preliminary version), in: *Proc. IEEE Foundations of Computer Science Symp.*, Nashville (1981) 376–385.
[19] Z. Miller, A linear time algorithm for the topological bandwidth of a tree with maximum vertex degree three, *SIAM J. Comput.*, to appear.
[20] T.D. Parsons, Pursuit-evasion in a graph, in: Y. Alavi and D.R. Dick, eds., *Theory and Application of Graphs* (Springer, Berlin, 1976) 426–441.

[21] T.D. Parsons, The search number of a connected graph, in: *Proc. 9th Southeastern Conf. on Combinatorics, Graph Theory, and Computing* (Utilitas Math., Winnipeg, Canada, 1978) 549-554.

[22] A. Rosenberg and I.H. Sudborough, Bandwidth and pebbling, *Computing* **31** (1983) 115-139.

[23] R. Sethi, Complete register allocation problems, *SIAM J. Comput.* **4** (1975) 226-248.

[24] R. Wilber, White pebbles help, in: *Proc. 17th Ann. ACM Symp. on Theory of Computing*, Providence (1985) 103-112.

[25] M. Yannakakis, A polynomial algorithm for the Min Cut Linear Arrangement of Trees, *J. ACM* **32**(4) (1985) 950-988.

[26] B. Monien, The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete, *SIAM J. Algebraic Discrete Methods* **7** (1986) 505-512.

[27] Ch. H. Papadimitriou, The NP-Completeness of the Bandwidth Minimization Problem, *Computing* **16** (1976) 263-270.