A Polynomial Algorithm for Recognizing Small Cutwidth in Hypergraphs

by Z. Miller and I. H. Sudborough

Department of Computer Science
University of Texas at Dallas
Richardson, Texas 75083

(1) On sabbatical from Department of Mathematics, Miami University, Oxford, Ohio

Abstract: The Min Cut Linear Arrangement (Min Cut) problem for hypergraphs was previously considered by Cahoon and Sahni [CS], where it was called the Board Permutation problem (BP). They gave $O(n)$ and $O(n^3)$ algorithms for determining cutwidth 1 and 2, respectively, and cited the open problem: Is there an algorithm that determines in $O(n^{ck})$ time if a hypergraph has cutwidth $k$? We describe an $O(n^m)$ algorithm, with $m=k^2+3k+3$, which determines if a hypergraph has cutwidth $k$. The Min Cut or BP problem, where one wishes to minimize "backplane area" in automating circuit design, is the subject of several recent papers [CS2], [Y], [W], [L], [S], [GCT], [C], [GS].

I. Introduction.

We consider linear arrangements of circuit boards or gates for the sake of minimizing channel width. This problem, which for graphs is called the Min Cut Linear Arrangement problem [GJ], [CMST], [Y], and for hypergraphs is called the Board Permutation problem [S], [CS], [CS2], is applicable to designing Gate Matrices [L], [W], obtaining an optimal backboard ordering [C], [GCT], [SS], and is related to several graph problems [MHGJP], [MS], [Ch], [M], [MPS], [MoS].

Let $G$ be a finite hypergraph with an 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 $\{1, \ldots, \mid\text{vertices}(G)\}$ ( $L$ is a partial layout if it maps a subset $V$ of vertices($G$) to $\{1, \ldots, \mid V\}$). Consider, for a given layout (or partial layout) $L$ of $G$ and integer $i$, the set $\text{cut},L(i) = \{A : A \text{ is a hyperedge which contains both (1) a vertex mapped by } L \text{ to an integer } \leq i \text{ and (2) a vertex either not in the domain of } L \text{ or mapped by } L \text{ to an integer }> i \}$. (We will denote this set by $\text{cut}(i)$, when $G$ and $L$ are understood.) Let $\text{sum},L(i)$ denote the summation of $\{w(A) : A \text{ is in } \text{cut},L(i)\}$. The cutwidth of $G$ under a linear layout $L$, denoted by $\text{cutwidth}(G,L)$, is $\max\{\text{sum},L(i) : 1 \leq i \leq \mid\text{vertices}(G)\}\}$. We say that a partial layout $L$ is $k$-plausible if cutwidth($G,L$)$\leq k$. The Weighted Min Cut $< k$ problem.

Input: A finite hypergraph $G$ and an integer weight $w(A)$ associated with each hyperedge $A$.

Question: Is cutwidth($G$) $< k$?

We will for the most part describe results for the Min Cut $< k$ problem, where the edge weights are uniformly one, although our results generalize easily. Figure 1 describes a finite hypergraph and its cutwidth under two linear layouts.

The Weighted Min Cut $< k$ problem, when restricted to graphs, can be solved in $O(n^{k-1})$ steps, for all $k \geq 1$, where $n$ is the number of
vertices in the graph [MS]. (This improved an earlier O(n^k) result [GS].) The Min Cut Linear Arrangement (Min Cut) problem, where the bound on the cutwidth is part of the input instead of being fixed, is known to be NP-complete [GJ] and to remain NP-complete for planar graphs with maximum vertex degree 3 [MoS]. (We refer here to the unweighted problem.) The Min Cut problem can be solved in O(n log n) time for trees [Y]. (An earlier O(n log^{d+1} n) algorithm for the Min Cut problem on trees with maximum vertex degree d is described in [CMST], where a characterization is given and used to equate cutwidth and search number [MHGJP] for trees with maximum vertex degree three. 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].

Let G be a finite hypergraph and 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 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 \( \min \{ \text{cutwidth}(G,L) : L \text{ is an A-anchored layout of } G \} \). When the vertices in A are not connected to each other, their relative order in an A-anchored layout isn't important. Consequently, in such cases we replace the vector A by a unordered set A. Observe that the cutwidth of a hypergraph G, as defined earlier, is simply \( \emptyset \)-cutwidth(G). Let E be a set of hyperedges of G. Removing the hyperedges in E from G results in a new, possibly disconnected, hypergraph, say G-E. Consider a connected component \( H \) of G-E. For any subset \( E' = \{ e_1, \ldots, e_s \} \) of E the \( E' \)-augmentation of \( H \), denoted by \( H(E') \), is the hypergraph with (1) the vertices: \( \text{vertices}(H) \cup A \), where \( A = \{ x_1, \ldots, x_s \} \) is a set of s new vertices, and (2) the hyperedges:

\[ \text{hyperedges}(H) \cup \{ \{ x_i, y_1, \ldots, y_t \} : 1 \leq i \leq s \} \]

where \( \{ y_1, \ldots, y_t \} = e_i \). That is, \( H(E') \) is the hypergraph obtained by adding, for each edge \( e_i \) in E' one new vertex \( x_i \) and a new hyperedge \( e_i' \) connecting the new vertex \( x_i \) with all vertices that were part of the deleted hyperedge \( e_i \). We shall use for simplicity the term \( E' \)-anchored cutwidth of \( H(E') \) to denote the A-anchored cutwidth of \( H(E') \), where A is the set of vertices added to create the hyperedges E'. We shall need the notion of anchored cutwidth later.

Our algorithm for the Min Cut \( \leq k \) problem uses dynamic programming. To get a rough idea consider first a straightforward procedure which tries every possible partial layout. Let \( \text{domain}(L) \), for any partial layout L, be the set of vertices of the hypergraph that are mapped by L to some integer. Similarly, let \( \text{range}(L) \) denote the set of integers mapped to. A hyperedge \( e = \{ x_1, x_2, \ldots, x_k \} \) is dangling (from a partial layout L) if there exist i and j such that \( x_i \) is in \( \text{domain}(L) \) and \( x_j \) is not. Let \( \text{dangling}(L) \) denote the set of edges dangling from L. We describe the straightforward process here only to make easier our descriptions later. Let "stack" denote a pushdown stack and \( L_0 \) denote the partial layout whose domain is the empty set:

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