Balancing Communication in Ring-Structured Networks

Ami Litman & Arnold L. Rosenberg

CMPSCI Technical Report 93-80
November, 1993

Balancing Communication in Ring-Structured Networks

Ami Litman
Department of Computer Science
The Technion
Haifa, Israel

Arnold L. Rosenberg
Department of Computer Science
University of Massachusetts
Amherst, Mass., USA

Abstract. We present a polynomial-time algorithm for finding an almost optimal solution to the following combinatorial problem, which is motivated by the problem of balancing communication in networks having the structure of rings. *Input:* a circle and a multiset of c chords, having n distinct endpoints, representing desired point-to-point communications; c' of the chords are distinct. *The task:* to map each chord to an arc having the same endpoints, in a way that minimizes the maximum number of arcs crossing any normal to the circle. Our algorithm operates in time $O(n^2c)$ and uses $O(n + c')$ words of storage; it finds a mapping that is within $+1$ of optimal.

1 Introduction

1.1 Motivation

We study a combinatorial problem related to the task of balancing communication in networks having the structure of a ring of nodes. Such networks arise in at least two situations. In one scenario, the ring represents a multiprocessor network with a reconfigurable multi-line bus interconnecting the processors (which are the nodes of the ring); see, e.g., [3, 5, 7]. In another scenario, the nodes of the ring are transceivers, and the ring structure arises from the connection pattern of the point-to-point transmission "cables" (say, using fiber-optic technology); see, e.g., [2]. In either of these scenarios, the ring structure carries with it several benefits. First, ring structures are biconnected: there are two distinct paths connecting any two nodes; this property lends a measure of fault tolerance to the network, as well as avoiding the bottlenecks caused by unique paths. Biconnectivity gives the ring a distinct advantage over tree structures. Second, the ring structure is easily clustered in a variety of ways; this enhances both the ease of packaging the ring (which is particularly useful in the multiprocessor metaphor where one might seek scalable multi-chip realizations of large rings) and creates the possibility of endowing the ring with "shortcuts;" both of these measures enhance the efficiency of the structure;

cf. [1, 8]. Finally, ring structures are sparse and planar, allowing them to share with tree structures efficiency of construction.

Our combinatorial problem takes the following form. We are given a ring of nodes, together with a set of chords in the ring that represent point-to-point connections that are to be established. Our task is to transform each of the given chords into an arc of the ring having the same endpoints as the chord. Our goal is to accomplish this in a way that minimizes the *width* of the resulting configuration, namely, the largest number of arcs that cross above any part of the ring (i.e., that are cut by a line normal to the ring). This cost measure corresponds to the number of lines that the ring's bus must support in the multiprocessor scenario or the number of frequencies that the cable must carry in the communication network scenario. Note that we are assuming that only the number of tracks or frequencies is essential, not specifically which ones; this means that we are assuming that message packets can changes tracks (in the multiprocessor scenario) or frequencies (in the communication network scenario) in transit, as the packets pass through ring nodes. We show here that this combinatorial problem can be solved to within an error of $+1$ in low-degree polynomial time in the sizes of the network and the message set. Our algorithm could be of interest in situations where there is a repertoire of communication patterns that can be precomputed; such situations abound in the literature; cf. [6, 10, 11].

1.2 The Formal Problem

Ring Structures. An *n-node ring structure* $\mathcal{R}$ is a graph-theoretic object having the following components. $\mathcal{R}$ has:

- *n regular nodes*, $r_0, r_1, \ldots, r_{n-1}$;
- *n auxiliary nodes* (*x-nodes*, for short), $x_0, x_1, \ldots, x_{n-1}$;
- $2n$ *spine edges* that connect each regular node r_i to x-nodes x_{i-1} and x_i;[1]
- a multiset of *interval hyperedges* (*i-hyperedges*, for short) $M(\mathcal{R})$, each i-hyperedge being a simple set[2] of the form

$$\{r_i, x_i, r_{i+1}, x_{i+1}, r_{i+2}, x_{i+2}, \ldots, x_{j-1}, r_j\}.$$

  We denote each such i-hyperedge by the ordered pair of regular nodes $\langle r_i, r_j \rangle$ that are its *endpoints* when it is traversed in clockwise order.[3]

[1] Here, and in the sequel, all arithmetic on node indices is modulo n.

[2] The simplicity of these sets precludes i-hyperedges that "wrap around on themselves."

[3] Our ring structures generalize to ring-spines the interval hypergraphs of [9].

Remarks. Every x-node of a ring structure is an endpoint of two spine edges; no x-node is an endpoint of an i-hyperedge. The regular nodes of a ring structure represent the processors (in the multiprocessor scenario) or the transceivers (in the communication network scenario); the x-nodes are purely a technical device that facilitates the development of our study.

The *width of ring structure $\mathcal{R}$ at x-node x*, denoted $\mathcal{W}(\mathcal{R}, x)$, is the number of i-hyperedges that contain x (as sets). The *width of ring structure $\mathcal{R}$*, denoted $\mathcal{W}(\mathcal{R})$, is the largest width of $\mathcal{R}$ at any of its x-nodes:

$$\mathcal{W}(\mathcal{R}) = \max\{\mathcal{W}(\mathcal{R}, x) \ : \ x \text{ is an x-node of } \mathcal{R}\}.$$

An x-node x is *critical* for $\mathcal{R}$ if $\mathcal{W}(\mathcal{R}, x) = \mathcal{W}(\mathcal{R})$.

Communication Patterns. An *n-node communication pattern P* is a circle, together with a multiset of chords having at most n distinct endpoints. (The fact that we have a *multiset* of chords means that we allow several "messages" between the same pair of nodes.)

Realizing Communication Patterns. A *realization* of the n-node communication pattern P is an n-node ring structure $\mathcal{R}(P)$, together with two one-to-one maps: there is a one-to-one association μ of regular nodes of $\mathcal{R}(P)$ with endpoints of chords of P; there is a one-to-one association ϵ of chords of P with i-hyperedges of $\mathcal{R}(P)$. These two maps are chosen so that, for each chord χ of P, the endpoints of the i-hyperedge $\epsilon(\chi) \in M(\mathcal{R}(P))$ are associated (via μ) with the endpoints of χ.

The Problem. Since we use a ring structure $\mathcal{R}(P)$ to realize a communication pattern P, we can talk unambiguously about the *width of a realization of communication pattern P*, namely the width $\mathcal{W}(\mathcal{R}(P))$ of the ring structure $\mathcal{R}(P)$. We can build on this to talk about the *width of a communication pattern P*, denoted $\mathcal{W}(P)$, namely, the smallest width of any realization of pattern P:

$$\mathcal{W}(P) = \min\{\mathcal{W}(\mathcal{R}(P)) \ : \ \mathcal{R}(P) \text{ is a realization of } P\}.$$

Our goal is an efficient algorithm that finds, for any given communication pattern P, a realization $\mathcal{R}(P)$ whose width is (close to) $\mathcal{W}(P)$. We achieve this goal, to within an error of $+1$, in the following sense. For the duration of the paper, say that the communication pattern P comprises c chords with n distinct endpoints and that c' of the chords are distinct.

Theorem 1 *There is a polynomial-time algorithm that produces, for any given communication pattern P, a realization $\mathcal{R}(P)$ such that $\mathcal{W}(\mathcal{R}(P)) \leq \mathcal{W}(P) + 1$. In particular, the algorithm operates in time $O(n^2c)$ and uses $O(n + c')$ words of storage.*

The remainder of this note is devoted to proving the Theorem, by presenting such an algorithm.

Remark. If we attempt to produce, for an arbitrary given communication pattern P, a realization $\mathcal{R}(P)$ with minimal chromatic number — defined by coloring i-hyperedges so that intersecting ones get distinct colors — then we encounter a problem that is NP-complete. This coloring problem corresponds to eliminating our assumption that message packets can change tracks (or frequencies) in transit. The classical ARC COLORING problem [4] is reducible to this coloring problem.

2 On Realizing Communication Patterns

The algorithm that is the main contribution of this note is a relaxation algorithm that starts with any "almost arbitrary" realization $\mathcal{R}_0(P)$ of an input communication pattern P and produces a sequence $\mathcal{R}_0(P), \mathcal{R}_1(P), \mathcal{R}_2(P), \ldots$ of successively "better" realizations of P, ending with a realization $\mathcal{R}^*(P)$ whose width is no greater than $\mathcal{W}(P) + 1$. By "almost arbitrary," we mean that $\mathcal{R}_0(P)$ can be any realization of P whose width at *some* x-point is 0; i.e., $\mathcal{R}_0(P)$ contains an x-node that is not a member of any i-hyperedge. We indicate later why requiring such a starting point makes our algorithm more efficient. Our notion of a "better realization" depends on the successive reduction of (at least one of) two measures of the cost of a realization $\mathcal{R}(P)$: the width $\mathcal{W}(\mathcal{R}(P))$ of $\mathcal{R}(P)$, as defined earlier, and the *width-density* $\mathcal{WD}(\mathcal{R}(P))$ of $\mathcal{R}(P)$, which is the number of x-nodes x at which $\mathcal{R}(P)$ has maximal width, i.e., such that $\mathcal{W}(\mathcal{R}(P), x) = \mathcal{W}(\mathcal{R}(P))$. That is, for each pair of successive realizations of P produced by our algorithm — say, $\mathcal{R}_i(P)$ and $\mathcal{R}_{i+1}(P)$ — either $\mathcal{W}(\mathcal{R}_{i+1}(P)) < \mathcal{W}(\mathcal{R}_i(P))$, or both $\mathcal{W}(\mathcal{R}_{i+1}(P)) = \mathcal{W}(\mathcal{R}_i(P))$ and $\mathcal{WD}(\mathcal{R}_{i+1}(P)) < \mathcal{WD}(\mathcal{R}_i(P))$.

2.1 Preliminaries

In this Section, we simplify our algorithm by verifying that only one type of relaxation device needs be considered when seeking to improve the quality of a given realization according to the width and width-density cost measures.

Simple vs. nonsimple i-hyperedges. The first — and easiest — observation about our problem is that replacing any nonsimple (i.e., self-overlapping) i-hyperedge of a ring structure $\mathcal{R}$ by the simple i-hyperedge having the same endpoints always decreases the width of $\mathcal{R}$. Fig. 1 verifies the following obvious lemma which justifies our restricting attention henceforth to realizations that use only simple i-hyperedges.

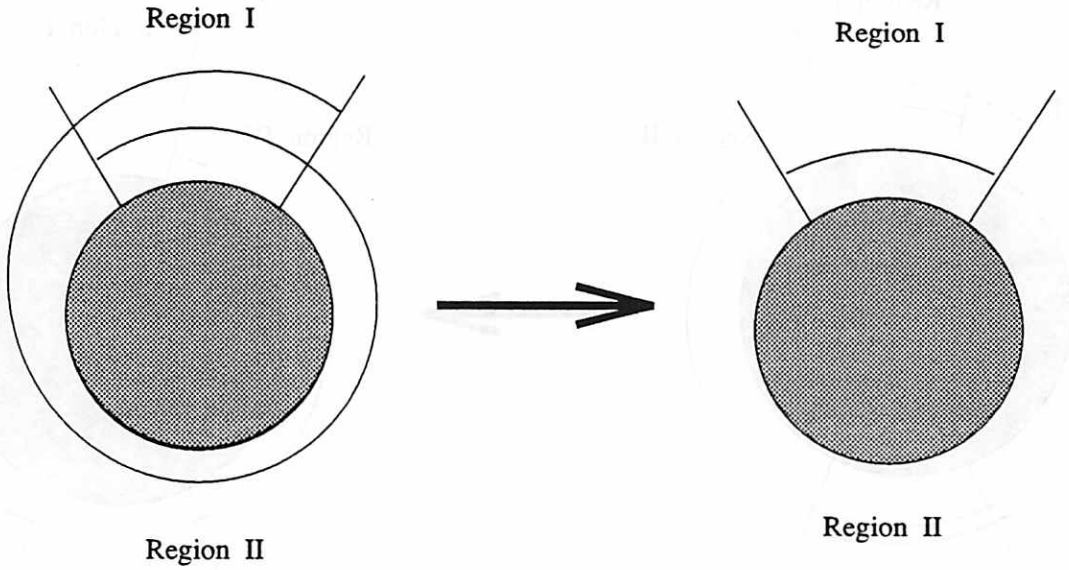

Figure 1: *Simple arcs always decrease width.*

Lemma 1 *The minimal-width realization of any communication pattern contains only simple i-hyperedges.*

We now define the basic operation that our algorithm uses to improve realizations, the act of *flipping* i-hyperedges. One *flips* the i-hyperedge

$$\langle r_i, r_j \rangle = \{r_i, x_i, r_{i+1}, x_{i+1}, \ldots, x_{j-1}, r_j\}.$$

that proceeds clockwise from regular node r_i to regular node r_j in a ring structure by replacing that i-hyperedge by the *complementary* one

$$\langle r_j, r_i \rangle = \{r_j, x_j, r_{j+1}, x_{j+1}, \ldots, x_{i-1}, r_i\}.$$

that proceeds clockwise from r_j to r_i. The successive realizations our algorithm produces result from flipping *pairs of* i-hyperedges.

Type-A flips: Doubly overlapping pairs of i-hyperedges. While one must generally be careful when choosing a pair of i-hyperedges to flip, in order to ensure that the flip does not increase the cost of the current realization (by increasing either its width or its width-density), there is one class of pair-flips that is guaranteed never to be harmful. A pair of i-hyperedges in a ring structure $\mathcal{R}$, call them $\langle r_{i_1}, r_{j_1} \rangle$, $\langle r_{i_2}, r_{j_2} \rangle$, have *double overlap* if:

- the i-hyperedges do not share an endpoint;

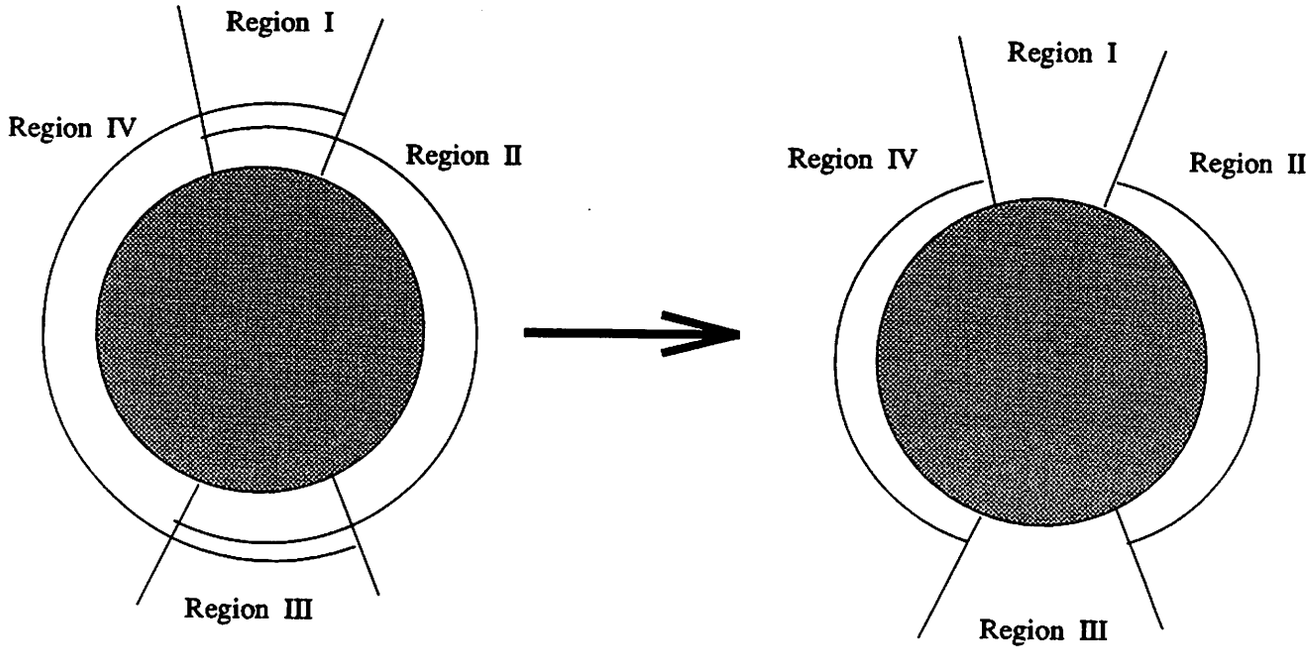

Figure 2: *Type-A flips never increase width.*

- there is a simple clockwise path in $\mathcal{R}$ that encounters the nodes $r_{i_1}, r_{j_2}, r_{i_2}, r_{j_1}$ in that order;

see Fig. 2. One easily notes in the figure that flipping a pair of i-hyperedges that have double overlap — which action we term a *Type-A flip* — reduces the width of $\mathcal{R}$ in Regions I and III while leaving the width unchanged in Regions II and IV; hence, a Type-A flip can never increase the cost of a realization. More importantly, a Type-A flip can often improve the quality of a realization. To wit, if the ring structure $\mathcal{R}$ contains a critical x-node that lies in either Region I or Region III of a doubly overlapping pair of i-hyperedges, then the ring structure $\mathcal{R}'$ obtained by Type-A flipping this pair of i-hyperedges has either smaller width or smaller width-density than $\mathcal{R}$. We thus have the following.

Lemma 2 *Every communication pattern has a minimal-width realization that has no doubly overlapping i-hyperedges.*

Lemma 2 asserts that one never increases the cost of a realization by eliminating a pair doubly overlapping i-hyperedges via a Type-A flip; however, such flips do increase the cost (i.e., the time) of our algorithm; moreover, some of this extra time is wasted, because some Type-A flips do not affect any critical x-node, hence do not decrease the

cost of a realization. Fortunately, we are able to avoid doubly overlapping i-hyperedges altogether, *at no algorithmic cost*, as follows. First, we ensure that our initial realization does not have any doubly overlapping i-hyperedges, hence requires no Type-A flips. Then, we proceed from realization to realization using only a genre of pair-flip that cannot introduce any doubly overlapping i-hyperedges. The first of these goals is achieved simply by insisting that some x-point in the initial realization be *exposed*, in the sense of not belonging to any i-hyperedge. The second of these goals is achieved automatically, by the nature of the Type-B flips that will be our tool for improving realizations. (This assertion is proved in Lemma 6.) These Type-B flips are described next.

Type-B flips: Singly overlapping pairs of i-hyperedges. A pair of i-hyperedges in a ring structure $\mathcal{R}$, call them $\langle r_{i_1}, r_{j_1} \rangle$ and $\langle r_{i_2}, r_{j_2} \rangle$, have *single overlap* if

- $r_{i_1} \neq r_{j_2}$;
- there is a simple clockwise path in $\mathcal{R}$ that encounters the nodes $r_{i_1}, r_{j_2}, r_{j_1}, r_{i_2}$ in that order;

see Fig. 3. One easily notes in the figure that flipping a pair of i-hyperedges that have single overlap — which action we term a *Type-B flip* — reduces the width of $\mathcal{R}$ in Region I, increases the width in Region III, and leaves the width unchanged in Regions II and IV. What this means is that one must exercise care in deciding which singly overlapping pairs of i-hyperedges to flip and which to leave alone. This determination is a major component of our algorithm.

Type-C flips: Degenerate Type-B flips. Our algorithm chooses pairs of i-hyperedges for Type-B flipping based on the clockwise and counterclockwise extreme nodes of the set of i-hyperedges that contain a selected x-node. On occasion, the selected pair will be degenerate, in the sense that a single i-hyperedge will have both the extreme clockwise node and the extreme counterclockwise node. (The import of the previous sentences will be clearer after the description of our algorithm.) In this case, we cannot perform a Type-B flip, which requires two i-hyperedges. Instead, we perform a *Type-C flip*, which consists in complementing the chosen i-hyperedge; see Fig. 4. We separate this case explicitly because it requires a few special words as we validate and analyze our algorithm in Section 2.3.

Orthogonal x-nodes and orthogonal regions. Two x-nodes of a ring structure $\mathcal{R}$ are *orthogonal* if no i-hyperedge of $\mathcal{R}$ contains both of them. Easily, an x-node x has at least one orthogonal partner if, and only if, x belongs to no doubly overlapping pair of i-hyperedges. Moreover, the set $\mathcal{O}R(x)$ (which we call the *orthogonal region of* x) of all x-nodes that are orthogonal to x-node x form an "interval," in the sense that $\mathcal{O}R(x)$ is

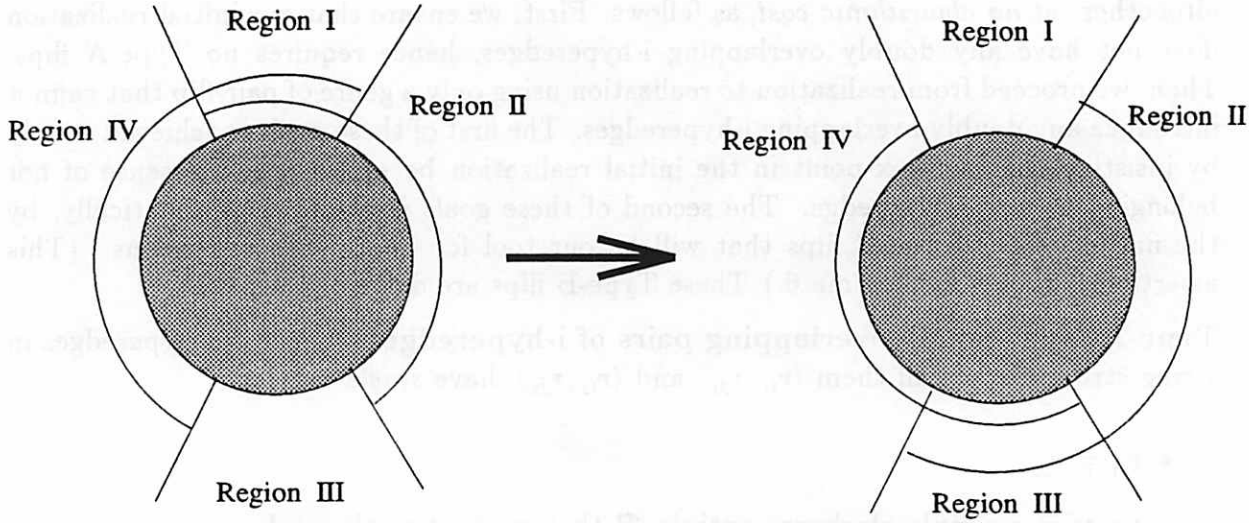

Figure 3: *Type-B flips increase width in places and decrease in other places, hence must be used carefully.*

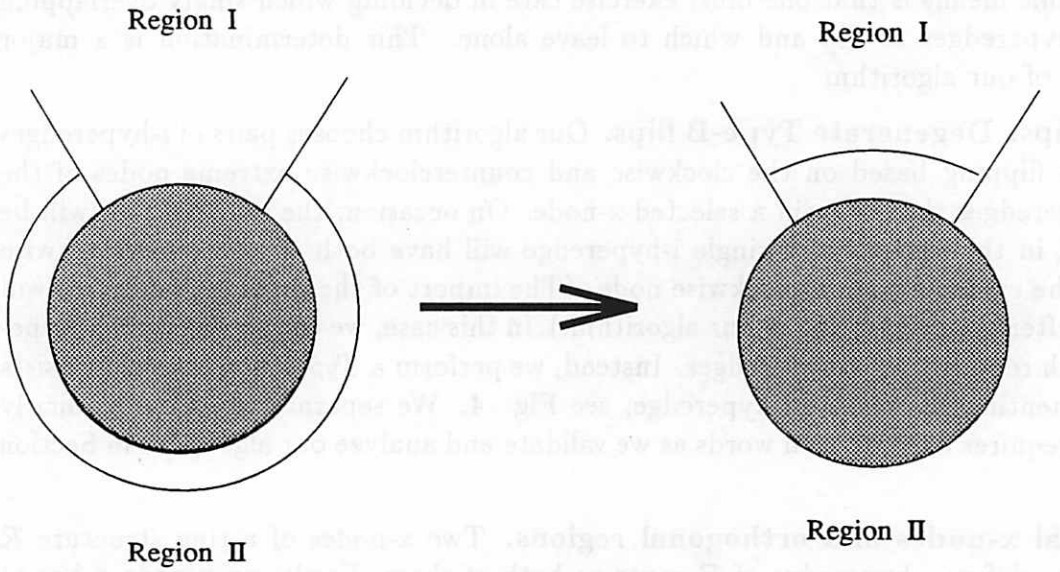

Figure 4: *Type-C flips increase width in places and decrease in other places, hence must be used carefully.*

either empty or has the form

$$\mathcal{O}R(x) = \{x_i, x_{i+1}, x_{i+2}, \ldots, x_j\} \tag{1}$$

for some x-nodes x_i and x_j. We introduce the following terminology (letting x_i and x_j be as in (1)):

- any i-hyperedge $\langle r, r_i \rangle$ that contains x is a *left-extremal i-hyperedge for* x;
- any i-hyperedge $\langle r_{j+1}, r \rangle$ that contains x is a *right-extremal i-hyperedge for* x.

The technical utility of the notion of orthogonality resides in the following property.

Lemma 3 *Let P be a communication pattern and $\mathcal{R}(P)$ a realization of P. For any x-node x of $\mathcal{R}(P)$ and any $y \in \mathcal{O}R(x)$,*

$$\mathcal{W}(\mathcal{R}(P), x) + \mathcal{W}(\mathcal{R}(P), y) \leq 2\mathcal{W}(P).$$

Proof. Add the x-nodes of $\mathcal{R}(P)$ explicitly to the circle used to define the communication pattern P. Say that k chords of pattern P cross the chord (x, y). Then, easily, $\mathcal{W}(P) \geq \lceil k/2 \rceil$, for each of these k chords must map onto some i-hyperedge of any realization of P. Since x and y are orthogonal x-nodes, no i-hyperedge that realizes one of these k chords contains both x and y. Hence,

$$\mathcal{W}(\mathcal{R}(P), x) + \mathcal{W}(\mathcal{R}(P), y) = k \leq 2\lceil k/2 \rceil \leq 2\mathcal{W}(P). \quad \square$$

2.2 The Realization Algorithm

A. An Overview of the Algorithm

Given: a communication pattern P
Find: A realization $\mathcal{R}(P)$ with $\mathcal{W}(\mathcal{R}(P)) \leq \mathcal{W}(P) + 1$

Step 0. Construct an initial realization $\mathcal{R}_0(P)$ that has at least one exposed x-point. Set the iteration counter $k = 0$.

Step 1. Select a critical x-node x of $\mathcal{R}_k(P)$.
Select a left-extremal i-hyperedge λ for x and a right-extremal i-hyperedge ρ for x.

Step 2. If $\lambda = \rho$, **then** update $\mathcal{R}_k(P)$ to $\mathcal{R}_{k+1}(P)$ by performing a Type-C flip of $\lambda = \rho$; **if** $\lambda \neq \rho$, **then** update $\mathcal{R}_k(P)$ to $\mathcal{R}_{k+1}(P)$ by performing a Type-B flip of λ and ρ.

Step 3. **If either** $\mathcal{W}(\mathcal{R}_{k+1}(P)) < \mathcal{W}(\mathcal{R}_k(P))$ **or both** $[\mathcal{W}(\mathcal{R}_{k+1}(P)) = \mathcal{W}(\mathcal{R}_k(P))$ **and** $\mathcal{WD}(\mathcal{R}_{k+1}(P)) < \mathcal{WD}(\mathcal{R}_k(P))]$
then increment k to $k+1$ and return to Step 1 with the new value;
else return $\mathcal{R}_k(P)$ as the desired realization of P and **halt**. □

B. The Enabling Data Structures

The data structures. Two basic data structures enable our algorithm, the first quite simple, the second a bit more complex.

1. There is a three-field register containing the integer $\mathcal{W}(\mathcal{R}_k(P))$ (the [global] width of the current realization), the integer $\mathcal{WD}(\mathcal{R}_k(P))$ (the width-density of the current realization), and a pointer to a critical x-node (any critical x-node will do).

2. There is an n-entry *linear array*. The ith entry of the array, where $1 \leq i \leq n$, plays the dual role of representing the ith regular node r_i of the current realization $\mathcal{R}_k(P)$ of problem P and its succeeding x-node x_i. For each i, the ith entry of the array comprises:

- a register containing the integer $\mathcal{W}(\mathcal{R}_k(P), x_i)$ (the [local] width of the current realization at x-node x_i);
- a *pair of linear lists*, named CW(i) (for clock*w*ise) and CCW(i) (for counterclock*w*ise), respectively. List CW(i) [resp., list CCW(i)] will contain, *in decreasing order of size*: (the name of) each i-hyperedge of the form $\langle r, r_i \rangle$ [resp., (the name of) each i-hyperedge of the form $\langle r_i, r \rangle$], together with the multiplicity of the i-hyperedge. The purpose of these lists is to allow us to extract extremal i-hyperedges and delimit orthogonal regions easily. Specifically, list CW(i) [resp., list CCW(i)] will allow us to extract a left-extremal i-hyperedge [resp., a right-extremal i-hyperedge] for a selected critical node in time $O(n)$.

Easily, our data structures occupy $O(n + c')$ words of storage.

The utility of the data structures. The data structures we have defined are quite well suited for implementing our algorithm. In particular, Type-B and Type-C flips, which are our tools for progressively improving realizations, affect the nodes of a realization within regions which are either contiguous blocks in our array or the union of two contiguous blocks, one at the beginning of the array and one at the end. For Type-B flips, see Fig. 3: in Region I (which is the orthogonal region of the current critical x-node), these flips add two i-hyperedges "above" each x-node; in Region III, they delete two i-hyperedges

"above" each x-node; and, in Regions II and IV, they add one i-hyperedge and delete one i-hyperedge "above" each x-node. At least three of these regions are contiguous blocks in the array, while the fourth is clustered either as one contiguous block or as two of them. For Type-C flips, see Fig. 4: in Region I (which is again the orthogonal region of the current critical x-node), these flips add one i-hyperedge "above" each x-node; in Region II, they delete one i-hyperedge "above" each x-node. At least one of these regions is a contiguous block in the array, while the other is clustered either as one contiguous block or as two of them. Each of the major operations in our algorithm will be implemented via a scan of the array, followed by some simple manipulation of the associated CW and CCW lists.

C. Details of the Algorithm

Creating realization $\mathcal{R}_0(P)$. Say that we are given the communication problem P as a set of pairs of integers, each pair indicating the endpoints of one of the chords we are to realize. The range of the integers indicates the number of nodes the realizing ring structures must have, which allows us to declare the length of the linear array. Say that, by default, we let x-node x_0 be the x-node that is left "exposed" in realization $\mathcal{R}_0(P)$. Then we initialize the CW and CCW lists by doing a pass over the array for each input chord χ. We represent chord $\chi = (i, j)$ by whichever of the i-hyperedges $\langle r_i, r_j \rangle$ and $\langle r_j, r_i \rangle$ does not contain x-node x_0. Say that $\langle r_i, r_j \rangle$ is the chosen i-hyperedge. Then we insert (the name of) this i-hyperedge into CW(j) (the CW list of node r_j) and into CCW(j) (the CCW list of node r_i). If i-hyperedge $\langle r_i, r_j \rangle$ does not yet exist in these lists, it is inserted behind (the names of) all bigger i-hyperedges and before (the names of) all smaller ones; if it already exists, then its multiplicity is increased. As we create these lists, we simultaneously accumulate the local widths $\{\mathcal{W}(\mathcal{R}_0(P), x_i)\}$ and the global width $\mathcal{W}(\mathcal{R}_0(P))$ and width-density $\mathcal{WD}(\mathcal{R}_0(P))$.

Selecting a critical x-node. By hypothesis, the global register always contains both the current global width $\mathcal{W}(\mathcal{R}_k(P))$ and a pointer to an x-node that is critical for $\mathcal{R}_k(P)$. We use this pointer to select the current critical x-node, since we are satisfied with any such x-node.

Selecting extremal i-hyperedges. Say that x_i is the selected critical x-node. In order to find a left extremal i-hyperedge for x_i, we scan the node array in the following order: we scan entry $i - 1$, then entry $i - 2$, then entry $i - 3$, and so on, until we find an i-hyperedge that contains node x_i as the first entry on the current CW list; that i-hyperedge is the selected left extremal i-hyperedge for x_i. Note that, since x-node x_i is critical for $\mathcal{R}_k(P)$, we are guaranteed to find a left extremal i-hyperedge by the time we reach entry $i + 1$. We then find a right extremal i-hyperedge for x_i by an analogous scan of the array, scanning in the order, entry $i + 1$, followed by entry $i + 2$, and so on, and using the CCW lists rather than the CW lists. If the selected extremal i-hyperedges are

distinct, we perform a Type-B flip; if they are identical, we perform a Type-C flip.

Performing Type-B and Type-C flip. As we noted earlier, performing a Type-B or a Type-C flip amounts to adding and/or deleting i-hyperedges from the sets of i-hyperedges that cover the x-nodes in the (respectively, four or two) regions defined by the chosen extremal i-hyperedge(s). We effect these additions and deletions by performing the appropriate insertions into and deletions from the lists associated with the endpoints of the flipped i-hyperedge(s). For instance, to flip i-hyperedge $\langle r_i, r_j \rangle$, we:

- decrease the multiplicity of $\langle r_i, r_j \rangle$ in CW(j) and CCW(i); if this decreases the multiplicity of the i-hyperedge to 0, then we remove it from the lists;
- insert the i-hyperedge $\langle r_j, r_i \rangle$ in CW(i) and CCW(j), in the way described during the construction of $\mathcal{R}_0(P)$.

In addition to performing these additions and deletions, we scan the entire array in order to:

- tally the (possibly new) maximum local width at each x-node;
- keep track of a (possibly new) critical x-node;
- keep track of the current (possibly new) maximum global width and the "density" of this width.

At the end of the scan, we use this information to update the global register.

2.3 Validation and Analysis

This section has two goals. First, we verify that our algorithm actually produces an efficient realization of the given communication pattern P, i.e., a realization $\mathcal{R}(P)$ satisfying $\mathcal{W}(\mathcal{R}(P)) \leq \mathcal{W}(P) + 1$, as claimed in Theorem 1. Second, we verify that our algorithm operates in time polynomial in the size of the pattern P, as claimed in Theorem 1.

A. Algorithm Validation

Our algorithm iterates the following pair of operations, until an execution of the pair fails to improve the current realization of the given communication pattern P.

1. Select a critical x-node of the current realization.

2. Perform a Type-B or Type-C flip (as appropriate) on (an) extremal i-hyperedge(s) that contain(s) the selected critical x-node.

It then announces that the current realization (before the last flip) is the sought realization $\mathcal{R}^*(P)$ whose width is within $+1$ of optimal. In order to validate the operation, we must prove that the algorithm terminates (so that realization $\mathcal{R}^*(P)$ exists) and that realization $\mathcal{R}^*(P)$ satisfies Theorem 1. We begin with the former task.

Lemma 4 *Our algorithm performs no more than $O(nc)$ Type-B or Type-C flips before halting.*

Proof. Our algorithm "accepts" a Type-B or Type-C flip that transforms the current realization $\mathcal{R}_k(P)$ to its successor $\mathcal{R}_{k+1}(P)$ precisely when the flip either reduces $\mathcal{W}(\mathcal{R}_k(P))$ or keeps it fixed while reducing $\mathcal{WD}(\mathcal{R}_k(P))$. Since, for any realization $\mathcal{R}(P)$, $\mathcal{W}(\mathcal{R}(P)) \leq c$ and $\mathcal{WD}(\mathcal{R}(P)) \leq n$, our algorithm "accepts" no more than $O(nc)$ flips before encountering its halting criterion. □

Finally, we verify that $\mathcal{R}^*(P)$ has small width.

Lemma 5 *If the last flip performed by our algorithm — the flip that was not "accepted" — was a Type-B flip, then $\mathcal{W}(\mathcal{R}^*(P)) \leq \mathcal{W}(P) + 1$. If the last flip was a Type-C flip, then $\mathcal{W}(\mathcal{R}^*(P)) = \mathcal{W}(P)$.*

Proof. Since our algorithm performs only Type-B and Type-C flips (and never Type-A flips), we simplify our argument by verifying that the algorithm never produces any doubly overlapping pairs of i-hyperedges in the course of "improving" successive realizations. (Otherwise, we would have to argue that it produces only "benign" doubly overlapping pairs, i.e., ones that do not increase the global width.) This is the task of the next lemma.

Lemma 6 *If the ring structure $\mathcal{R}$ contains no doubly overlapping pair of i-hyperedges, then the same is true for any ring structure $\mathcal{R}'$ obtained from $\mathcal{R}$ by*

- *Type-B flipping a singly overlapping left-extremal i-hyperedge and right-extremal i-hyperedge for some x-node.*
- *Type-C flipping an i-hyperedge that is both left-extremal and right-extremal for some x-node.*

Proof. We present a proof only for Type-B flips, leaving to the reader the similar argument about Type-C flips.

Focus on a ring structure $\mathcal{R}$ that contains no doubly overlapping pair of i-hyperedges. Let x be an x-node of $\mathcal{R}$, let λ be a left-extremal i-hyperedge that contains x, and let ρ be a right-extremal i-hyperedge that contains x. Say that we Type-B flip the i-hyperedges λ and ρ, and, after the flip, the resulting ring structure $\mathcal{R}'$ contains a doubly overlapping pair of i-hyperedges, call them $\alpha = \langle r_{i_1}, r_{j_1} \rangle$ and $\beta = \langle r_{i_2}, r_{j_2} \rangle$. Clearly, one of this pair is the complement of either λ or ρ, or else the double overlap would have existed in ring structure $\mathcal{R}$. Say, for definiteness, that α is the complement of λ. Now, since α and β have double overlap, we know that they have distinct endpoints and that there is a simple clockwise path in $\mathcal{R}'$ that encounters the nodes $r_{i_1}, r_{j_2}, r_{i_2}, r_{j_1}$ in that order. Since r_{j_1} follows r_{i_2} in this path, i-hyperedge β must contain x-node x, because λ does; moreover, since r_{j_2} follows r_{i_1} in this path, we see that i-hyperedge β's clockwise extreme endpoint goes further clockwise than does λ's (since their endpoints are distinct). This, however, contradicts the left-extremism of i-hyperedge λ. We conclude that i-hyperedge β cannot exist. □

Return to the proof of Lemma 5. We know now that we need consider only Type-B and Type-C flips. Let us focus, therefore, on the last flip performed by our algorithm.

Assume first that the last performed flip was a Type-B flip. Say that this flip was performed on realization $\mathcal{R}_k(P)$, so that $\mathcal{R}_k(P)$ was subsequently announced by the algorithm to be the desired realization $\mathcal{R}^*(P)$; say further that the critical x-node chosen for this final flip was x. Recall that a Type-B flip alters the local widths of $\mathcal{R}_k(P)$ only at x-nodes in regions I and III (as depicted in Fig. 3): the flip decreases each x-node's local width by 2 in Region III, while increasing each x-node's local width by 2 in Region I.

Say, for the sake of argument, that each x-node $y \in \mathcal{O}R(x)$ has $\mathcal{W}(\mathcal{R}_k(P), y) < \mathcal{W}(\mathcal{R}_k(P), x) - 2$. Easily then, after the flip, each x-node in Regions I and III will have local width $< \mathcal{W}(\mathcal{R}_k(P))$. Since the flip does not alter the local width of any x-node in Regions II or IV, it follows that either $\mathcal{W}(\mathcal{R}_{k+1}(P)) < \mathcal{W}(\mathcal{R}_k(P))$, or both $\mathcal{W}(\mathcal{R}_{k+1}(P)) = \mathcal{W}(\mathcal{R}_k(P))$ and $\mathcal{WD}(\mathcal{R}_{k+1}(P)) < \mathcal{WD}(\mathcal{R}_k(P))$. (The former disjunct occurs when $\mathcal{R}_k(P)$'s only critical x-nodes were in Region III; the latter occurs (without the former) when $\mathcal{R}_k(P)$ had critical x-nodes in Region II or Region IV as well as in Region III.) In other words, if each x-node $y \in \mathcal{O}R(x)$ had had "small" local width, as defined above, then the realization $\mathcal{R}_{k+1}(P)$ would have improved $\mathcal{R}_k(P)$, in the sense of the disjunctive cost reduction. Since the improvement did not actually occur, we conclude that at least one x-node $y \in \mathcal{O}R(x)$ did *not* have "small" local width.

Focus now on any x-node $y \in \mathcal{O}R(x)$ for which $\mathcal{W}(\mathcal{R}_k(P), y) \geq \mathcal{W}(\mathcal{R}_k(P), x) - 2$.

Since x-node x is critical,

$$\mathcal{W}(\mathcal{R}_k(P), x) = \mathcal{W}(\mathcal{R}_k(P)).$$

Since x-nodes x and y are orthogonal, by Lemma 3,

$$\mathcal{W}(\mathcal{R}_k(P), x) + \mathcal{W}(\mathcal{R}_k(P), y) \leq 2\mathcal{W}(P).$$

Finally, by hypothesis,

$$\mathcal{W}(\mathcal{R}_k(P), y) \geq \mathcal{W}(\mathcal{R}_k(P), x) - 2.$$

These three relations combine as follows to verify the Lemma when the "rejected" flip was of Type B.

$$\begin{aligned} 2\mathcal{W}(\mathcal{R}^*(P)) - 2 &= 2\mathcal{W}(\mathcal{R}_k(P), x) - 2 \\ &\leq \mathcal{W}(\mathcal{R}_k(P), x) + \mathcal{W}(\mathcal{R}_k(P), y) \\ &\leq 2\mathcal{W}(P). \end{aligned}$$

When the "rejected" flip was of Type C, we can perform an analysis similar to the preceding one. The major differences are as follows. The single extremal hyperedge now partitions the ring into two regions; cf. Fig. 4. The flip increases the local widths of x-nodes in Region I by 1, while it decreases the local widths of x-nodes in Region II by 1. If the Type-C flip with critical x-node flip is "rejected," then, it is because there is an x-node $y \in \mathcal{O}R(x)$ for which $\mathcal{W}(\mathcal{R}_k(P), y) \geq \mathcal{W}(\mathcal{R}_k(P), x) - 1$. This alters the analysis from the Type-B case by changing its final chain of relations to:

$$\begin{aligned} 2\mathcal{W}(\mathcal{R}^*(P)) - 1 &= 2\mathcal{W}(\mathcal{R}_k(P), x) - 1 \\ &\leq \mathcal{W}(\mathcal{R}_k(P), x) + \mathcal{W}(\mathcal{R}_k(P), y) \\ &\leq 2\mathcal{W}(P). \end{aligned}$$

Since $\mathcal{W}(\mathcal{R}^*(P))$ is an integer, this implies that $\mathcal{W}(\mathcal{R}^*(P)) = \mathcal{W}(P)$. □

B. Timing Analysis

We consider the constituent tasks of our algorithms in turn and estimate the worst-case time for each. Say that the input communication pattern P consists of c chords based on n endpoints.

Creating realization $\mathcal{R}_0(P)$. While creating the n-node ring structure that is realization $\mathcal{R}_0(P)$, we are setting up our algorithm's system of data structures. We process each chord χ of pattern P as follows; let x_0 be the x-node that is left exposed in realization $\mathcal{R}_0(P)$, and say that chord χ has endpoints r_i and r_j.

1. Select either i-hyperedge $\langle r_i, r_j \rangle$ or $\langle r_j, r_i \rangle$ to represent chord χ, whichever does not contain x-node x_0; say, with no loss of generality, that i-hyperedge $\langle r_i, r_j \rangle$ is selected.

2. Insert $\langle r_i, r_j \rangle$ into the lists CW(j) and CCW(i), after all larger i-hyperedges and before all smaller ones. (If $\langle r_i, r_j \rangle$ was already present in a list, just increase its multiplicity.)

3. Increment the register containing $\mathcal{W}(\mathcal{R}_0(P), x)$ for each x-node belonging to i-hyperedge $\langle r_i, r_j \rangle$, and update the register containing $\mathcal{W}(\mathcal{R}_0(P))$ and $\mathcal{WD}(\mathcal{R}_0(P))$ appropriately.

For each of pattern P's c chords: The computation that implements Step 1 takes time $O(1)$. The computation that implements Step 2 takes time $O(1)$ to access the desired lists within the array and time $O(n)$ to scan the lists in order to perform the insertion. Step 3 takes time $O(n)$ to scan the array updating local widths and time $O(1)$ to update the global register following each local update. Hence, within our framework, one can create realization $\mathcal{R}_0(P)$ in time $O(nc)$.

Selecting a critical x-node. Since this selection involves just accessing a register, it takes time $O(1)$.

Selecting extremal i-hyperedges. Selecting each extremal i-hyperedge involves scanning some portion of the array and testing the first element of either the CW or CCW list at each scanned array entry (to determine if the tested hyperedge contains the selected critical x-node). The scan takes time $O(n)$; the testing takes time $O(1)$.

Performing a Type-B or Type-C flip. Since performing a Type-B flip takes precisely twice the number of steps as performing a Type-C flip, we analyze only the former operation. Once having selected extremal i-hyperedges, performing a Type-B flip entails:

1. removing the extremal i-hyperedges from the lists they resided in,

2. inserting the complementary i-hyperedges into the "complementary" lists,

3. scanning the array to adjust the local widths of each x-node in accord with the added and deleted i-hyperedges,

4. updating the global width and width density in accord with the changes in the local widths.

Step 1 takes time $O(1)$. Steps 2 and 3 take time $O(n)$ each. Step 4 takes time $O(1)$ for each execution of step 3. Hence, performing a Type-B flip takes time $O(n)$; performing a Type-C flip takes roughly half the time of a Type-B flip.

Testing for improved cost. The new global width and width-density are computed while updating the array during the Type-B or Type-C flip. Thus, the algorithm needs only compare the old contents of the global width register and width-density register with the new contents in order to determine if the latest flip yields an improved realization. This determination thus takes time $O(1)$.

The grand total. Since one constructs the initial realization $\mathcal{R}_0(P)$ just once, and since one iterates the select-flip-decide sequence only $O(nc)$ times (cf. Lemma 4), the entire process of finding an almost-optimal realization of communication pattern P takes time $O(n^2c)$.

ACKNOWLEDGMENTS. We are grateful to Shimon Even for helpful conversations.

The research of A. L. Rosenberg was supported in part by NSF Grant CCR-92-21785.

After completing the research described here, we learned, via private communication, that Noga Alon and Paul Seymour have a nonconstructive proof of the polynomial-time solvability of some variant of the problem of optimally realizing a given communication pattern. We have not seen any details of their work, which we believe has not been written up.

References

[1] G. Bilardi and F.P. Preparata (1993): Horizons of parallel computation. *Symp. for the 25th Anniversary of INRIA. Lecture Notes in Computer Science 653*, Springer-Verlag, N.Y., 1992, 155-174.

[2] R. Cohen (1993): One-bit delay in ring networks. *IEEE Trans. Comp. 42*, 735-737.

[3] P. Fraigniaud, S. Miguet, Y. Robert (1990): Complexity of scattering on a ring of processors. *Parallel Computing 13*, 377-383.

[4] M.R. Garey, D.S. Johnson, G.L. Miller, C.H. Papadimitriou (1980): The complexity of coloring circular arcs and chords. *SIAM J. Algebr. Discr. Meth. 1*, 216-227.

[5] J. Hromkovič, V. Müller, O. Sýkora, I. Vrťo (1992): On embedding interconnection networks into rings of processors. *PARLE '92. Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

[6] S.L. Johnsson and C.-T. Ho (1989): Optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Comp. 38*, 1249-1268.

[7] S. Miguet and Y. Robert (1990): Path planning on a ring of processors. *Intl. J. Computer Math. 32*, 61-74.

[8] A.L. Rosenberg (1984): On designing fault-tolerant VSLI processor arrays. In *Advances in Computing Research 2* (F.P. Preparata, ed.) JAI Press, Greenwich, CT, 181-204.

[9] A.L. Rosenberg (1989): Interval hypergraphs. In *Graphs and Algorithms* (R.B. Richter, ed.) *Contemporary Mathematics 89*, Amer. Math. Soc., 27-44.

[10] Y. Saad and M.H. Schultz (1989): Data communication in parallel architectures. *Parallel Computing 11*, 131-150.

[11] Q.F. Stout and B. Wagar (1990): Intensive hypercube communication, I: prearranged communication in link-bound machines. *J. Parallel Distr. Comput. 10*, 167-181.