Math. Systems Theory **24**, 11–40 (1991)

Mathematical Systems Theory

© 1991 Springer-Verlag New York Inc.

A Polynomial Algorithm for Recognizing Bounded Cutwidth in Hypergraphs*

Zevi Miller[1] and I. H. Sudborough[2]

[1] Department of Mathematics and Statistics, Miami University, Oxford, OH 45056, USA

[2] Computer Science Program, University of Texas at Dallas, Richardson, TX 75083, USA

Abstract. The Min Cut Linear Arrangement (Min Cut) or Backboard Permutation (BP) problem, where it is desired to minimize "backplane area" or "cutwidth" in hypergraphs, has a long history of interest. To determine, for given graph G and integer k, whether G has cutwidth at most k is known to be NP-complete even for planar graphs with maximum vertex degree 3. (As graphs are a special case of hypergraphs, it is also NP-complete for hypergraphs.) Recently, Cahoon and Sahni described $O(n)$ and $O(n^3)$ algorithms for determining if a hypergraph had cutwidth 1 and 2, respectively. However, for any fixed $k > 2$, it remained open whether determining if an arbitrary hypergraph has cutwidth at most k was in the class P. We show a positive answer; specifically, we describe an $O(n^m)$ algorithm, with $m = k^2 + 3k + 3$, which determines if a hypergraph with n vertices has cutwidth k.

1. Introduction

We consider linear arrangements of circuit boards or gates for the sake of minimizing the channel width of their interconnections. This problem, which for graphs has been called the Min Cut Linear Arrangement problem [GJ], [CMST], [Y], and for hypergraphs has been called the Board Permutation problem [S], [CS1], [CS2], is applicable to designing gate matrices [L], [W], obtaining an

* The work of Z. Miller was begun during the fall semester 1985 spent as a Visiting Professor in the Computer Science Program, University of Texas at Dallas. Support was also provided for this author by ONR Grant No. N00014-85-K-0621.

optimal backboard ordering [Ce], [GCT], [SS], and is related to several graph problems [MHG*], [MS], [Ch], [M], [MPS], [MoS].

By a *hypergraph* G we mean a set V, denoted *vertices*(G), whose elements are called *vertices* or *points* of G, along with a collection of subsets of V, denoted *hyperedges*(G), whose elements are called *hyperedges* or *edges* of G. We allow edges to appear in *hyperedges*(G) with multiplicity greater than one. Let G be a finite hypergraph with positive integer weight $w(A)$ associated with each hyperedge A. A *linear layout* L of G (or simply a layout) is a one-to-one mapping from vertices(G) to $\{1, \ldots, |\text{vertices}(G)|\}$. ($L$ is a *partial layout* if it maps a subset V of vertices(G) to $\{1, \ldots, |V|\}$.) Consider, for a given layout (or partial layout) L of G and an integer i, the set $\text{cut}_{G,L}(i) = \{A \mid A$ is a hyperedge which contains both (1) a vertex mapped by L to an integer less than or equal to i and (2) a vertex either not in the domain of L or mapped by L to an integer greater than $i\}$. Writing I for the interval $(i, i+1)$, we also denote this set by $cut(I, L)$ when G is understood. Let $sum_{G,L}(i)$, or $sum(i)$ when G and L are understood, denote the summation of weights in $\{w(A) \mid A$ is in $\text{cut}_{G,L}(i)\}$. The *cutwidth of G under a layout L*, denoted by *cutwidth*(G, L), is $\max\{\text{sum}_{G,L}(i) \mid 1 \le i \le |\text{domain}(L)|\}$. We say that a partial layout L is *k-plausible* if cutwidth$(G, L) \le k$. We define *cutwidth*(G) to be $\min\{\text{cutwidth}(G, L) \mid L$ is a linear layout of G and domain$(L) = G\}$. The problem of deciding, for a given graph G and positive integer k, whether cutwidth$(G) \le k$ is known to be NP-complete, even when the graph G is planar with maximum vertex degree 3 [MoS]. Consequently, we restrict ourselves to the class of problems in which the cutwidth bound is fixed. For each fixed integer k, we have the following problem:

The (Weighted) Min Cut $\le k$ Problem

Input: A finite hypergraph G with positive integer weight $w(A)$ associated with each hyperedge A.
Question: Is cutwidth$(G) \le k$?

We give in Figure 1 an example of a hypergraph with uniform hyperedge weight one and its cutwidth under two linear layouts.

Previously, no general polynomial-time algorithm for the Min Cut $\le k$ problem was known (even with uniform hyperedge weights) except for the special cases when k equals one or two [CS1]. We explicitly describe a polynomial-time algorithm for all values of k. Our description is for the case when hyperedge weights are one, although the technique easily generalizes to hypergraphs with arbitrary (polynomial size) hyperedge weights.

Recently, Fellows and Langston [FL] have used results of Robertson and Seymour [RS1], [RS2] to show the existence of an algorithm to decide the Min Cut $\le k$ problem, for all k, in $O(n^2)$ time. However, these results only guarantee the *existence* of an algorithm. They are inherently nonconstructive and do not actually tell us how to find an algorithm. Furthermore, algorithms deducible from the work of Robertson and Seymour, such as those described by Fellows and Langston [FL] with time complexity $C(k)n^2$, have an enormous constant $C(k)$ that grows at several exponential levels with k, rendering them

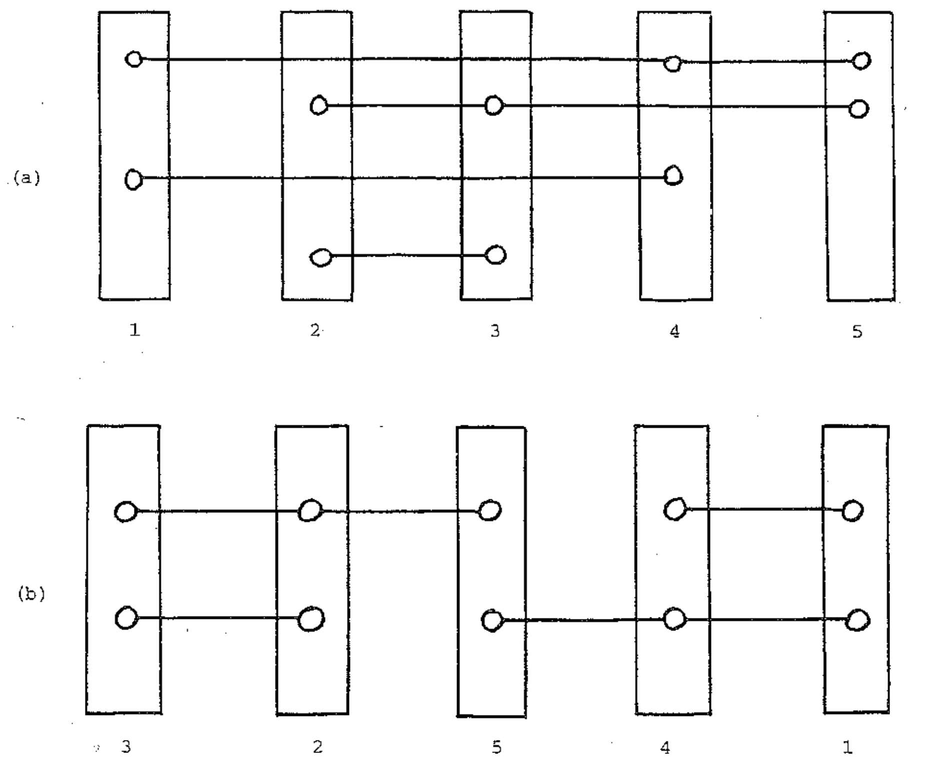

Fig. 1. A hypergraph H = {{1, 4, 5}, {2, 3, 5}, {1, 4}, {2, 3}} and (a) a cutwidth 4 layout of H and (b) a cutwidth 2 layout of H.

impractical for problems of any nontrivial size [J]. Thus, such work serves to direct attention to a search for alternate algorithms that can be explicitly described and hopefully eventually made truly practical.

The Min Cut $\leq k$ problem for hypergraphs seems, unfortunately, to be harder than for graphs. We review previous results for the Min Cut $\leq k$ problem when restricted to graphs. It can be solved in $O(n^{k-1})$ steps, for all $k > 1$, where n is the number of vertices in the graph [MS]. The Min Cut problem can be solved in $O(n \log n)$ time for trees [Y]. (An earlier $O(n \log^{d-2} n)$ algorithm for the Min Cut problem on trees with maximum vertex degree d is described in [CMST], where a characterization of cutwidth in trees is used to equate cutwidth and search number [MHG*] for trees with maximum vertex degree 3. In fact, cutwidth and search number are identical for all graphs with maximum vertex degree 3 [MS].) The weighted Min Cut problem for trees with polynomial-size edge weights is known to be *NP*-complete [MoS].

Our algorithm for the Min Cut $\leq k$ problem uses dynamic programming. To get a rough idea of this process consider first a straightforward use of dynamic programming which tries every possible partial layout of a connected hypergraph G. Let *domain*(L), for any partial layout L, be the set of vertices V of the hypergraph that are mapped by L to some integer. Similarly, let *range*(L) = L(domain(L)). A hyperedge $A = \{x_1, x_2, \ldots, x_k\}$ is *dangling* (from a partial layout L) if there exist i and t such that x_i is in domain(L) and x_t is not. Let *dangling*(L) denote the set of hyperedges dangling from L and *unassigned*(L) denote vertices(G) − domain(L).

We describe the following straightforward algorithm here to make easier the understanding of our descriptions later. The algorithm assumes that the hypergraph is connected. Note that when a hypergraph is not connected all of its connected components can be laid out separately. Let "stack" denote a pushdown stack and let L_0 denote the partial layout whose domain is the empty set:

(1.1) place L_0 on stack;
(1.2) **while** stack is not empty **do**
  begin
(1.3)   delete the top partial layout L from stack;
(1.4)   **if** dangling(L) $= \varnothing$ and $L \neq L_0$ **then** stop and answer "G has cutwidth $\leq k$";
(1.5)   **for** each vertex x that is unassigned in L **do**
    begin
(1.6)   let L' be the partial layout such that $L'(x) = |\text{range}(L)| + 1$ and, for all y in domain(L), $L'(y) = L(y)$, i.e., L' results from placing x at the right end of the layout L;
(1.7)   **if** cutwidth(G, L') $\leq k$ **then** place L' on stack;
    end
  end
(1.8) stop and answer "G has cutwidth $> k$."

It should be noted that testing cutwidth(G, L') $\leq k$ in line (1.7) can be done in linear time as the layout L' is given. As there are $n!$ permutations of n vertices, the running time of the procedure is at least $n!$.

We can do better by defining an equivalence relation R on partial layouts which satisfies the following two properties:

(1) R defines a polynomial number of equivalence classes and
(2) if (L, L') is in R, then either both L and L' can be extended to a complete cutwidth $\leq k$ linear layout of G or neither of them can be so extended.

The idea is that an algorithm, such as the one described above, can be modified so that at most one partial layout from each equivalence class of R is considered. That is, line (1.7) of the above algorithm is changed to include a test of whether the partial layout L' is in an equivalence class that has already been considered. If not, L' is placed on the stack and the equivalence class to which it belongs is marked as considered. If the class has been previously considered, L' is discarded.

This has been done earlier in a dynamic programming algorithm for the Min Cut $\leq k$ problem for graphs [GS]. The equivalence relation used was: L and L' are equivalent if (active(L), dangling(L)) = (active(L'), dangling(L')), where active(L) denotes the set of vertices in domain(L) that are incident to an edge in dangling(L). Notice that, for each fixed value k, this defines an equivalence relation on the class of partial layouts with cutwidth at most k which has $O(n^k)$ equivalence classes. That is, any partial layout L with cutwidth at most k has at most k active vertices. So, there are $O(n^k)$ ways to choose a set of active vertices from a set of n vertices. Note also that a graph cannot have cutwidth k if it has a vertex with more than $2k$ incident edges. So, for a given set of k active vertices, the number of different choices of dangling edges incident to those vertices is a function of k and not of n. So, the number of equivalence classes defined is $O(n^k)$. Unfortunately, for hypergraphs this equivalence relation is not appropriate. If we define active(L), for a hypergraph G and a partial layout L, to be the set of all vertices in domain(L) that are part of a dangling hyperedge, there can be $O(n)$ vertices in active(L) and, more importantly, $O(2^n)$ sets of such vertices. Thus, there are exponentially many equivalence classes and the efficiency of the dynamic programming approach is lost.

The straightforward algorithm, as described above, includes only one operation for extending a partial layout L, namely, assigning a new vertex to a position at the right end of the layout. We add a new operation called "(connected) component absorption" for the sake of extending partial layouts. The "connected components", referred to in the preceding sentence are the connected components in the hypergraphs obtained by deleting various subsets of the set of dangling hyperedges of a given partial layout L. If these components have small enough cutwidth they are absorbed into an existing interval of L. That is, assume we are testing for cutwidth k and L is a partial layout with m dangling hyperedges. Let C be a connected component of the hypergraph obtained by deleting the first i dangling hyperedges (ordered by the position from left to right across the layout L of their leftmost assigned vertices). Let C be completely unassigned by L and let C have cutwidth p. We absorb C into an interval that is to the right of all of the leftmost vertices of the first i dangling hyperedges deleted in the definition of C. (By restricting the absorption intervals in this way, we guarantee that absorption does not add any new intervals to those that the dangling hyperedges already pass over.) So, assume j is such a position to the right of all leftmost vertices of the first i dangling hyperedges and sum(j) $\leq k - p$. Then C can be absorbed into the interval $(j, j + 1)$. The absorption is accomplished by "dilating" the interval and inserting a cutwidth p layout of C. Notice that all of the dangling edges already passed over the interval $(j, j + 1)$ and, after the component C is absorbed, the interval just after the layout of the inserted component C has no more than sum(j) hyperedges. So, a similar absorption can take place again at the latter interval. In fact, when C is absorbed it may result in some of L's dangling hyperedges having all of their vertices assigned and, if the rightmost vertices of these hyperedges lie in C, then these hyperedges will no longer pass over intervals to the right. In other words, there may be more room for future absorptions. Moreover, C under these circumstances could have cutwidth larger than p. It need only have "anchored cutwidth" $p + |E|$, where E is the set of hyperedges that are completed in the component C in this way. The definition of anchored cutwidth follows.

Let G be a finite hypergraph and let $A = (x_1, x_2, \ldots, x_s)$ be a vector of vertices of G. An *A-anchored layout of G* is a linear layout of G in which the vertices in A are assigned to the first few integers. That is, it is a layout L such that $L(x_i) = i$ for all i $(1 \leq i \leq s)$. The *A-anchored cutwidth of G*, denoted by *A-cutwidth*(G), is min{cutwidth(G, L)$|L$ is an A-anchored layout of G}. Note that when the vertices in A are not connected to each other, the order of these vertices in an A-anchored layout has no effect on cutwidth. Consequently, in such cases we replace the vector A by an unordered set. Observe that the cutwidth of a hypergraph G, as defined earlier, is simply the anchored cutwidth where the set A is the empty set, i.e., cutwidth(G) = $\varnothing$-cutwidth(G).

Let G be a hypergraph and let H be a subhypergraph of G. We denote by ∂H the set of edges of G having at least one point in G and at least one point in $G\backslash H$. For any subset $E' \subset \partial H$ of E define the *E'-augmented component* H, denoted by $H(E')$, to be the hypergraph

(1) whose set of vertices is vertices(H) $\cup \{z\}$, where z is a new vertex, and
(2) whose set of hyperedges is hyperedges(H) $\cup N$, where

$$N = \{\{z\} \cup [e_i \cap H] | 1 \leq i \leq s\}.$$

That is, $H(E')$ is the hypergraph obtained from H by adding a new vertex z and, for each hyperedge e_i in E', a new hyperedge e_i' consisting of z and those vertices of e_i that are in H. Note that this augmentation operation may introduce multiple edges, but these are allowed in our definition of a hypergraph. For simplicity we use the term *E'-anchored cutwidth of* $H(E')$ to denote the $\{z\}$-anchored cutwidth of $H(E')$. Now let G be a hypergraph, let $A = \{x_1, x_2, \ldots, x_s\}$ be a set of vertices in G, and let $E = \partial A$. Then the A-anchored cutwidth of G, denoted by cw(G, A), is the same as the E-anchored cutwidth of $[G\text{-}A](E)$, denoted by cw(G, E). In our algorithm it is more convenient to compute cw(G, E) than cw(G, A) and this is the motivation for the definition of $H(E)$ above. See Figures 2 and 3.

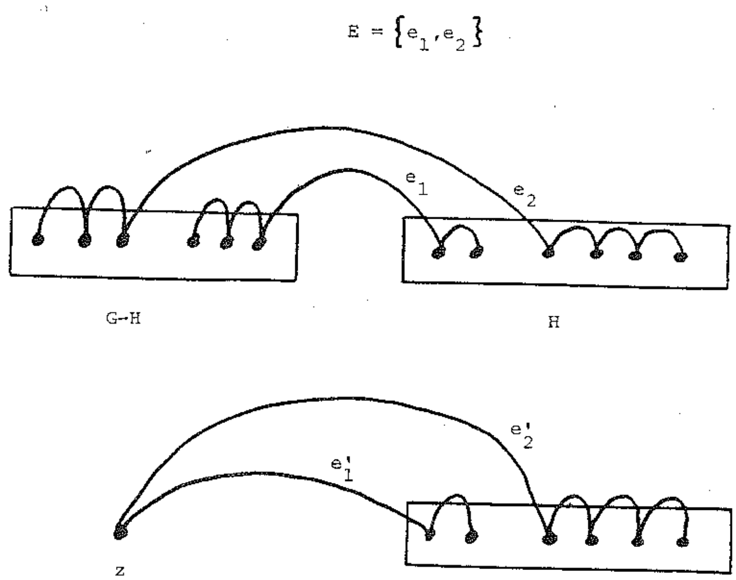

Fig. 2. The E-augmented component $H(E)$ of H.

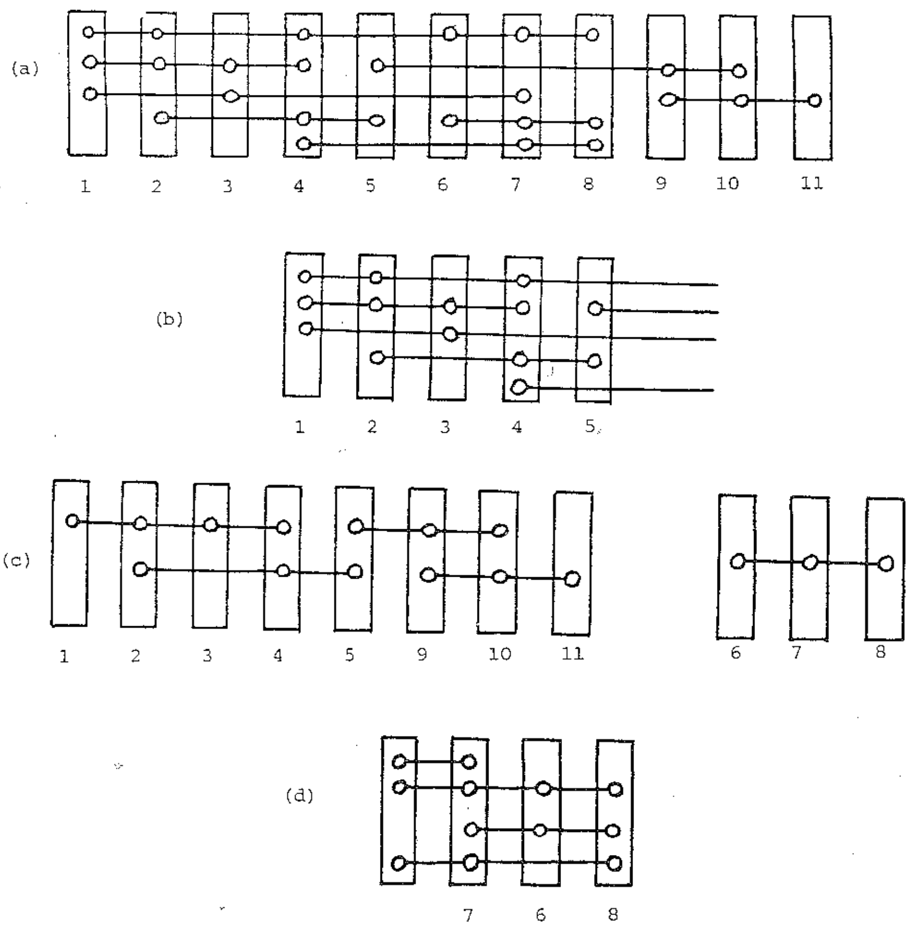

Fig. 3. (a) A hypergraph $H = \{\{1, 2, 4, 6, 7, 8\}, \{1, 2, 3, 4\}, \{1, 3, 7\}, \{2, 4, 5\}, \{4, 7, 8\}, \{5, 9, 10\}, \{6, 7, 8\}, \{9, 10, 11\}\}$, (b) a partial layout of H, (c) the connected components of $H - \{\{1, 2, 4, 6, 7, 8\}, \{1, 3, 4\}, \{4, 7, 8\}\}$, and (d) an anchored layout of the augmented component $A(\{e_1, e_2, e_3\})$, where A is the connected component containing vertices 6, 7, and 8, $e_1 = \{1, 2, 4, 6, 7, 8\}$, $e_2 = \{1, 3, 4\}$, and $e_3 = \{4, 7, 8\}$.

With the aid of our previous discussion, we may now describe in some detail the context in which $\mathrm{cw}(H, E)$ arises. Let there be a partial linear layout L of a hypergraph $G \supset H$ whose domain does not include H and whose dangling edges include ∂H. Now absorb H into the interval $(j, j + 1)$ of L as previously described and call the resulting layout L'. The set E will consist of those edges in ∂H all of whose unassigned points lie in H and whose rightmost points (under L) are assigned no further right than j. Thus E is precisely the set of edges of dangling(L) which in L' become totally assigned and extend no further to the right than $\max\{L'(p): p \in H\} < L'(L^{-1}(j + 1))$. Now consider the set of intervals $S = \{I_t = (j + t, j + t + 1): 1 \leq t \leq |H| - 1\}$ "created" in L' by the absorption of H. Then we see that $\mathrm{cw}(H, E)$ is the maximum contribution to $|\mathrm{cut}(I_t, L')|$, over all intervals

I_t in S, coming from the subhypergraph induced by H and E. The important consequence is that if L was k-plausible, then the k-plausibility if L' can be verified by checking that $\text{cw}(H, E) \leq k - \text{sum}(j) + |E|$, where here we assume that all edges have unit weight. We refer to the edges of E as *anchors* for H. When the set E is fixed by context, we write $anch(H, L')$ for $\text{cw}(H, E)$.

Utilizing the component absorption operation, we show in the next section how to define an equivalence relation R that satisfies the two properties mentioned earlier, namely,

(i) R has a polynomial number of equivalence classes and
(ii) whenever (L, L') is in R, then either both L and L' can be extended to a complete cutwidth $\leq k$ layout of G or neither of them can.

(A similar approach has been used recently to obtain a polynomial-dynamic programming algorithm for answering the Search Number $\leq k$ problem, for each $k \geq 1$ [EST].) As the cutwidth of a hypergraph is the maximum cutwidth of any of its connected components, we assume, without loss of generality, that each hypergraph considered by the algorithm is connected. Also, for simplicity we use the terms layout and partial layout interchangeably. For emphasis, a layout of the entire hypergraph will be called a *total layout.*

2. Definitions and Equivalence Relation

We begin with the definitions necessary to define an appropriate equivalence relation on partial layouts. (We assume throughout the remainder of this paper that all hyperedge weights are uniformly one. That is, we only describe an algorithm explicitly for the unweighted case. However, the technique is easily extended to the case with nonuniform hyperedge weights.)

Let L be a partial layout of a hypergraph H. Let $\text{dangling}(L) = \{e_1, e_2, \ldots, e_m\}$. For each e_i in $\text{dangling}(L)$, let $Left_i$ (respectively, $Right_i$) be the vertex of e_i with smallest (resp. largest) assigned value. In other words, Left_i (Right_i) is the leftmost (rightmost) vertex of e_i under L. We also write this as $left(e_i, L)$ or $left(I, L)$ (resp. $right(e_i, L)$ or $right(I, L)$), where I denotes the interval $(i, i + 1)$. We assume that the indexing of the hyperedges $e_1, \ldots, e_m$ is in nondecreasing order of the leftmost vertices. Let $Left(L)$ be the vector $(\text{Left}_1, \ldots, \text{Left}_m)$. (We often do not distinguish between the vertices and the integers they are assigned to, e.g., Left_i will denote both the leftmost vertex of the ith dangling hyperedge and the integer Left_i is mapped to by the layout L. The context should make the meaning clear.) For any two values x and y in the range of L, say with $x \leq y$, the subset $\{z : x \leq z \leq y\}$ of $\text{range}(L)$ is called a *segment* of L and is denoted (x, y). Again if $x', y' \in \text{domain}(L)$ with $L(x') \leq L(y')$, then we may write (x', y') for the segment $(L(x'), L(y'))$.

For any interval $J = (j, j + 1)$, where j is in the range of L, recall that $\text{sum}(j)$ is the number of hyperedges passing over $(j, j + 1)$. Let $absorb(J, L)$, or simply $absorb(j)$ when L is understood, be k-$\text{sum}(j)$, where k is the bound on cutwidth for which we are testing. We can think of $\text{absorb}(J, L)$ roughly as an upper bound on the cutwidth of any component that can be absorbed into the interval $(j, j + 1)$

without making the cutwidth of the expanded layout larger than the overall bound k. (As noted, we actually have to be more careful in computing the absorption possible in a partial layout L. When a component is absorbed it may result in some of L's dangling hyperedges having all of their vertices assigned and, if the rightmost vertices of these hyperedges lie in the component absorbed, then these hyperedges will no longer pass over intervals to the right. Hence a component absorbed into the interval $(j, j+1)$ can under these circumstances have cutwidth larger than that specified by absorb(j).)

Let *Components*(L, i), or simply *Components*(i) when the layout L is understood, for each integer i ($1 \leq i \leq m$), be the set of connected components in the hypergraph $H - \{e_1, e_2, \ldots, e_i\}$, where $\{e_1, e_2, \ldots, e_i\}$ is the subset of dangling(L) consisting of the first i dangling hyperedges. We say that a subhypergraph A is a *forward i-component of the partial layout L* if:

(1) A is in Components(i) and i is the smallest integer for which this holds, and
(2) all vertices of A are in domain(L) and have values larger than Left$_i$ (and hence larger than Left$_t$, for all $t < i$).

We let *forward components*(i) be the set of all forward i-components of L. Notice that if A is a forward i-component, then $A \cap e_t = \varnothing$ for $t \geq i + 1$. This follows since the condition $A \cap e_t \neq \varnothing$ is inconsistent with (2) in the definition of a forward i-component and the definition of a dangling hyperedge. Let *Forward-Components*(L) denote the union of *forward components*(i), over all i.

Consider now the set of all rightmost and leftmost points of all dangling edges of L. Remove from S all points which lie in some element of Forward Components(L), and call the points remaining the *distinguished points of* L. The vector $(x_1, x_2, \ldots, x_s)$ of distinguished points of L, indexed by the order in which they are assigned from left to right in L (with x_1 being leftmost), will be denoted by *Points*(L). A segment of range(L) whose leftmost and rightmost points are successive elements of Points(L) or whose leftmost point is x_s will be called a *range* of L. We denote by *range*(I, L) the range of L containing an interval I, and for $A \subset H$ we let *range*(A, L) be the range of L (if it exists) containing the set $L(A)$. (Thus, for example, range(A, L) $= (x_1, x_2)$ means that $L(A)$ is contained entirely in the segment of range(L) lying between x_1 and x_2.) For each range R of L let absorb(R) $= \max\{$absorb(I, L): range(I, L) $= R\}$ and let *interval*(R) be the rightmost interval I for which range(I, L) $= R$ and absorb(I, L) $=$ absorb(R). We refer to an interval I such that $I =$ interval(R) for some range R as a *distinguished interval of* L. The distinguished, interval lying in the range having x_r as its leftmost point is denoted by $P_r(L)$, or just by P_r when L is understood. The vector (Gap$_1$, Gap$_2$, $\ldots$, Gap$_s$), where *Gap*$_i$ $=$ absorb(P_i, L), is denoted by *Gap*(L).

The purpose of defining the above objects is to help define the equivalence relation mentioned previously, and to facilitate a "reorganization" of L into a certain "canonical" form. For the idea behind the latter purpose, consider a forward component C of L. It may be "spread out" over a large part of range(L) and in that way contribute to the cut of many intervals. An improvement in the ability of L to absorb components, and thereby be completable, might come from "compressing" the layout of C into some interval J of L; that is, removing C and then reabsorbing it into J. We choose a distinguished interval as this J (details are given later). After performing this reabsorption for *every* forward component (using possibly different distinguished intervals for different components), we obtain the desired canonical layout L''. Now consider the possible absorption of some element D of Components(i, L'') $\cap$ unassigned(L'') into some interval K of L''. The question of whether this absorption preserves k-plausibility depends partially on the number of edges in dangling(L'') $\cap\, \partial D$ that can be used as anchors for this embedding (see the discussion on cw(H, E)). We scan the layout L'' to find all points right(e, L), $e \in$ dangling(L'') $\cap\, \partial D$, satisfying right(e, L) $\leq$ left(K). If such an e also satisfies [$e \cap$ unassigned(L'')] $\subset D$, then e is in fact an anchor for this absorption. In effect, e stops contributing to the cut set of all intervals to the right of K. The vector Gap(L'') tells us the absorb values of the distinguished intervals into which absorption will take place, in particular absorb(K). This, a knowledge of the anchors for D relative to K, and cw(H, E) (for the appropriate E) tells us whether the k-plausibility preserving absorption is possible.

For each i, $1 \leq i \leq m$, let $Back_i$ be the set of hyperedges e in H satisfying:

(1) all verticles of e are assigned under L and
(2) e passes over the interval (Left$_i - 1$, Left$_i$).

Let *Back*(L) be the vector (Back$_1$, Back$_2, \ldots,$ Back$_m$). We need to ensure that equivalent partial layouts L and L' not only have the same set of dangling hyperedges, but that corresponding dangling hyperedges have the same leftmost vertices in both L and L'. Moreover, we require that the set of hyperedges passing over the intervals just to the left of the position of the leftmost vertices in corresponding dangling hyperedges be identical in L and L'. The reason for this is that we want, if L and L' are equivalent, their domains to differ only with respect to which connected components have been included and which have not. (That is, connected components obtained through deleting some number of the dangling hyperedges.) Furthermore, when a connected component C, say in the hypergraph obtained by deleting the first i dangling hyperedges, is included in the domain of L, but not in the domain of L', then we want to be able to deduce that C is laid out in L entirely to the right of the leftmost vertex of the ith dangling hyperedge and hence can also be absorbed into the partial layout L'. By requiring Back(L) = Back(L'), when L and L' are equivalent, we ensure that any such component in domain(L), but not in domain(L'), is laid out entirely to the right of Left$_i$.

Finally, for each layout L we combine the invariants introduced above into the 4-tuple (dangling(L), Points(L), Back(L), Gap(L)) which we denote by *certificate*(L).

As stated, the purpose of defining absorb(j) and then Gap(L) is to measure the maximum cutwidth of any subhypergraph H' that can be "adsorbed", i.e., assigned, in the interval ($j, j+1$) after this interval ($j, j+1$) after this interval has been dilated by an appropriate factor, while still retaining a cutwidth k plausible layout of H (for some fixed k). Clearly, each hyperedge e in cut(j) is an obstacle to this absorptive capacity. This method of enlarging a partial layout will now be made more precise.

We now define the operation of *component absorption*, also called component insertion, corresponding in our discussion above to absorbing a subhypergraph into some interval. Assume first that $(j, j+1)$ is an interval with j and $j+1$ both in domain(L). We say that a partial layout L' is obtained from L by *dilating* $(j, j+1)$ if domain(L') = domain(L) and if, for some integer d, we have the following for all x in domain(L):

(1) if $L(x) \leq j$, then $L'(x) = L(x)$
(2) if $L(x) > j$, then $L'(x) = L(x) + d$.

Notice that cutwidth(L') = cutwidth(L), independent of d and j. Assume that i is the smallest integer such that A is in Components(i) and $A \subseteq$ unassigned(L). Let J be the interval $(j, j+1)$ of L. Denote by *Terminate*(A, L, J) the set $\{e_t$ in dangling(L)$|t \leq i$, all vertices in e_t that are unassigned in the partial layout L are in A and Right$_t \leq j\}$. That is, Terminate(A, L, J) is the set of all dangling hyperedges of L that (a) will no longer be dangling after the absorption of A and (b) have their current rightmost vertex somewhere to the left of J. We also let *Terminate*(A, L) be the set of dangling edges e of L satisfying $[e \cap \text{unassigned}(L)] \subset A$, so that Terminate($A$, L, J) $\subset$ Terminate(A, L). The edges of Terminate(A, L) will be totally assigned no matter what interval of L we absorb A into.

Now let E = Terminate(A, L, J). Then we denote by $(L * A)_J$ the partial layout obtained from L by inserting a layout of A into the dilated interval $(j, j+d+1)$ of L, where $d = |A|$, in which A's vertices are placed in the order given by an E-anchored layout of $A(E)$ with anchored cutwidth at most $|E|$ + absorb(J, L). Note that if Terminate(A, L, J) = $\varnothing$, then we can use a minimum cutwidth layout of A in the dilated interval, provided this cutwidth is at most absorb(J, L). See Figure 4. Observe that if cutwidth(L) $\leq k$, then cutwidth($(L * A)_J$) $\leq k$, where k is the bound on cutwidth that is being tested. Also note that component absorption is only defined for components A all of whose vertices are unassigned. When J is the leftmost interval of L for which $(L * A)_J$ is defined, then we write $(L * A)$ for $(L*A)_J$.

We now define the canonical form, which we call *normal form*, for partial layouts alluded to earlier. The basic idea is that any partial layout L can be placed in normal form by removing all of its forward components and then reabsorbing them into the various distinguished intervals of L. Changing a partial layout to this kind of normal form could affect it ability to absorb components, but only by making it easier. That is, an arbitrary partial layout may have forward components spread out in less than optimum patterns. We now give the details of the process of converting a partial layout L to one in normal form.

Let $L(r)$ called *the reduced layout of* L, be the partial layout obtained from L by deleting from the domain of L all vertices in Forward Components(L). ($L(r)$ lays out the remaining vertices in the same order they are laid out by L.) Order the forward components of L arbitrarily by $C_1, C_2, \ldots, C_t$. We define a sequence of layouts as follows. Let $F_0 = L(r)$ and, inductively for $1 \leq i \leq t$, let $F_i = (F_{i-1} * C_i)_{J_i}$ where J_i is the leftmost distinguished interval of F_{i-1} for which the indicated absorption can be defined. We then define the *normal form of* L, denoted *Norm*(L), by Norm(L) = F_t.

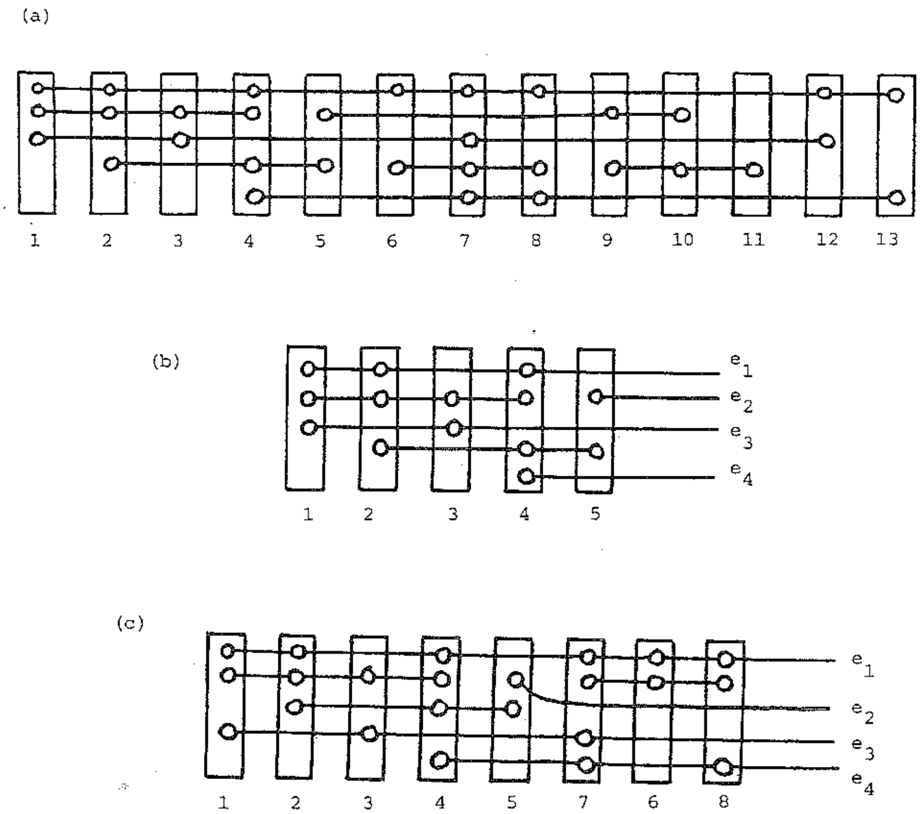

Fig. 4. (a) A hypergraph $H = \{\{1, 2, 4, 6, 7, 8, 12, 13\}, \{1, 2, 3, 4\}, \{1, 3, 7, 12\}, \{2, 4, 5\}, \{4, 7, 8, 13\}, \{5, 9, 10\}, \{6, 7, 8\}, \{9, 10, 11\}\}$, (b) a partial layout L of H, and (c) the partial layout $(L^*A)_J$. Here J is the interval $(5, 6)$ of L, and A is a connected component of $H - \{e_1, e_3, e_4\}$ consisting of the vertices $\{6, 7, 8\}$. Also $e_1 = \{1, 2, 4, 6, 7, 8, 12, 13\}$, $e_3 = \{1, 3, 7, 12\}$, and $e_4 = \{4, 7, 8, 13\}$.

The existence of the J_i essentially follows from the fact that the C_i were all originally embedded in L. More formally, we may see this as follows. Starting with $i = 1$, we consider an interval B of L for which the contribution to cut(B, L) from edges of C_1 is a maximum. Let I be the smallest interval of $L(r)$ containing B in L; that is, $I = (z, z+1)$ where B is contained in the segment $(L(L(r)^{-1}(z)), L(L(r)^{-1}(z+1)))$ of L. Also let R be the range of $L(r)$ corresponding to range(B, L). Then $I \subset R$ and absorb$(R, L(r)) \geq$ absorb$(I, L(r)) \geq$ absorb(B, L) + cutwidth$(C_1, L) >$ cutwidth(C_1, L). This shows that C_1 can be absorbed into *some* distinguished interval of $L(r)$ (namely, interval(R)). Thus J_1 exists and the distinguished intervals of F_1, call them $P_c(1)$ for $1 \leq c \leq s$, are related to the ones P_c of F_0 by absorb$(P_c(1), F_1)$ = absorb(P_c, F_0). This argument can now be iterated t times to show that J_i exists for all i.

We remark for the future reference that the layouts L, $L(r)$, and Norm(L) have the same sequence of dangling edges and the same sequence of distinguished points (equivalently, the same Points vector).

We now define an equivalence relation R on normal form partial layouts L_1 and L_2 of a hypergraph H as follows:

$$(L_1, L_2) \text{ is in } R \quad \Leftrightarrow \quad \text{certificate}(L_1) = \text{certificate}(L_2). \qquad [*]$$

As will be seen, this relation has a polynomial number of equivalence classes.

The following lemma shows that putting a layout into normal form never hurts its completability.

Lemma 0. *Let L be a partial layout of a hypergraph H. Then L is completable to a k-plausible total layout of $H \Rightarrow$ Norm(L) is completable to a k-plausible total layout of H.*

Proof. Let $L_1, L_2, \ldots, L_t$ be a sequence of k-plausible partial layouts of H with L_t being total, such that L_i is obtained from L_{i-1} by vertex addition or component insertion for all i. We simulate this sequence by constructing a sequence $S_1 =$ Norm(L_1), $S_2, \ldots, S_t$ of partial layouts of H with S_t being total. Our simulation is as follows.

Let $X = \{x_1, x_2, \ldots, x_s\}$ be the sequence of distinguished points in Norm(L) and let $P_r = P_r(0)$ be the distinguished interval of Norm(L) in the range $(L(x_r), L(x_{r+1}))$, $1 \leq r \leq s$, where $L(x_{s+1})$ is taken as $|\text{domain}(L)| + 1$.

(1) If $L_i = L_{i-1} + x$, then $S_i = S_{i-1} + x$.
(2) If $L_i = [L_{i-1} * D]_J$, then:
   (a) If J is to the right of $\max\{L_{i-1}(x)\colon x \in \text{domain}(L)\}$, then

   $$S_i = [S_{i-1} * D]_J.$$

   (b) Otherwise.

Let c be the integer such that J lies in the segment $(L_{i-1}(x_c), L_{i-1}(x_{c+1}))$ of L_{i-1}.

$S_i = [S_{i-1} * D]_{P_c(i-1)}$.
$M \leftarrow \max\{S_i(p)\colon p \in D\}$.
$P_c(i) \leftarrow$ the interval $(M, M+1)$ of S_i. (Update of intervals $P_r(i)$, $r \leq s$.)
$P_d(i) \leftarrow P_d(S_i)$ for $d \neq c$.

(*Comment*: D has been absorbed in the dilated interval P_c, and the new P_c becomes the interval just to the right of $\max\{S_i(p)\colon p \in D\}$. The other P_d, $d \neq c$, are left the same except for the obvious translation; that is, if P_d comes to the left of P_c, then it remains the same, while if P_d comes to the right of P_c, then it is translated by an amount $|D|$ to the right corresponding to the dilation required to absorb D.)

We now show inductively that the partial layouts S_i constructed by this simulation are k-plausible, from which the theorem follows. As observed previously, $S_1 =$ Norm(L) is k-plausible. Now assume inductively that S_{i-1} is k-plausible.

Assume first that $L_i = L_{i-1} + x$, so that by (1) we have $S_i = S_{i-1} + x$. Now the cuts of intervals to the left of x in S_i remain the same as they were in S_{i-1}, so by induction these have sizes at most k. As for the interval to the right of x in S_i, the fact that domain(S_i) = domain(L_i) for all i shows that its cut is the same as the cut of the interval to the right of x in L_i. Since L_i is k-plausible by assumption, this interval (in S_i) has cut size at most k. Hence S_i is k-plausible.

We may therefore assume that $L_i = [L_{i-1} * D]_J$ for some unassigned component D and interval J of L_{i-1}. We must show that the corresponding absorption into the interval, call it J', of S_{i-1} specified in (2) yields a k-plausible layout S_i. Note that $J' = J$ in step (2a) and $J' = P_c(i-1)$ in (2b). To do this, it suffices to show the following for all i:

(a) $|\text{cut}(J', S_i)| \leq |\text{cut}(J, L_i)|$ and
(b) the anchors used in the layout of D in J are also available as anchors for the layout of D in J', that is, Terminate(D, L_i, J) $\subseteq$ Terminate(D, S_i, J').

We prove (a) and (b) by induction. Assume first that $i = 1$ and consider (a). For any interval J of L, let $J(r)$ be the interval of $L(r)$ corresponding to J. That is, $J(r)$ is the interval $(z, z+1)$ of $L(r)$ such that J is contained in the segment $(L([\text{Norm}(L)]^{-1}(z)), L([\text{Norm}(L)]^{-1}(z+1)))$ of L. Clearly, $|\text{cut}(J, L)| \geq |\text{cut}(J(r), L(r))| \geq |\text{cut}(\text{interval}(\text{range}(J(r))), L(r))|$. But the last quantity is $|\text{cut}(P_c, \text{Norm}(L))|$, where P_c is the distinguished interval of Norm(L) into which D will be absorbed by step (2a). Hence (a) is proved.

For (b), let e be an anchor for D in L with right(e, L) < left(J). If $x \in$ domain($L(r)$), then $x \in X$ so that range(J, L) is right of x in L. Hence range($J(r), L(r)$) and consequently P_c is also right of x in Norm(L), so (b) follows. Assume then that $x \notin$ domain($L(r)$). Then x is in some forward component F of L. The definition of Norm(L) then guarantees that range(x, Norm(L)) is no further to the right than range(x, L). Hence if range(x, L) is to the left of range(P_c, Norm(L)), then (b) follows immediately. Now range(x, L) cannot be right of range(P_c, Norm(L)) (since x is left of J), so we may assume that range(x, L) = range(P_c, Norm(L)). But then F is one of the forward components absorbed into interval(range($J(r)$), $L(r)$) in the process of forming Norm(L). Since P_c is by definition the interval just to the right of these components in Norm(L), it follows that x is left of P_c in Norm(L). Hence (b) follows, so the base of the induction is complete.

Proceeding inductively, assume $i > 1$ and that (a) and (b) are true for S_{i-1} and L_{i-1}. In addition to (a) and (b), we need the following statement for all i. Its proof is also inductive.

(c) For each c, we have $|\text{cut}(P_c(i), S_i)| \leq |\text{cut}(J, L_i)|$ for any interval J of L_i in the segment $(L_i(x_c), L_i(x_{c+1}))$.

We note that the validity of (c) for S_1 = Norm(L) is immediate by definition of Norm(L). Following the structure of our simulation, it will be convenient to partition the range of L_{i-1} into two segments:

(i) the first segment $T_1 = \{z: z \leq \max\{L_{i-1}(x): x \in \text{domain}(L)\}$ and
(ii) the final segment $T_2 = \text{range}(L_{i-1}) \setminus T_1$.

A simple induction shows that, for all i,

$$L_i^{-1}(T_1) = S_i^{-1}(T_1), \qquad L_i^{-1}(T_2) = S_i^{-1}(T_2), \tag{$*$}$$

and

$$S_i(x) = L_i(x) \qquad \text{for any} \quad x \in S_i^{-1}(T_2). \tag{$**$}$$

Assume first that J is in the first segment. Observe that (c) implies (a) under this assumption since whenever J lies in the range $(L_{i-1}(x_c), L_{i-1}(x_{c+1}))$ the simulation in step (2b) gives $J' = P_c(i-1)$. Hence the inductive step for (a) will follow from the inductive step for (c). To do the latter, we apply the inductive hypothesis for (b) and conclude that the absorption of D into $P_c(i-1)$ uses at least as many anchors as the absorption of D into J. Hence at least as many dangling edges are terminated by the former as by the latter. Now the maximum difference between $|\text{cut}(P, L_{i-1})|$ and $|\text{cut}(Q, L_{i-1})|$, over all intervals P and Q in L_{i-1}, is at most $|\text{terminate}(D, L_{i-1}, J)|$. But this difference is actually achieved between $|\text{cut}(P_c(i-1), S_{i-1})|$ and $|\text{cut}(P_c(i), S_i)|$ from the above. Thus $|\text{cut}(P_c(i), S_i)|$ has the minimality property (c). Now $|\text{cut}(P_d(i-1), S_{i-1})| - |\text{cut}(P_d(i), S_i)| = |\text{terminate}(D, L_{i-1}, J)|$ for $d > c$ since $P_d(i)$ is to the right of $P_c(i)$, so these $P_d(i)$ satisfy (c). For $d < c$ we have $|\text{cut}(P_d(i-1), S_{i-1})| - |\text{cut}(P_d(i), S_i)| = 0 = |\text{cut}(P, L_{i-1})| - |\text{cut}(Q, L_i)|$ for intervals P to the left of $L_{i-1}(x_c)$ and Q to the left of $L_i(x_c)$. Thus again these $P_d(i)$ satisfy (c). Hence the inductive step for (c) is completed, and the same step for (a) follows.

We now prove the inductive step for (b) when J is in the first segment. Let $L_{i+1} = [L_i * E]_J$ for some interval J of L_i, and let e be an anchor for E in this absorption. If $x = \text{right}(e) \in X$, then essentially the same proof as given for $i = 1$ shows that e is also an anchor for E in S_i. If $x \notin X$, then x is in some forward component F for some L_{t-1} such that $L_t = [L_{t-1} * F]_G$, where $t < i$ and G is an interval of L_{t-1}. Hence by our simulation F was absorbed in $P_b(t)$ (in S_t) for some b. By the updating step we find that x is left of $P_b(i)$ in S_i. But J belongs to a segment $(L_i(x_c), L_i(x_{c+1}))$ for some c, so the fact that $x \le \text{left}(J)$ in L_i implies $b \le c$. Now the simulation embeds E into $J' = P_c(i)$, and $P_c(i)$ is right of $P_b(i)$ since $b \le c$. Hence x is left of J' in S_i, and the inductive step is completed.

Assume now that J is in the final segment (so step (2a) is executed), and again let $L_{i+1} = [L_i * E]_J$. Then the statements $(*)$ and $(**)$ earlier in the proof, the inductive hypothesis, and the fact that E is absorbed into the same interval in both L_i and S_i immediately imply (a) and (b) for L_i and S_i. Also (c) holds for L_i and S_i using induction and the fact that there is no change in the cut sets of intervals in the first segment in going from L_{i-1} to L_i or S_{i-1} to S_i because they are all left of J. □

Given a vertex x in $V(H)$, where x is not in domain(L), we write $L + x$ for the partial layout, called a *vertex addition of* L, satisfying $\text{domain}(L + x) = \text{domain}(L) \cup \{x\}$, $(L + x)(v) = L(v)$, for all v in domain(L), and $(L + x)(x) = M + 1$, where M is the maximum value in the range of L. Clearly, we can compose the operations of vertex addition and component absorption arbitrarily many times. We say that a partial layout $L^{(1)}$ is an *expansion* of L if $L^{(1)}$ can be obtained from L by a sequence of component absorptions and vertex additions. If there exists an expansion $L^{(1)}$ of L which is total (i.e., domain$(L^{(1)}) = H$) and satisfies cutwidth$(L^{(1)}) \leq k$, then we say *L can be completed with cutwidth k.*

3. Fundamental Results

A necessary idea in the proof of validity of our algorithm for hypergraph cutwidth is that the operation of component absorption does not hurt the completability of a partial layout. This is the subject of the next lemma.

Lemma 1. *Let L be a partial layout of a hypergraph H. Assume $(L * A)_J$ is k-plausible for some A and J, and let $\Psi = (L * A)$. Then Ψ can be completed with cutwidth $k \Leftrightarrow L$ can be completed with cutwidth k.*

Proof. Note that the $\Rightarrow$ direction follows trivially, since the first step of a cutwidth k completion of L, when given a cutwidth k completion of $L*A$, can always be simply to absorb the component A.

Assume that L can be completed with cutwidth k and let $L = L_1, L_2, \ldots, L_t = L^C$ be a sequence of partial layouts such that L^C is a total layout of the hypergraph H with cutwidth k, and each L_i results from vertex addition or component applied to its predecessor L_{i-1}.

We need the following operation on any partial layout T and $B \subset$ domain(T). We let T/B be the partial layout satisfying domain$(T/B) =$ domain$(T)\backslash B$ in which points of domain $(T)\backslash B$ are mapped to the segment $[1, |\text{domain}(T)\backslash B|]$ in the same relative order they were mapped in T. Note that $B \subset$ unassigned(T/B) and the set $\{e\colon e \in \partial B \cap \text{domain}(T)\backslash B\}$ becomes a subset of dangling(T/B). We refer to T/B as "T modulo B."

We "simulate" the given sequence of partial layouts L_i by a sequence of partial layouts $L * A = \Psi_1, \Psi_2, \ldots, \Psi_s$, $s \leq t$, resulting in a total layout Ψ_s. Before defining the simulation formally, we give a brief overview of its main ideas.

The goal is to construct the Ψ_i inductively by performing the vertex additions and component absorptions defining the L_i in the same order and, as much as possible, in the "same" location. Let $I = (x, x + 1)$ be the interval of L at which A was absorbed, and let $L_i = (L_{i-1} * D)_J$ for some component D and interval J of L_{i-1}. The simulation requires a corresponding absorption $\Psi_i = (\Psi_{i-1} * D)_{J'}$, and the issue is to define J' so that the resulting Ψ_i is k-plausible. This definition takes one of two possible directions, depending upon the placement of A in L_{i-1} relative to $L_{i-1}(L^{-1}(x))$:

(i) If not all of A has yet been assigned, or if A is assigned and placed to the right of $L_{i-1}(L^{-1}(x))$, then it can be shown that $\Psi_i/A = L_i/A$ for all i. The simulation then defines J' to be the same interval of Ψ_i/A as J. Also Ψ_i is then "corrected," if necessary, by reabsorbing A into a position corresponding to the one it had in L, namely, just to the right of the image of $L^{-1}(x)$. The result is that Ψ_i and L_i map the points of $H\backslash A$ in the same relative order, differing only in their placement of A. The "correction" applied to Ψ_i guarantees that A will be placed in Ψ_i to the left of its placement in L_i. These facts then give an easy argument that Ψ_i is k-plausible.

(ii) Assume that A has been assigned by L_{i-1} to the left of $L_{i-1}(L^{-1}(x))$. Let $m_{i-1} = \min\{L_{i-1}(p): p \in A\}$. We partition the range of L_{i-1} into three segments, $(1, m_{i-1} - 1)$, $(m_{i-1}, \text{left}(G_{i-1}))$, and $(\text{right}(G_{i-1}), |\text{domain}(L_{i-1})|)$ listed in order from first to third, where G_{i-1} is a special "garbage collection" interval. The identical partition of Ψ_{i-1} into three segments may also be defined (where m_{i-1}, left(G_{i-1}), and right (G_{i-1}) are viewed as integers in the range of Ψ_{i-1}). When J is in the initial or last segment of L_{i-1}, then the simulation lets $J' = J$. With this it can be shown inductively that L_t and ψ_t are the same for all t on the first and terminal segments in the sense that they map to these segments the same set of points in the same relative order. When J is in the middle segment it may happen that the interval of Ψ_{i-1} corresponding to J may not be able to absorb D. In that case a "garbage collection" interval G_{i-1} at the extreme right of the middle segment of Ψ_{i-1} will be able to accept this absorption, and the simulation then lets J' be this G_{i-1}. The layouts L_i and Ψ_i resulting in this case is illustrated in Figure 5. Components absorbed to the left of $L_{i-1}(L^{-1}(x))$ in L_{i-1} (like C_3) which could not be absorbed into the corresponding interval of Ψ_{i-1} are instead absorbed into the garbage collection interval of Ψ_{i-1}. The new garbage collection interval G_i in Ψ_i is then updated to be just to the right of these components.

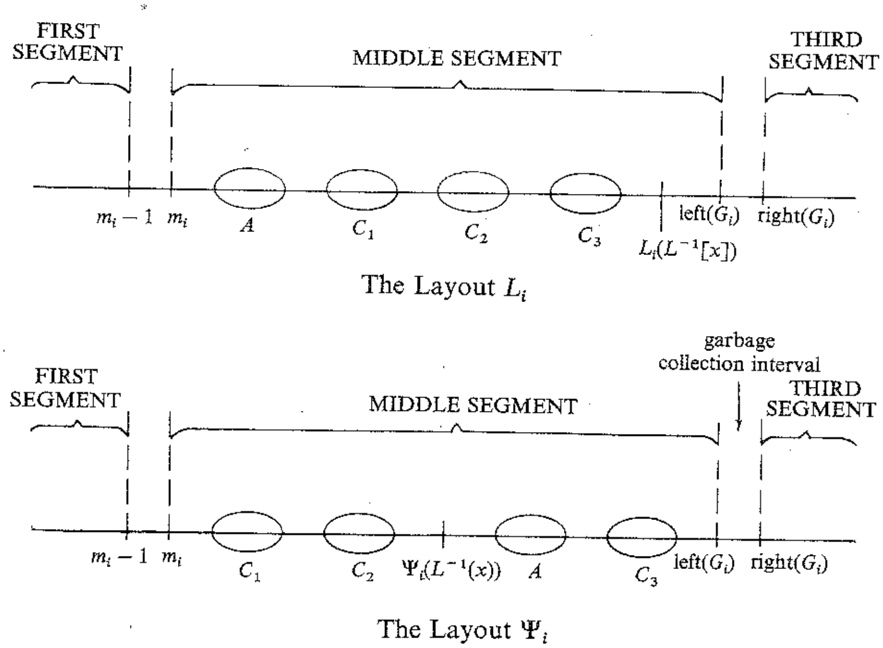

Fig. 5. A comparison of L_i and ψ_i.

Our simulation is as follows:

(1) If $L_i = L_{i-1} + z$, then $\Psi_i = \Psi_{i-1} + z$ if $z \notin A$, and $\Psi_i = \Psi_{i-1}$ if $z \in A$.
(2) Assume $L_i = (L_{i-1} * D)_R$ for some D and interval R of L_{i-1}.

Let $I = (x, x+1)$ be the interval of L at which A was absorbed to form $\Psi_1 = (L * A)$. Initialize the garbage collection interval at $G_1 = (M, M+1)$, where $M = \max\{\Psi_1(p): p \in A\}$. Let $P = (y, y+1)$ be the interval in L_{i-1}/A equivalent to R; that is, R is contained in the segment

$$(L_i((L_{i-1}/A)^{-1})(y), L_i((L_{i-1}/A)^{-1}(y+1))).$$

Let $Y' = L^{-1}(y)$, $Y'' = L^{-1}(y+1)$, and $X = L^{-1}(x)$.

(2.1) If $D \subseteq A$, then $\Psi_i = \Psi_{i-1}$.
(2.2) If $D \not\subseteq A$ (so $D \cap A = \varnothing$ as will be seen below), then:
(a) If $L_t \neq (L_{t-1} * A)_S$ for any S and all $t < i$, then execute Procedure **Absorb(*D, i*)**
(b) If $L_t = (L_{t-1} * A)_S$ for some $t < i$, then execute Procedure **Segment(*D, i*)**.

(Procedure **Absorb(*D, i*)**)
If R is outside the segment $(L_{i-1}(L^{-1}(x)), L_{i-1}(L^{-1}(x+1)))$, **then**

$$\Psi_i = (\Psi_{i-1} * D)_{(\Psi_{i-1}(Y'')-1,\, \Psi_{i-1}(Y''))}.$$

(*Comment*: We form Ψ_i by absorbing D into the same interval mod A as it was absorbed in L_{i-1}. The assumption on R implies that D will be assigned to either the first or third segment of Ψ_i.)

Otherwise (so R is inside the segment $(L_{i-1}(L^{-1}(x)), L_{i-1}(L^{-1}(x+1)))$)

$$\Psi_i = ((\Psi_{i-1}/A * D)_{((\Psi_{i-1}/A)(Y'')-1,\, (\Psi_{i-1}/A)(Y''))} * A)_{(\Psi_{i-1}(X),\, \Psi_{i-1}(X)+1)}.$$

(*Comment*: Again we place D in Ψ_{i-1}/A as it would be placed in L_{i-1}/A. After "reinserting" points of A to their old locations we would find that D may now be assigned somewhere between the min and max of the set $\{\Psi_{i-1}(p): p \in A\}$. As an aid in the analysis we avoid this possibility by reabsorbing A back into the interval $(\Psi_{i-1}(X), \Psi_{i-1}(X)+1)$, so that all points between the above-mentioned min and max are images from A. The placement of D in $((\Psi_{i-1}/A)(Y'')-1, (\Psi_{i-1}/A)(Y''))$ ensures that it will always be in the third segment of all subsequent Ψ_r, $r \geq i$.)

$$M = \Psi_i(\Psi_{i-1}^{-1}(\text{left}(G_{i-1}))).$$

$$G_i = (M, M+1).$$

(end Procedure **Absorb(*D, i*)**)

(Procedure **Segment(*D, i*)**)
Let $m_{i-1} = \min\{L_{i-1}(p): p \in A\}$.
If R is to the right of $L_{i-1}(L^{-1}(x))$, or R is to the left of m_{i-1}, or $m_{i-1} > L_{i-1}(L^{-1}(x)$ **then**

(*Comment*: The following instructions will assign D to either the first or third segments of Ψ_i in case $m_{i-1} < L_{i-1}(L^{-1}(x))$. If this is not the case, then D will be assigned to the same interval in Ψ_{i-1} as R, modulo A.)

$\Psi_i = (\Psi_{i-1} * D)_{(\Psi_{i-1}(Y'')-1,\, \Psi_{i-1}(Y''))}$.

$M = \Psi_i(\Psi_{i-1}^{-1}(\text{left}(G_{i-1})))$

$G_i = (M, M+1)$

(*) **otherwise** (embedding D into the middle segment of Ψ_{i-1})
Let $F = (w, w+1)$ be the interval of Ψ_{i-1} contained in the segment $(\Psi_{i-1}(Y'), \Psi_{i-1}(Y''))$ such that P is contained in the segment $((L_{i-1}/A)(\Psi_{i-1}^{-1}(w)), (L_{i-1}/A)(\Psi_{i-1}^{-1}(w+1)))$.

(*Comment*: F corresponds to R and to P in Ψ_{i-1}.)

If the layout $(\Psi_{i-1} * D)_F$ is k-plausible, **then**
$\Psi_i = (\Psi_{i-1} * D)_F$
$M = \Psi_i(\Psi_{i-1}^{-1}(\text{left}(G_{i-1})))$
$G_i = (M, M+1)$

otherwise
$\Psi_i = (\Psi_{i-1} * D)_{G_{i-1}}$ (garbage collection step)
$M = \max\{\Psi_i(p) \colon p \in D\}$
$G_i = (M, M+1)$

(end Procedure **Segment(D, i)**)

First we show that the partial layouts Ψ_i, for all i, are well defined, i.e., that the vertex z, described in (1), is not in the domain of the partial layout Ψ_{i-1} and that D, described in (2), is a connected component of H-E, where E is some subset of the dangling hyperedges in the partial layout Ψ_{i-1}. After this we will show that the Ψ_i are k-plausible for all i.

For well definedness we begin by showing, using induction on i, that domain(Ψ_i) = domain(L_i) $\cup$ vertices(A). Clearly, this equality holds for $i = 1$. Assume that domain(Ψ_{i-1}) = domain(L_{i-1}) $\cup$ vertices(A). Note that there is nothing to show if $\Psi_i = \Psi_{i-1}$, since then the vertices added to L_{i-1} are elements of A. Assume first that $\Psi_i = \Psi_{i-1} + x$. Then $L_i = L_{i-1} + x$ and it follows, using the induction hypothesis, that domain(Ψ_i) = domain(Ψ_{i-1}) $\cup \{x\}$ = domain$\{L_{i-1}\}$ $\cup$ vertices(A) $\cup \{x\}$ = domain(L_i) $\cup$ vertices(A). Next assume that $\Psi_i = \Psi_{i-1} * D$. Then $L_i = L_{i-1} * D$ and, using the induction hypothesis, domain (Ψ_i) = domain(Ψ_{i-1}) $\cup$ vertices(D) = domain(L_{i-1}) $\cup$ vertices(A) $\cup$ vertices(D) = domain(L_i) $\cup$ vertices(A).

Notice that the vertex addition of z to Ψ_{i-1} is well defined, since this is done when z is not in domain(L_{i-1}), x is not in A and, hence, is not in domain(Ψ_{i-1}) = domain(L_{i-1}) $\cup$ vertices(A).

Now consider the insertion of D into partial layout Ψ_{i-1}. Notice that if vertices(D) is a subset of vertices(A), then there is nothing to prove, since no operation is performed in obtaining Ψ_i from Ψ_{i-1}. We assume therefore that vertices(D) is not a subset of vertices(A). For any partial layout L of a hypergraph H, let *components*(L) denote the set of all subhypergraphs C of H satisfying: (a) $C \subset$ unassigned(L) and (b) C is a connected component in the subhypergraph of H obtained from H by deleting some subset of dangling(L).

We need to show that D is in components(Ψ_{i-1}).

We begin by observing that $D \cap A = \varnothing$. Assume not. Then since D is not a subset of A (as noted above) and D is a connected component in components(L_{i-1}), it follows that there are vertices x in $A \cap D$, y in D-A, and a hyperedge e of D containing x and y. We claim that e is in dangling(L). The reason is that, since A is a connected component in the hypergraph obtained by removing some set of dangling edges from H, it follows that the only edges of H having points in A and vertices outside of A must be dangling edges. As e is a dangling hyperedge of the partial layout L, in each successive partial layout it must either remain a dangling hyperedge or all of its vertices must become part of the domain of a partial layout. Neither of these conditions are satisfied by the hyperedge e in the partial layout L_{i-1}. That is, all of e's vertices are in D, and D is entirely unassigned in L_{i-1}. ($D \subset$ unassigned(L_{i-1}) follows from the fact that, by definition, $L_i = L_{i-1} * D$ and component absorption is defined only for components all of whose vertices are unassigned.) From the contradiction we conclude that $D \cap A = \varnothing$.

Let dangling(L_{i-1}) $= X \cup Y$, where $Y =$ dangling(L_{i-1}) $\cap$ hyperedges(A) (and X is implicitly defined to be dangling(L_{i-1})$\setminus Y$). Then from the relation domain(Ψ_{i-1}) $=$ domain(L_{i-1}) $\cup$ vertices(A) it follows that dangling(Ψ_{i-1}) $\subset X$. See Figure 6. If $Y = \varnothing$, the same set of hyperedges in dangling(L_{i-1}) whose removal isolates D also isolates D in Ψ_{i-1}. Hence D is an element of components(Ψ_{i-1}) which is contained in unassigned(Ψ_{i-1}).

So assume that $Y \neq \varnothing$. Let S be a minimal subset of dangling(L_{i-1}) whose removal isolates D. Since $Y \subset$ hyperedges(A), we have $S \subset X$. In order to show D is in components(Ψ_{i-1}) we must prove that $S \subset$ dangling(Ψ_{i-1}). To see this, recall that domain(Ψ_{i-1}) $=$ domain(L_{i-1}) $\cup A$, so X-dangling(Ψ_{i-1}) is a set of edges whose only unassigned vertices are in A. But each element of S must have at least one vertex in D, and we know $D \cap A = \varnothing$. Thus, $S \cap (X\text{-dangling}(\Psi_{i-1})) = \varnothing$. Hence $S \subset$ dangling(Ψ_{i-1}), so we have D is in components(Ψ_{i-1}) as desired.

This completes the proof that the Ψ_i are well defined, and we now proceed to show that they are k-plausible.

We proceed by induction on i, the case $i = 1$ (where $\Psi_1 = L^*A$) being true by definition. Also recall that the partial layouts L_i are k-plausible for all i. Assume then that Ψ_{i-1} is k-plausible.

Assume first that $L_i = L_{i-1} + z$. If $z \in A$, so that $z \in$ domain(Ψ_{i-1}), then

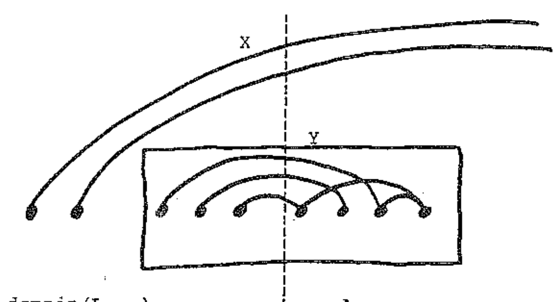

Fig. 6. The connected component A is partially in domain (L_{i-1}).

$\Psi_i = \Psi_{i-1}$ so there is nothing to prove. Assume then that $z \notin A$. Now each interval $Q = (q, q+1)$, $1 \leq q \leq |\mathrm{domain}(\Psi_i)| - 1$, satisfies $\mathrm{cut}(Q, \Psi_i) = \mathrm{cut}(Q, \Psi_{i-1}) \leq k$ by the inductive hypothesis. It remains to show that

$$|\mathrm{cut}((\Psi_i(z), \Psi_i(z) + 1), \Psi_i)| \leq k.$$

But the set enumerated on the left is identical with dangling(Ψ_i), and we have seen previously that dangling(Ψ_i) $\subset$ dangling(L_i) for all i. Hence Ψ_i is k-plausible.

Assume then that $L_i = (L_{i-1} * D)_J$ for some component D and interval J.

We begin with some preliminary reductions. Note that if A is never absorbed as a component in forming any of the L_i, $1 \leq i \leq t$ (so that A's points are mapped by vertex addition), or if A is absorbed by some L_i into an interval to the right of $L_i(L^{-1}(x))$, then the garbage collection step of the simulation is never executed. This leads to a simple inductive proof that the construction defining the Ψ_i can be carried out, yielding a k-plausible layout. Indeed we can prove that for all i:

(a) $L_i/A = \Psi_i/A$,
(b) the points of A are mapped consecutively by Ψ_i to the segment $(\Psi_i(L^{-1}(x))+1, \Psi_i(L^{-1}(x)) + |A|)$, and the obvious
(c) no point of A is mapped in L_i to the left of $L_i(L^{-1}(x))$.

Now each interval Q of Ψ_i in the segment $(\Psi_i(L^{-1}(x)), \Psi_i(L^{-1}(x)) + |A| + 1)$ satisfies $|\mathrm{cut}(Q, \Psi_i)| \leq k$ by (b) and the fact that the original layout $\Psi_i = L^*A$ was k-plausible. Now let $Q = (s, s+1)$ be any interval of Ψ_i lying outside this segment, and consider the segment $S = (L_i(\Psi_i^{-1}(s)), L_i(\Psi_i^{-1}(s+1)))$ of L_i. Then for any interval $Q' \subset S$ we have $\mathrm{cut}(Q', L_i) \supseteq \mathrm{cut}(Q, \Psi_i)$ by (a). (In fact, the containment is proper only when internal edges of A pass over Q'.) Hence $|\mathrm{cut}(Q, \Psi_i)| \leq k$ since L_i is k-plausible.

We are therefore reduced to the case where A is absorbed by some L_c into an interval B lying to the left of $L_c(L^{-1}(x))$. We remark that that this could happen, despite the choice of $(x, x+1)$ as the leftmost interval in L able to k-plausibly absorb A, because the successors L_i of L, $1 \leq i \leq c-1$, may involve component absorptions into intervals to the left of B which terminate edges. This may reduce the cut of B sufficiently so as to make the absorption of D possible in L_c.

First we note that for $i \leq c-1$ we see inductively that the domain of Ψ_i still does not include A and hence does not involve the garbage collection step. Thus as above we have $L_i/A = \Psi_i/A$ with the k-plausibility of Ψ_i as a consequence. We may therefore restrict our attention to the partial layouts Ψ_r, $r \geq c$, still assuming inductively that Ψ_b is k-plausible for $b \leq r-1$.

To analyze these Ψ_r, we partition the range of L_r into three segments. Let $m_r = \min\{L_r(p) : p \in A\}$. The segments are as follows:

the initial segment S_1, defined as $(1, m_r - 1)$,
the middle segment S_2, defined as $(m_r, \mathrm{left}(G_r))$,
the final segment S_3, defined as $(\mathrm{right}(G_r), |\mathrm{domain}(L_r)|)$.

The relation of L_r and Ψ_r on these segments is given by

(i) $\Psi_r^{-1}(S_i) = L_r^{-1}(S_i)$, $1 \leq i \leq 3$, and
(ii) for each $z \in \Psi_r^{-1}(S_1 \cup S_3)$, we have $\Psi_r(z) = L_r(z)$.

The proof is a straightforward induction based on the fact that a component absorption into an interval in $S_1 \cup S_3$ occurs only when the garbage collection step is not executed, from which (i) applied to $i = 1$ and 3 and (ii) follow. When the garbage collection step is executed, both L_r and Ψ_r absorb D into S_2, though into possibly different intervals, and this implies (i) applied to $i = 2$.

We can now prove that component absorptions defining Ψ_r, $r \geq c$, can be carried out while preserving k-plausibility by considering the segment of L_{r-1} to which the interval J (into which a component D is being absorbed) belongs. We will show that the corresponding interval, call it J', into which D will be absorbed to create Ψ_r (as specified by the simulation) has a sufficiently small cut set and can use at least the same set of anchors as did the embedding into J'. When J is in the initial or final segment, this can be done by comparing J' with J. When J is in the middle segment and $J' = G_{r-1}$, an argument based on the relation between the layouts L, L_r, and Ψ_r is given.

Assume that J belongs to the first segment. The simulation gives $J' = (\Psi_{r-1}(Y'')-1, \Psi_{r-1}(Y''))$. Conditions (i) and (ii) now imply that $J' = J$, and $\mathrm{cut}(J', \Psi_{r-1}) = \mathrm{cut}(J, L_{r-1})$. Also any anchors used by L_{r-1} when absorbing into J must also be available for Ψ_{r-1} when absorbing into J', since such anchors have their rightmost points left of J and hence by (i) and (ii) left of J'. The k-plausibility of L_r then implies the same for Ψ_r.

Assume now that J belongs to the final segment. The simulation gives $J' = (\Psi_{r-1}(Y'')-1, \Psi_{r-1}(Y''))$, and again (i) and (ii) imply that $J' = J$. First we argue that the same anchors used for absorbing D into J can also be used for absorbing it into J'. Consider first those anchors e satisfying $\mathrm{right}(e, L_{r-1}) \in S_1 \cup S_2$. The condition $\Psi_{i-1}^{-1}(S_1 \cup S_2) = L_{r-1}^{-1}(S_1 \cup S_2)$ implies that these also satisfy $\mathrm{right}(e, \Psi_{r-1}) \in S_1 \cup S_2$, and hence that they are anchors for absorbing into J'. As for anchors e of L_{r-1} satisfying $\mathrm{right}(e, \Psi_{r-1}) \in S_3$ conditions (i) and (ii) immediately show that they are also available for $J' = J$. Next we argue that $X_1 = \mathrm{cut}(J(\Psi_{r-1}), \Psi_{r-1})$ is sufficiently small by comparing it with $X_2 = \mathrm{cut}(J, L_{r-1})$. First note that (i) and (ii) imply that all edges $e \notin \partial A$ either contribute to both X_1 and X_2 or to neither. Now assume $e \in \partial A$. Then $e \notin X_1$ implies that e is totally assigned in L_{r-1} and $\mathrm{right}(e, L_{r-1})$ is to the left of J. But then the facts that J is the same as J' (since $J \in S_3$) and $\Psi_{r-1}(A) \subseteq S_2$ imply that $\mathrm{right}(e, \Psi_{r-1})$ is to the left of J'. Hence $e \notin X_1$, so $X_2 \subseteq X_1$. Thus the absorption into J' can be done with k-plausibility of Ψ_r preserved.

If J is between the first segment and the middle segment (i.e., $J = (m_{r-1} - 1, m_{r-1})$), then we proceed as if J were in the first segment, while if J is between the middle segment and the final segment (i.e., $J = G_{r-1}$), then we proceed as if J were in the final segment as above. We omit these details.

Finally, assume that J belongs to the middle segment, so that $(*)$ is executed. If D can be absorbed into the interval F initially attempted by the simulation yielding a k-plausible Ψ_r, then there is nothing to prove. We may therefore assume that D cannot be so absorbed, and assume for contradiction that the attempted absorption into G_{r-1} also fails to preserve k-plausibility. Let $F' = (f, f+1)$ be the interval of L such that F lies in the segment $(\Psi_{r-1}(L^{-1}(f)), \Psi_{r-1}(L^{-1}(f+1)))$. Thus F' corresponds in L to F and to J, while F corresponds in Ψ_{r-1} to J. Also let

$\alpha = |\{z: z = \text{right}(e, L) \text{ for some } e \in \text{terminate}(A, L), f+1 \leq L(z) \leq L(x)\}|$ and let β be the number of dangling edges e of L which are no longer dangling in L_{r-1} and such that right(e, L_{r-1}) is no further right than left(J). Let anch(D, L_{r-1}) be the anchored cutwidth of whatever anchored layout of D is used in absorbing D into J (where possibly the set of anchors is empty).

We now explore the consequences of our contrary assumption. Let $E =$ Terminate(D, L_{r-1}, J). Clearly,

$$\text{absorb}(J, L_{r-1}) \geq \text{anch}(D, L_{r-1}) - E$$

since L_r remains k-plausible. We claim G_{r-1} lies to the right of $r_e = \text{right}(e, \Psi_{r-1})$ for all edges $e \in \text{Terminate}(D, L_{r-1}, J)$. For certainly $L_{r-1}(\Psi_{r-1}^{-1}(r_e))$ must be to the left of J and hence belongs to $S_1 \cup S_2$. Thus condition (i) implies $\Psi_{r-1}(r_e) \in S_1 \cup S_2$, which forces r_e to be no further right than $\max\{\Psi_{r-1}(S_1 \cup S_2)\} = \text{left}(G_{r-1})$ and the claim follows. It follows that a layout of D in G_{r-1} could use the same set of anchors (and possibly more) as the anchored layout of D in J. The contrary assumption that a k-plausibility preserving absorption of D into G_{r-1} is not possible therefore says that

$$\text{anch}(D, L_{r-1}) > \text{absorb}(G_{r-1}, \Psi_{r-1}) + |E|.$$

Now the edges counted in β constitute the set $\text{cut}(F', L)\backslash\text{cut}(J, L_{r-1})$. Hence $|\text{cut}(F', L)| \leq \beta + |\text{cut}(J, L_{r-1})|$, and it follows that

$$\begin{aligned}\text{absorb}(F', L) &= k - |\text{cut}(F', L)|\\ &\geq k - (\beta + |\text{cut}(J, L_{r-1})|)\\ &= \text{absorb}(J, L_{r-1}) - \beta.\end{aligned}$$

Similarly the edges counted in $\beta + \alpha$ are contained in the set $\text{cut}(I, L)\backslash\text{cut}(G_{r-1}, \Psi_{r-1})$. Now recall that G_1 is initialized to be just right of $\Psi_1(A)$ and G_i is updated to be just right of the most recently absorbed component into G_{i-1}. Thus no edge inside a component absorbed in the construction of the L_i, contributes to $\text{cut}(G_i, \Psi_i)$. This leads to a simple inductive proof that $\text{cut}(G_{r-1}, \Psi_{r-1}) \subseteq \text{cut}(I, L)$. Hence

$$|\text{cut}(I, L)| \geq \beta + \alpha + |\text{cut}(G_{r-1}, \Psi_{r-1})|,$$

and therefore

$$\begin{aligned}\text{absorb}(I, L) &= k - |\text{cut}(I, L)|\\ &\leq k - (\beta + \alpha + |\text{cut}(G_{r-1}, \Psi_{r-1})|)\\ &= \text{absorb}(G_{r-1}, \Psi_{r-1}) - \beta - \alpha.\end{aligned}$$

Combining these inequalities we get

$$\begin{aligned}\text{absorb}(F', L) + \beta &\geq \text{absorb}(J, L_{r-1})\\ &\geq \text{anch}(D, L_{r-1}) - |E|\\ &> \text{absorb}(G_{r-1}, \Psi_{r-1})\\ &\geq \text{absorb}(I, L) + \beta + \alpha.\end{aligned}$$

Thus

$$\text{absorb}(F', L) > \text{absorb}(I, L) + \alpha. \tag{1}$$

Now Terminate$(A, L, I)\backslash$Terminate(A, L, F') has at most α elements. Inequality (1) therefore shows that a k-plausible layout results from absorbing into the interval F' of L the layout of A used in I, even at the cost of using the members of Terminate$(A, L, I)\backslash$Terminate(A, L, F') as edges passing over the entire dilated F'. This contradicts the definition of I as the leftmost interval of L into which A can be k-plausibly absorbed. This completes the proof of Lemma 1. □

Our next lemma justifies the use of our equivalence relation R.

Lemma 2 (Main Lemma). *Let H be a hypergraph. Let L_1 and L_2 be normal form partial layouts of H such that (L_1, L_2) is in R (as defined in [*]). Then, L_1 can be completed with cutwidth $k \Leftrightarrow L_2$ can be completed with cutwidth k.*

Proof. First observe that $(L_1, L_2) \in R$ implies that the Points, Gap, and dangling entries of certificate$(L_1(r))$ and certificate$(L_2(r))$ are the same since removal of forward components from a normal form layout does not affect these entries.

Next observe that domain$((L_1(r))$ = domain$(L_2(r))$. For this it suffices to show that the symmetric difference of domain(L_1) and domain(L_2) lies in forward components$(L_1) \cup$ forward components(L_2). This in turn is equivalent to showing that for any j we have

$$\begin{aligned}&[\text{components}(j, L_1) \cap \text{unassigned}(L_1) \cap \text{domain}(L_2)]\\ &\quad \subset \text{forward components}(L_2)\end{aligned} \tag{*}$$

with a similar statement holding where L_1 and L_2 are interchanged.

Consider then an arbitrary component B belonging to the left-hand side of (*). Since B is unassigned in L_1, no edge of B can appear in any coordinate of the vector Back(L_1). It follows that no edge of B can appear in any coordinate of the vector Back(L_2) since $(L_1, L_2) \in R$ means that L_1 and L_2 must have the same certificate. Thus, as desired, B must be a forward component of L_2 since by assumption it belongs to domain(L_2) and has no edge in Back(L_2).

Now this fact and the relation between certificate$(L_1(r))$ and certificate$(L_2(r))$ given above imply that $L_1(r)$ is completable to a total k-plausible layout of H if and only if $L_2(r)$ is. This can be proved by a straightforward simulation. Assume, for example, that $L_1(r)$ is completable and that $L_1^1 = L_1(r), L_1^2, \ldots, L_1^t$ is a sequence of k-plausible partial layouts, each L_1^i being an extension of L_1^{i-1}, ending in a total layout L_1^t. A corresponding sequence of extensions $L_2^1 = L_2(r), L_2^2, \ldots, L_2^t$ is defined by letting $L_2^i = L_2^{i-1} + x$ if $L_1^i = L_1^{i-1} + x$, and $L_2^i = (L_2^{i-1} * D)_{J'}$ if $L_1^i = (L_1^{i-1} * D)_J$, where J' is the distinguished interval of L_2^{i-1} in range(J, L_1^{i-1}). This sequence is well defined since a simple induction shows that domain(L_1^i) = domain(L_2^i) and the sequences of distinguished points of L_1^i and L_2^i are the same. In addition we can prove as part of the induction that the absorb value of any distinguished interval in L_2^i majorizes the absorb values of all intervals in the corresponding range in L_1^i. We omit the details as they are similar to the proof of

Lemma 0. A symmetric argument with the roles of L_1 and L_2 interchanged may be given.

We now proceed to show that: L_i is completable to a total k-plausible layout $\Leftrightarrow L_i(r)$ is so completable, $i = 1, 2$. The $\Rightarrow$ direction is obvious since L_i is an extension of $L_i(r)$. For the $\Leftarrow$ direction (with $i = 1$), let forward components$(L_1) = \{C_1, C_2, \ldots, C_s\}$. Also let $F_0 = L_1(r), F_1, F_2, \ldots, F_s$ be the sequence of extensions used in constructing $L_1 = \text{Norm}(L_1)$ from $L_1(r)$, where $F_i = (F_{i-1} * C_i)_{J_i}$ and J_i is the leftmost distinguished interval of F_{i-1} into which C_i may be k-plausibly absorbed.

We begin by proving that F_1 is completable if $L_1(r)$ is. Now by Lemma 1 we know that $(L_1(r) * C_1)_J$ is completable, where J is the leftmost interval of $L_1(r)$ into which C_1 can be k-plausibly absorbed. By definition of normal form we know interval(range$(J, L_1(r))) = J_1$ since C_1 is assigned to the leftmost possible range of $L_1(r)$ (subject to k-plausibility) in constructing L_1 from $L_1(r)$. Thus $\text{Norm}((L_1(r) * C_1)_J) = F_1$ since dangling$(L_1(r))$ = dangling$((L_1(r) * C_1)_J)$. Now since normal form preserves completability by Lemma 0, it follows that F_1 is completable.

This argument may be iterated s times to give the completability of L_1. The key idea is that at each step we have dangling$((F_{i-1} * C_i)_K)$ = dangling(F_i), where K is the leftmost interval of F_{i-1} into which C_i can be k-plausibly absorbed. This gives $\text{Norm}((F_{i-1} * C_i)_K) = F_i$, which in combination with Lemmas 0 and 1 imply the completability of F_i.

Finally, we combine our observations to get

$$\begin{aligned} L_1 \text{completable} &\Leftrightarrow L_1(r) \text{ completable} \\ &\Leftrightarrow L_2(r) \text{ completable} \\ &\Leftrightarrow L_2 \text{ completable,} \end{aligned}$$

completing the proof of the lemma. □

4. Description of Algorithm

We now give a description of the algorithm, called cutwidth(H, k, E), which determines whether a hypergraph H has E-anchored cutwidth at most k. Roughly speaking, by examining in our algorithm at most one partial layout from each equivalence class we indirectly examine all possible layouts that can be completed with cutwidth k. Our main lemma is the justification for this. The algorithm builds successively larger expansions of a partial layout until it reaches an expansion whose certificate has already been considered. Since cutwidth(H, k, E) carries out component absorptions, it calls itself recursively to determine if the component can be absorbed without exceeding cutwidth k.

We use two main data structures. One is a stack whose elements are pairs of the form (certificate(L), L). Since cutwidth(H, k, E) wants to examine at most one layout in each class, we also require a boolean function Table of one argument: a certificate of a partial layout. Table answers "yes" iff its argument represents an equivalence class that has been previously considered.

Boolean procedure cutwidth(H, k, E)

begin / let $E = \{e_1, e_2, \ldots, e_m\}$, let $H(E)$ be the E-augmentation of H with new vertex z, and let L_0 be the partial layout of $H(E)$ such that $L_0(z) = 1$. Note that if $E = \varnothing$, then L_0 is the partial layout with the empty set for its domain. /

add (certificate(L_0), L_0) to stack; set Table(certificate(L_0)) to yes;

/ A vertex is initially added in order to begin the layout of H and to avoid getting into a loop i.e., trying to absorb the same component over and over again. /

for every vertex x not in domain(L_0) **do**
begin
$L \leftarrow L_0 + x$;
$L \leftarrow$ Norm(L);
compute certificate(L);
if L is k-plausible and Table(certificate(L)) = no
then
begin add (certificat(L), L) to stack;
set Table(certificate(L)) to yes
end
end

while stack is not empty **do**
begin
remove top pair (certificate(L), L) from stack;
if dangling(L) = $\varnothing$ and $L \neq L_0$ **then** return true;
/ $H(E)$ has E-anchored cutwidth $\leq k$ /
for $i = 1$ to k **do**

(1.1) **for** each component A of Components(i) $\cap$ unassigned(L) **do**
begin

/ We will determine whether A can be absorbed into some distinguished interval of L (and the latter may be assumed to be in normal form). Thus let P be a distinguished interval of L with absorb(P) = r. Let E = Terminate(A, L, P). Note that membership in E can be determined just by scanning L. We want to compute the E-anchored cutwidth of $A(E)$, because it is precisely the set of hyperedges E that will no longer extend to the right of A when A is absorbed into P. We repeat this computation over all distinguished intervals P until the leftmost one is found, if it exists, into which A can be absorbed. /

(1.2) **If** cutwidth($A, r + |E|, E$) = true, **then** $L \leftarrow L * A$;
$L \leftarrow$ Norm(L)

/ If the statement $L \leftarrow L * A$ is executed, then A has been absorbed as an i-forward component of L. Indeed, A will be absorbed into the leftmost interval J (not necessarily distinguished) of L which preserves k-plausibility. The points of A are laid out in the order given by the layout provided by the call to the procedure cutwidth($A, r + |E|, E$). The new vertex z is discarded. Note that the hyperdeges in E will be totally assigned and will connect A to vertices in domain L which lie to the left of J. /

```
(1.3)         end
(2.1)   for every vertex x not in domain(L) do
            begin
(2.2)       L ← L + x;
(2.3)       L ← Norm(L);
(2.4)       compute certificate(L);
(2.5)       if L is k-plausible and Table(certificate(L)) = no
              then
(2.6)         begin add (certificate(L), L) to stack;
(2.7)           set Table(certificate(L)) to yes
              end
(2.8)       end
          end
return false
end
```

We call the process indicated in lines (1.1)–(1.3) of the algorithm "Stage 1" and the process indicated in lines (2.1)–(2.8) "Stage 2."

Theorem 1. *The algorithm is correct.*

Proof. It is clear that if cutwidth(H, k, E) answers "yes," then indeed $H(E)$ has an E-anchored cutwidth k, since the algorithm actually produces an E-anchored layout L with cutwidth $\leq k$.

Suppose that the procedure answers "no." First we remark that if a partial layout L of H can be completed with cutwidth $\leq k$, then the same is true for Norm(L). Thus it is sufficient to consider, as the algorithm does, only layouts in normal form. We need to show that $H(E)$ does not have an E-anchored layout with cutwidth k. Assume to the contrary that there exists a total cutwidth $\leq k$ E-anchored layout of $H(E)$. Since the algorithm answers "no" the stack must have been emptied. Let $M = \max\{|\text{domain}(L)| \mid L$ is a k-plausible E-anchored layout considered in Stage 2 of the algorithm$\}$. This would include all pairs that ever appear on the stack, as well as all pairs discarded because their Table value is "yes." Note that, since the stack empties, $M < |\text{vertices}(H)|$.

Given a partial layout L, consider the layout $L\langle\text{insert}\rangle$, which is obtained from L by performing in sequence all possible absorptions of components in Components(i), $1 \leq i \leq m$. We call a layout L' a *successor* of L if $L' = L\langle\text{insert}\rangle + x$, i.e., if L' is obtained by vertex addition of some vertex to $L\langle\text{insert}\rangle$. Note that $(L\langle\text{insert}\rangle)\langle\text{insert}\rangle = L\langle\text{insert}\rangle$, since by definition all possible absorptions are performed to get $L\langle\text{insert}\rangle$ and dangling($L\langle\text{insert}\rangle$) is a subset of dangling(L). Also note that if L is on the stack, then $L\langle\text{insert}\rangle$ is found in Stage 1 and therefore each of L's successors are examined during Stage 2.

Claim. *If L can be completed with cutwidth k, then $L\langle\text{insert}\rangle$ can be completed with cutwidth k and therefore some successor of L can be completed with cutwidth k.*

Proof of Claim. By an iterative use of Lemma 1 we see that $L\langle\text{insert}\rangle$ can be completed with cutwidth $\leq k$. Since $L\langle\text{insert}\rangle$ can be completed with cutwidth k and can experience no component absorptions, it follows that some vertex addition of $L\langle\text{insert}\rangle$ (which is a successor of L) can be completed with cutwidth k. (End of proof of claim.)

We now show that, for each partial layout of L that can be completed with cutwidth k and which is considered by the algorithm, there is a layout $L^{(1)}$ that can be completed with cutwidth k, which is considered by the algorithm, and satisifies $|\text{domain}(L^{(1)})| > |\text{domain}(L)|$. We consider two cases:

Case 1. L is placed on the stack at the end of some execution of Stage 2 of the algorithm.

Then eventually L is taken off the stack and passed through stages 1 and 2 of the algorithm and, consequently, all of its successors are considered. By the above claim at least one of the successors of L can be completed with cutwidth k. That successor has a larger domain than L (as do all of L's successors). (End of the proof of Case 1.)

Case 2. L is not placed on the stack at the end of some execution of Stage 2. Then Table(certificate(L)) is "yes." So there must be a layout Q which is either on the stack or was previously on the stack such that certificate(Q) = certificate(L). By Lemma 2, Q can be completed with cutwidth k. When Q is processed through Stages 1 and 2 of the algorithm, all successors of Q are considered. By an argument similar to that used in Lemma 2, $|\text{domain}(Q\langle\text{insert}\rangle)| \geq |\text{domain}(L)|$ and therefore $|\text{domain}(Q\langle\text{insert}\rangle + x)| > |\text{domain}(L)|$, for all x. Thus each successor of Q has larger domain than L, can be completed with cutwidth k by the claim above, and is considered by the algorithm (after an execution of Stage 2). (End of proof for Case 2.)

The argument is concluded as follows: Start with a layout L that can be completed with cutwidth k, which appears on the stack, and is therefore considered by the algorithm (the partial layout L_0 will do). By applying the above observation iteratively we get a sequence of partial layouts that can be completed with cutwidth k, which are considered by the algorithm, and have increasing domain sizes. The sequence can only break off when a total layout of H is reached, a contradiction to the existence of an upper bound M on the size of considered partial layouts. □

5. Time Complexity

Theorem 2. *The running time of the algorithm is* $O(n^m)$, *where* $m = k^2 + 3k + 3$.

Proof. For reasons of efficiency we replace the recursive step, i.e., the call to cutwidth($A, r + |E|, E$) in line (1.2) of the algorithm by a "table look up." We develop a table, call it Look(k), which contains, for every possible set E of at most k hyperedges, every possible connected component A of the hypergraph obtained by deleting the hyperedges in E from the given hypergraph H, and for every possible subset E' of E, the E'-anchored cutwidth of A. Having Look(k) available, the insertion step can be done by looking up cutwidth($A, r + |E'|, E'$) in the table rather than executing the procedure. We now explain how the table Look(k) is obtained.

Look(k) is constructed by an iterative approach. We first remove each subset E of $k - 1$ hyperedges from H and determine which connected components of $H - E$ have cutwidth 1. (This takes linear time, as noted by Cahoon and Sahni [CS1]). Next we remove each subset E of k-2 hyperedges and run our algorithm on the connected components A of $H - E$ and thereby determine which of them have cutwidth 2. We need only the previous information put into Look(k), since the algorithm for cutwidth 2 only makes a recursive call by deleting one hyperedge and asking for cutwidth 1. Continuing in this way we add to Look(k) the cutwidth information about components obtained by deleting $k - j$ hyperedges, which we can do by using already-computed information in Look(k) for components obtained through deleting $k - j + t$ hyperedges, for $t \geq 1$.

We may thus consider for removal the set of all possible hyperedge subsets of size j, for $j < k$. This set has size $O(n^k)$. For each subset of hyperedges deleted we run the algorithm on at most $O(n)$ components in the resulting hypergraph. The running of the algorithm may result in the consideration of one representative from each equivalence class, and for each considered representative there is an amount of time in computing new representative layouts and their certificates. The time needed for a single new layout and its certificate is $O(n)$. Hence the total update time is $O(n^2)$, since there are $O(n)$ new layouts. The number of equivalence classes is the product of the number of choices for dangling(L), Points(L), Back(L), and Gap(L), which are bounded by $O(n^k)$, $O(n^{2k})$, $O(n^m)$, where $m = k^2$, and k^{2k}, respectively. Since k is fixed, the total running time of the algorithm is $O(n^p)$, where $p = k^2 + 3k + 3$.

References

[CS1] J. Cahoon and S. Sahni, Exact algorithms for special cases of the Board Permutation Problem, *Proc. 21st Ann. Allerton Conf. on Communication, Control, and Computing* (1983), pp. 246–255.

[CS2] J. Cahoon and S. Sahni, Heuristics for the Board Permutation Problem, *Proc. IEEE Int. Conf. on Computer-Aided Design* (1983), pp. 81–83.

[Ce] I. Cederbaum, Optimal Backboard Ordering Through the Shortest Path Algorithm, *IEEE Trans. Circuits and Systems*, **21** (5) (1974), 626–632.

[Ch] F. R. K. Chung, On the cutwidth and the Topological Bandwidth of a Tree, *SIAM J. Algebraic Discrete Methods*, **6** (1985), 268–277.

[CMST] 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.

[EST] J. A. Ellis, I. H. Sudborough, and J. S. Turner, Graph Separation and Search Number, to appear.

[FL] M. R. Fellows and M. A. Langston, Fast Self-Reduction Algorithms for Combinatorial Problems of VLSI Design, *Proc. Third Aegean Workshop on Computing: VLSI Algorithms and Architectures*, Lecture Notes in Computer Science, Vol. 319, Springer-Verlag, Berlin (1988), pp. 278–287.

[GJ] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979).

[GCT] S. Goto, I. Cederbaum, and B. S. Ting, Suboptimum Solution of the Backboard Ordering Problem with Channel Capacity Constraint, *IEEE Trans. Circuits and Systems*, **24** (11) (1977), 645–652.

[GS] E. Gurari and I. H. Sudborough, Improved Dynamic Programming Algorithms for Bandwidth Minimization and the Min Cut Linear Arrangement Problem, *J. Algorithms*, **5** (1984), 531–546.

[J] D. S. Johnson, The Many Faces of Polynomial Time *NP*-completeness Column: An Ongoing Guide, *J. Algorithms*, **8** (1987), 285–303.

[L] J. T. Li, Algorithms for Gate Matrix Layout, *Proc. IEEE Int. Symp. on Circuits and Systems* (1983), pp. 1013–1016.

[MPS] F. Makedon, C. H. Papadimitriou, and I. H. Sudborough, Topological Bandwidth, *SIAM J. Algebraic Discrete Methods*, **6** (3) (1985), 418–444.

[MS] F. Makedon and I. H. Sudborough, Minimizing Width in Linear Layouts, *Proc. 10th ICALP*, Lecture Notes in Computer Science, Vol. 154, Springer-Verlag, Berlin (1983), pp. 478–490.

[MHG*] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, The Complexity of Searching a Graph (Preliminary Version), *Proc. IEEE FOCS* (1981), pp. 376–385.

[M] Z. Miller, A Linear Algorithm for Topological Bandwidth in Degree 3 Trees, *SIAM J. Comput.*, **17** (5) (1988), 1018–1035.

[MoS] B. Monien and I. H. Sudborough, Min Cut is *NP*-Complete for Edge Weighted Trees, *Proc. 1986 ICALP*, Lecture Notes in Computer Science, Vol. 26, Springer-Verlag, Berlin (1986), pp. 265–274.

[RS1] N. Robertson and P. D. Seymour, Graph Minors, XIII: The Disjoint Paths Problem, to appear.

[RS2] N. Robertson and P. D. Seymour, Graph Minors, XVI; Wagner's Conjecture, to appear.

[SS] A. Sangiovanni-Vincentelli and M. Santomauro, A Heuristic Guided Algorithm for Optimal Backboard Wiring, *Proc. 13th Ann. Allerton Conf. on Circuits and Systems* (1975).

[S] L. Steinberg, The Back Board Wiring Problem: A Placement Algorithm, *SIAM Rev.*, **3** (1) (1961), 37–50.

[Y] M. Yannakakis, A Polynomial Algorithm for the Min Cut Linear Arrangement of Trees, *J. Assoc. Comput. Mach.*, **32** (4) (1985), 950–959.

[W] O. Wing, Automated Gate Matrix Layout, *Proc. IEEE 1982 Symp. on Circuits and Systems* (1982), pp. 681–685.

Received December 26, 1987, and in revised form November, 1988, and January, 1990.