Lower Bounds on Communication Loads and Optimal Placements in Torus Networks*

M. Cemil Azizoğlu† and Ömer Eğecioğlu
Department of Computer Science
University of California at Santa Barbara
$\{azizoglu, omer\}@cs.ucsb.edu$

Abstract

Fully-populated torus-connected networks, where every node has a processor attached, do not scale well since load on edges increases superlinearly with network size under heavy communication, resulting in a degradation in network throughput. In a *partially-populated network*, processors occupy a subset of available nodes, and a routing algorithm is specified among the processors placed.

Analogous to multistage networks, it is desirable to have the total number of messages being routed through a particular edge increase at most linearly with the size of the placement on torus networks. Recently, Blaum, Bruck, Pifarré, and Sanz investigated placements and provided both a lower bound, and optimal placements in the cases of 2 and 3-dimensional k-tori, consisting of k and k^2 processors, respectively.

In this paper we show formally that to achieve linear load in a d-dimensional k-torus, the number of processors in the placement must be $O(k^{d-1})$. We use this to construct a tighter lower bound for the maximum load of a placement for 4 or more dimensions. Based on these results, we give optimal placements and their corresponding routing algorithms in tori with arbitrary number of dimensions.

Keywords: Torus, routing, placement, bisection, interconnection network, edge separator, congestion.

1 Introduction

Meshes and torus based interconnection networks have been utilized extensively in the design of parallel computers in recent years [5]. This is mainly due to the fact that these families of networks have topologies which reflect the communication pattern of a wide

*An extended abstract was presented in the IEEE Symposium IPPS/SPDP 1998, April 1998, Orlando.
†Supported in part by a fellowship from İzmir Institute of Technology, İzmir, Turkey.

variety of natural problems, and at the same time they are scalable, and highly suitable for hardware implementation. An important factor determining the efficiency of a parallel algorithm on a network is the efficiency of communication itself among processors. The network should be able to handle "large" number of messages without exhibiting degradation in performance. *Throughput*, the maximum amount of traffic which can be handled by the network, is an important measure of network performance [3]. The throughput of an interconnection network is in turn bounded by its *bisection width*, the minimum number of edges that must be removed in order to split the network into two parts each with about equal number of processors [8].

Here, following Blaum, Bruck, Pifarré, and Sanz [3, 4], we consider the behavior of torus networks with bidirectional links under heavy communication load. We assume that the communication latency is kept minimum by routing the messages through only *shortest (minimal length) paths*. In particular, we are interested in the scenario where every processor in the network is sending a message to every other processor (also known as *complete exchange* or *all-to-all personalized communication*). This type of communication pattern is central to numerous parallel algorithms such as matrix transposition, fast Fourier transform, distributed table-lookup, etc. [6], and central to efficient implementation of high-level computing models such as the PRAM and Bulk-Synchronous Parallel (BSP). In Valiant's BSP-model for parallel computation [14] for example, routing of h-relations, in which every processor in the network is the source and destination of at most h packets, forms the main communication primitive. Complete-exchange scenario that we investigate in this paper has been studied and shown to be useful for efficient routing of both random and arbitrary h-relations [7, 12, 13].

The network of d-dimensional k-torus is modeled as a directed graph where each node represents either a router or a processor-router pair, depending on whether or not a processor is attached at this node, and each edge represents a communication link between two adjacent nodes. Hence, every node in the network is capable of message routing, i.e. directly receiving from and sending to its neighboring nodes.

A fully-populated d-dimensional k-torus where each node has a processor attached, contains k^d processors. Its bisection width is $4k^{d-1}$ (k even), which gives $k^d/2$ processors on each component of the bisection. Under the complete-exchange scenario, the number of messages passing through the bisection in both directions is $2(k^d/2)(k^d/2)$. Dividing by the bisection bandwidth, we find that there must exist an edge in the bisection with a load $\geq k^{d+1}/8$. This means that unlike multistage networks, the maximum load on a link is not linear in the number of processors injecting messages into the network. To alleviate this problem, Blaum et al. [3, 4] have proposed *partially-populated tori*. In this model, the underlying network is torodial, but the nodes do not all inject messages into the network. We think of the processors as attached to a (relatively small) subset of nodes (called a *placement*), while the other nodes are left as routing nodes. This is similar to the case of a multistage network: A multistage network with $k \times k$ switches (routing nodes) and $\log_k n$ stages serves n injection points, and utilizes $n \log_k n$ routing nodes [3].

In partially-populated tori, a routing algorithm which utilizes shortest paths is specified together with the placement. An *optimal placement* is a placement that achieves linear load on edges using maximum number of processors possible.

The notion of *resource placement* in general has been investigated by a number of researchers such as Bose et al. [5], Alverson et al. [1], F. Pitteli and D. Smitley [11]. Our aim is to give placements and routing algorithms which will enable efficient communication between processors, and at the same time reduce the susceptibility of the network to link faults by reducing the number of messages relying upon a particular edge [3]. This is achieved by providing routing algorithms in which the number of minimal paths specified between pairs of processors in the placement is kept large, without compromising the linearity of load.

Let $\mathcal{E}_{max}$ denote the maximum load over all the edges for the placement P. Blaum et al. give the lower bound

$$\mathcal{E}_{max} \geq (|P| - 1)/(2d) \tag{1}$$

which means that for $d = 2$, $\mathcal{E}_{max} \geq |P|/4$ and for $d = 3$, $\mathcal{E}_{max} \geq |P|/6$. If $|P|$ is constrained to be of the form k^i, then they also give placements of sizes k for $d = 2$ and k^2 for $d = 3$, together with routing algorithms. These placements are *optimal* in the sense that the two lower bounds are actually achieved by the placements.

How do we justify that in general a maximal size placement that can achieve linear load is $O(k^{d-1})$? If the placement has size ck^{d-1} for some constant c, then mimicking the case of the fully-populated d-dimensional k-torus,

$$2(ck^{d-1}/2)(ck^{d-1}/2)/(4k^{d-1}) \;=\; O(k^{d-1}) \;.$$

This seems to imply that linear load is at least possible for $|P| = ck^{d-1}$. This is a faulty argument however, as we do not know a priori that number of edges needed to split P into two equal size pieces is the same as the bisection width of the whole torus. This may push the size of an optimal placement above or below k^{d-1}. In this paper, we introduce the concept of *bisection width with respect to a placement* P, and use its properties to prove that in a d-dimensional k-torus, the size of an optimal placement is $\Theta(k^{d-1})$. Given a placement P of maximal size, we also prove that there exists an edge separator of size $\Theta(k^{d-1})$ which splits the torus into two components with $\Theta(k^{d-1})$ processors of P on each side. This gives a lower bound of the form

$$\mathcal{E}_{max} \geq ck^{d-1} \tag{2}$$

for maximum load. In (2) c is a constant independent of d. This is a tighter lower bound for the load for large d than the lower bound (1).

Finally, we give optimal placements (called *linear placements*) achieving the lower bound (2) and corresponding routing algorithms (*Ordered Dimensional Routing (ODR)* and *Unordered Dimensional Routing (UDR)*) in tori with arbitrary number of dimensions. Of the two routing algorithms ODR is simpler, but UDR provides fault tolerance by allowing more routes. We also show how tho extend these to more general placements in tori that we refer to as *multiple linear placements.*

The outline of the paper is as follows. Section 2 gives necessary definitions and the formal statement of the problem. In section 3, a lower bound on the maximum load on an edge is given, which is also a generalization of the lower bound given by [3]. This bound, along with the notion of bisection width with respect to a placement, is used to get an upper bound on the number of processors in an optimal placement. We introduce the notion of $\alpha-$*separator with respect to a placement* in section 4, and use it to give a new lower bound on the maximum load which is independent of the dimension parameter in section 5. Finally, in sections 6-8, we define and analyze an important class of placements called linear placements, and give associated routing algorithms which achieve linear load and fault tolerance. Section 9 includes conclusions and some future considerations.

2 Preliminaries and Problem Definition

In this section, we start out with the problem definition, and follow it by a sequence of formal definitions and terminology that will be used in the rest of the paper.

Problem Definition

Our aim is to find *placements* and associated *routing algorithms* in the d-dimensional k-torus T_k^d that have *linear message load* (in number of processors in the placement) on edges under the *complete exchange* scenario. Specifically, we like to devise a placement P, and a routing algorithm A for P for which $\mathcal{E}_{max} \; = \; c|P|$, for some constant c.

Definition 1 {d-Dimensional k-Torus}
The d-dimensional k-torus is a directed graph $T_k^d = (V, E)$*, with vertex set*

$$V = \{\vec{a} \mid \vec{a} = (a_1, a_2, \ldots, a_d),\;\; a_i \in \mathbb{Z}_k\}$$

where $\mathbb{Z}_k$ *denotes the integers modulo k, and edge set*

$$E = \{(\vec{a}, \vec{b}) \mid \exists j \;\; such\ that\ a_j \equiv b_j \pm \;\; 1\ (mod\ k)\ and,\ a_i = b_i\ for\ i \neq j,\;\; 1 \leq i \leq d\}.$$

T_k^d has a total of k^d nodes. Each node has two neighbors in each dimension, for a total of $2d$ neighbors. Directed edges of T_k^d are also referred to as *links.*

Definition 2 {**Placement**}
A placement P *of processors in* $T_k^d = (V, E)$ *is a subset of* V.

We use the term *node* for a generic element of the vertex set of T_k^d. A node with a processor attached is simply called a *processor*.

Definition 3 {**Routing Algorithm**}
Let P *be a placement in* T_k^d. *A* routing algorithm A *is a subset* $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ *of the set of all shortest paths between* $\vec{p}$ *and* $\vec{q}$ *for every pair* $\vec{p}$, $\vec{q} \in P$ *(See Figure 1).*

The routing algorithm A is used to deliver packets from $\vec{p}$ to $\vec{q}$: When $\vec{p}$ needs to communicate with $\vec{q}$, a shortest path in $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ is selected randomly with uniform probability.

For any link l, we denote the set of paths in $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ going through l by $\mathcal{C}^A_{\vec{p}\to l\to\vec{q}}$, and use the following definition of load as given in [3].

Definition 4 {**Load**}
Given a placement P *in a* T_k^d *along with a routing algorithm* A, *the* load *of an edge* l *is defined as*

$$\mathcal{E}(l) = \sum_{\substack{\vec{p}, \vec{q} \in P \\ \vec{p} \neq \vec{q}}} \frac{|\mathcal{C}^A_{\vec{p}\to l\to\vec{q}}|}{|\mathcal{C}^A_{\vec{p}\to\vec{q}}|} \tag{3}$$

Definition 5 {**Maximum Load**}
The maximum value of $\mathcal{E}(l)$ *for a network with placement* P *and a routing algorithm* A *is called the* maximum load *and denoted by* $\mathcal{E}_{max}$. *Thus*

$$\mathcal{E}_{max} = \max_{l \in E} \mathcal{E}(l).$$

Considering the expression (3) for $\mathcal{E}(l)$, the more paths the routing algorithm provides between any two processors, the smaller the load on any edge that is used to route messages between these processors. In addition to this, availability of a large number of choices means better fault tolerance.

We shall consider algorithms which use minimal (shortest) paths. Minimal paths are associated with the notion of *cyclic distance* and *Lee distance* which we define next.

Definition 6 {**Cyclic Distance, Lee Distance**}
Given three integers, i, j *and* k, *the* cyclic distance *between* i *and* j *modulo* k *is given by*

$$min\{i - j \ (mod\ k),\ \ j - i \ (mod\ k)\}$$

where the equivalence classes modulo k *are taken to be* $0, 1, \ldots, k-1$. *The* Lee distance *between two nodes* $\vec{p}, \vec{q} \in T_k^d$ *is the sum of the cyclic distances between the coordinates of* $\vec{p}$ *and* $\vec{q}$.

The Lee distance between $\vec{p}, \vec{q} \in T_k^d$ is the length of a shortest path between $\vec{p}$ and $\vec{q}$ on the torus [5, 9].

Definition 7 {**Bisection Width**}
The bisection width *of a graph is the minimum number of edges which must be removed in order to split the node set into two parts of equal (within one) cardinality.*

Definition 8 {**Bisection Width with respect to a Placement**}
The bisection width with respect to a placement P *of* $T_k^d = (V, E)$ *is the minimum number of edges which must be removed from* E *in order to split* V *into two parts each of which containing an equal (within one) number of processors in* P.

We denote by $\partial_b P$ a minimal cardinality set of edges of T_k^d which needs to be removed to bisect P. Thus $|\partial_b P|$ is the bisection width with respect to the placement P.

Definition 9 {**α-separator Width with respect to a Placement**}
An α-separator with respect to a placement P *in* T_k^d *is a set of edges whose removal splits the graph into two parts containing (approximately)* $\alpha|P|$ *and* $(1-\alpha)|P|$ *processors, for* $0 < \alpha < 1$. *The* α-separator width of T_k^d with respect to a placement P *is the size of a minimal* α*-separator with respect to* P.

We denote the set of edges in an α-separator by $\partial_\alpha P$. Thus $|\partial_\alpha P|$ is the α-separator width of T_k^d with respect to P. Note that when $\alpha = 1/2$, $\partial_\alpha P$ and $\partial_b P$ are equivalent.

Definition 10 $\{O, \Omega, \Theta, o\}$ *We use the notation* $f(k) = O(k^{d-1})$ *to mean* $f(k) \leq ck^{d-1}$ *for some constant* $c > 0$ *whenever* $k \geq k_0$. $f(k) = \Omega(k^{d-1})$ *if for some constant* $c > 0$, $f(k) \geq ck^{d-1}$ *for infinitely many values of* k. $f(k) = \Theta(k^{d-1})$ *iff* $f(k) = O(k^{d-1})$ *and* $f(k) = \Omega(k^{d-1})$. *We loosely use* $f(k) = o(k^{d-1})$ *to mean* $\lim_{k \to \infty} f(k)/k^{d-1} = 0$.

The problem we are interested in is the construction of placements P and associated routing algorithms in T_k^d such that under the complete exchange scenario, $\mathcal{E}_{max} = c|P|$, for some constant c.

For $d = 2, 3$, Blaum et al. [3, 4] have investigated placements with k^{d-1} processors. Evidently, placements with provably maximum possible number of processors are desirable. This raises another important question which we shall address: *what is the maximum number of processors a placement could have on* T_k^d *without compromising linear load on edges?*

Another important issue is *fault tolerance*. Specifically, the routing algorithm should provide multiple routing paths between each pair of processors so that, if any of the links fails, the network will remain functional by routing the messages through paths which do not include the defective link. Consequently we also address the following problem: *is it possible to construct optimal placements which are at the same time fault tolerant?*

In the following sections, we analyze lower bounds for maximum load and study the above questions.

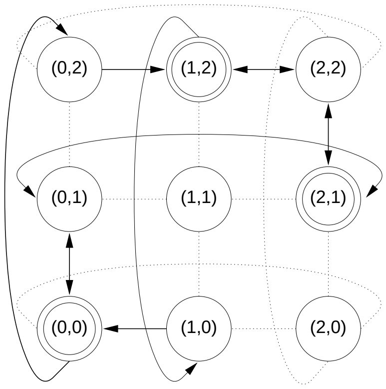

Figure 1: A placement of 3 processors on T_3^2. Among the links, the ones on specified shortest paths between the processors are highlighted.

3 A General Lower Bound for Maximum Load

We start out with an important lemma which will prove to be a very useful tool in the subsequent sections. The lower bound for maximum load originally given by Blaum et al. [3] is

$$\mathcal{E}_{max} \geq \frac{|P|-1}{2d} \tag{4}$$

The following lemma gives a more general form of (4).

Lemma 1 *Let P be a placement in a $T_k^d = (V, E)$, also, let $S \subset P$ and ∂S be the set of all edges each connecting a node in S with another node not in S. Then*

$$\mathcal{E}_{max} \geq \frac{2|S|(|P|-|S|)}{|\partial S|} \tag{5}$$

Proof The total number of messages exchanged between processors in S and processors in $P-S$, in either direction, is $2|S|(|P|-|S|)$ under all-to-all personalized communication scenario. Also, these messages must go through one of the edges in ∂S. The average number of messages going through an edge in ∂S is $2|S|(|P|-|S|)/|\partial S|$ and the lemma follows. $\square$

It is easy to see that (5) reduces to (4) if the set S is taken to contain only one processor, i.e., $|S| = 1$ and $|\partial S| = 4d$. The lower bound (5) is valid independent of the routing algorithm used. Another interesting form of (5) that we shall subsequently make use of is obtained when the set S consists of half of the processors in P, i.e.,

$$\mathcal{E}_{max} \geq \frac{2(\frac{|P|}{2})^2}{|\partial_b P|} \tag{6}$$

Note that in this case ∂S becomes $\partial_b P$, which is the bisection width of T_k^d with respect to placement P. Next we give an upper bound on the size of $\partial_b P$, which we then use to calculate the maximum number of processors an optimal placement can contain.

Proposition 1 *Any subset P of nodes of the d-dimensional k-torus can be bisected by removing $O(k^{d-1})$ edges of T_k^d. In particular, any subgraph of T_k^d has bisection width $O(k^{d-1})$.*

Proof Omitted. See Appendix I. $\square$

The constant in $O(k^{d-1})$ of Proposition 1 is no larger than $6d$ when we consider directed edges. As a particular case of the proposition we view a placement P as a subgraph of T_k^d with no edges. Then

Corollary 1 *The d-dimensional k-torus T_k^d has bisection width of at most $6dk^{d-1}$ with respect to any placement P, i.e. $|\partial_b P| \leq 6dk^{d-1}$.*

Remark
Although the assertion of Proposition 1 appears intuitive because of its geometric nature for $d = 2$ and $d = 3$, it is easy to come up with examples of general graphs for which the bisection width of subgraphs can become arbitrarily far from that of the original graph. As an example, two copies of the complete graph K_{2n} on $2n$ nodes joined by a single edge has bisection width 1. Its subgraphs with $2n$ nodes have bisection widths ranging from 1 to $\Omega(n^2)$, depending on how evenly the $2n$ nodes are distributed among the two copies of K_{2n}.

Maximum Placement Size

An upper bound for the maximum number of processors an optimal placement can contain can now be obtained by substituting the bound for $|\partial_b P|$ given in the corollary into the inequality (6), while at the same time insuring that $\mathcal{E}_{max} = O(|P|)$, i.e. the load remains linear in the number of processors in the placement.

$$\mathcal{E}_{max} \geq \frac{2(\frac{|P|}{2})^2}{|\partial_b P|} \geq \frac{2(\frac{|P|}{2})^2}{6dk^{d-1}} \quad \Rightarrow \quad c_1|P| \geq \frac{2(\frac{|P|}{2})^2}{c_2 k^{d-1}} \quad \Rightarrow \quad |P| \leq ck^{d-1}$$

for $c_2 = 6d$ and $c = 2c_1c_2$ for some constant c_1. That is, the size of an optimal placement in T_k^d is $O(k^{d-1})$. Thus, we are justified in seeking placements which have ck^{d-1} processors, for some constant c. However, for ease of exposition, we will give analyses of placements of size k^{d-1} first (i.e. $c = 1$). Subsequently, we also consider the construction of placements with $c > 1$.

4 α-Separator Width with respect to a Placement

From corollary 1, we know that the bisection width of T_k^d with respect to a placement P is no larger than $6dk^{d-1}$. The lower bound on maximum load that one can obtain using this result is a function of dimension d, however. In this section we show that given a placement P on T_k^d with $|P| = \Theta(k^{d-1})$, it is possible to divide the network into two parts each having $\Theta(|P|)$ processors by removing $O(k^{d-1})$ edges, where the constants involved in $\Theta(|P|)$ and $O(k^{d-1})$ are independent of d. This will be useful in obtaining a better lower bound for load on edges than (4). We give a proof of this result in Theorem 1, assuming that the number of processors placed in subtori of T_k^d are given by "reasonable" functions of k, such as polynomials. Theorem 1 holds in general without this assumption, see Appendix II.

Theorem 1 *Relative to a placement P of size $\Theta(k^{d-1})$, T_k^d has an edge separator $\partial_\alpha P$ of size $O(k^{d-1})$ which splits T_k^d into two parts each with $\Theta(k^{d-1})$ processors (specifically, αk^{d-1} and $(1-\alpha)k^{d-1}$ processors, respectively, for some α, $0 < \alpha < 1$), where the constants are independent of d.*

Proof

Let us fix a dimension. There are k copies of a lower dimensional torus T_k^{d-1} embedded along this dimension, indexed from 0 through $k-1$. Let $a_j(k)$ be the number of processors P has in the j^{th} subtorus, $0 \leq j \leq k-1$. There are two cases to consider depending on whether or not $a_i(k) = \Omega(k^{d-1})$ for some i, $0 \leq i \leq k-1$. First we assume that there is such an $a_i(k)$. The situation is especially simple if there are *two* indices i, j for which $a_i(k) = \Omega(k^{d-1})$ and $a_j(k) = \Omega(k^{d-1})$. Then we remove the edges between subtori i and $i+1$ ($mod\ k$) and subtori j and $j+1$ ($mod\ k$), separating T_k^d into two parts each with $\Theta(k^{d-1})$ processors. The total number of edges removed is $\Theta(k^{d-1})$ $(= 4k^{d-1})$.

If there is only a single index i for which $a_i(k) = \Omega(k^{d-1})$, then if $\sum_{j \neq i} a_j(k) = \Omega(k^{d-1})$, an argument as above can again be used to split P by removing $4k^{d-1}$ edges. We can thus assume that $\sum_{j \neq i} a_j(k) = o(k^{d-1})$. This time we cannot separate T_k^d using the argument above however. Now let S_k^{d-1} be the subtorus for which $a_i(k) = \Omega(k^{d-1})$. By Proposition 1, S_k^{d-1} can be *bisected* relative to the processors it contains by removing no more than $6dk^{d-2}$ edges. Therefore T_k^d has a bisection relative to P obtained by removing at most $6dk^{d-2} + 4d\sum_{j \neq i} a_j(k)$ edges. This is because there are a total of $4d \sum_{j \neq i} a_j(k)$ edges incident on the processors in all the subtori excluding S_k^{d-1}. Therefore

$$|\partial_\alpha P| \leq 6dk^{d-2} + 4d \sum_{j \neq i} a_j(k)$$

$$|\partial_\alpha P| \leq 6dk^{d-2} + o(k^{d-1}) \ = \ o(k^{d-1})$$

whenever $d < k$. Thus, it is possible to split the T_k^d into two parts, each containing $\Theta(k^{d-1})$ processors by removing no more than $O(k^{d-1})$ (actually $o(k^{d-1})$) edges.

Now consider the case that no $a_i(k)$ is $\Omega(k^{d-1})$. Then $a_i(k) = o(k^{d-1})$, for all i, and at the same time $\sum_{j=0}^{k-1} a_j(k) = \Theta(k^{d-1})$. We outline the proof here for the case when the $a_i(k)$ are polynomial functions of k (It follows from Proposition 2 in Appendix II that this assumption is not necessary). Let $U = \{i \mid a_i(k) = \Omega(k^{d-2}),\ 0 \leq i \leq k-1\}$. Note that $|U|$ must be $\Omega(k)$, since otherwise $|P|$ could not be $\Theta(k^{d-1})$. Since $a_i(k)$ are polynomial functions and all $a_i(k) = o(k^{d-1})$, the subtori whose indices are in U must each have $\Theta(k^{d-2})$ processors. Furthermore this forces $|U| = \Theta(k)$. Thus we can find ρk subtori that have $\Theta(k^{d-2})$ processors, $0 < \rho < 1$. Consequently we can split T_k^d into two parts each having $\frac{\rho}{2}k\Theta(k^{d-2}) = \Theta(k^{d-1})$ processors by removing $O(k^{d-1})$ $(= 4k^{d-1})$ edges. $\square$

5 An Improved Lower Bound for Maximum Load

We have shown in the previous section that given a placement P of size c_1k^{d-1}, it is possible to split T_k^d into two parts having $\alpha|P|$ and $(1-\alpha)|P|$ processors by removing at most c_2k^{d-1} edges, for some constants c_1, c_2, and α, $0 < \alpha < 1$. We use this result to establish a lower bound on load which shows that the lower bound $\mathcal{E}_{max} \geq (|P|-1)/2d$ becomes too small as the parameter d grows larger.

Taking $|P| = c_1k^{d-1}$, $|S| = \alpha|P|$ and $|\partial S| = c_2k^{d-1}$ in (5), we have

$$\mathcal{E}_{max} \geq \frac{2\alpha|P|(1-\alpha)|P|}{c_2k^{d-1}} \geq \frac{2\alpha c_1k^{d-1}(1-\alpha)c_1k^{d-1}}{c_2k^{d-1}} \geq ck^{d-1}$$

where $c = 2\alpha(1-\alpha)c_1^2/c_2$. Note that the constant c is independent of parameter d. Hence, this lower bound comes to characterize the quantity $\mathcal{E}_{max}$ more closely than (4) as the parameter d grows. We will use this lower bound to gauge the optimality of the placements and routing algorithms that we give next.

6 Linear Placements

We have established in section 3 that optimal placements have $\Theta(k^{d-1})$ processors. In this section, we introduce the notion of a *linear placement*: these are placements in which the coordinates of each processors in P satisfy a particular type of a linear equation over $\mathbb{Z}_k$.

Definition 11 *A placement P on T_k^d which satisfies*

$$P = \{\vec{p} \mid c_1p_1 + c_2p_2 + \ldots + c_dp_d \equiv c \ (mod\ k)\} \tag{7}$$

where $c \in \mathbb{Z}_k$, and at least one of $c_i \in \mathbb{Z}_k$ is relatively prime to k is called a linear placement.

For simplicity, we will use placements where $c_1 = c_2 = \ldots = c_d = 1$, even though our analyses apply to linear placements in general form (i.e. (7)) provided that c_1 and c_d are relatively prime to k. Note that there are exactly k^{d-1} processors satisfying the expression $p_1 + p_2 + \ldots + p_d \equiv c \ (mod\ k)$ for any specific $c \in \mathbb{Z}_k$. Originally, linear placements of this form were used in three dimensional tori by Blaum et al. [3, 4], where they were called *shifted diagonal placements.*

We can also specify placements of size tk^{d-1} where t is a fixed integer less than k. For instance, the placement $P = P_1 \cup P_2 \cup \ldots \cup P_t$ where

$$\begin{aligned} P_1 &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 0 \ (mod\ k)\} \\ P_2 &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 1 \ (mod\ k)\} \\ &\vdots \\ P_t &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv t-1 \ (mod\ k)\} \end{aligned}$$

has tk^{d-1} processors. We shall call such placements *multiple linear placements.*

Remark

We would like to point out that linear (and multiple linear) placements themselves do not guarantee the linearity of the load on edges. *Linear* only refers to the fact that the coordinates of the processors in the placement satisfy a linear equation over $\mathbb{Z}_k$. We still need to construct routing algorithms which enable communication between pairs of processors in a way that yields load that is linear in $|P|$.

In the remaining sections, we will specify different routing algorithms and analyze their maximum communication load on edges. As we have mentioned earlier, the routing algorithms will use minimal (shortest) paths between processors. To deliver a message from processor $\vec{p}$ to $\vec{q}$, the value of $\vec{p}$ in each dimension is "corrected" towards the corresponding value in $\vec{q}$ by the amount and direction ($\pm$) dictated by the shortest cyclic distance between the values in that dimension. The exact way of correcting the dimensions to route the packets is specified by the routing algorithm.

We consider two classes of routing algorithms and the analysis of the load in each case both for linear and multiple linear placements: *Ordered Dimensional Routing (ODR)* and *Unordered Dimensional Routing (UDR).*

7 The Ordered Dimensional Routing Algorithm (ODR)

The algorithm is simple. Given a placement P on T_k^d to route a packet from $\vec{p} = (p_1,\ p_2, \ldots,\ p_d)$ to $\vec{q} = (q_1,\ q_2, \ldots,\ q_d)$, both in P:

for i := *1* **to** d **do**

Correct p_i *in the direction of shortest cyclic distance*

That is, the routing path will include the following nodes:

$$\vec{p} \rightarrow (q_1,\ p_2, \ldots,\ p_d) \rightarrow (q_1,\ q_2,\ p_3, \ldots,\ p_d) \rightarrow \ldots \rightarrow (q_1,\ q_2, \ldots,\ q_{d-1},\ p_d) \rightarrow \vec{q}.$$

Note that if k is odd, $|\mathcal{C}^{ODR}_{\vec{p}\rightarrow\vec{q}}| = 1$, i.e. there is only 1 path specified by the ODR algorithm for any given $\vec{p}$ and $\vec{q} \in P$. However, when k is even the ODR algorithm may result in multiple paths between some pairs of processors in the placement. To aid in the analysis, we will use the following (restricted) version which ensures the existence of only one canonical routing path between any given pair of processors regardless of the parity of k.

for i := *1* **to** d **do**
begin
if *there is more than one way of correcting* p_i **then**
Pick the path that corrects p_i *in the (+) direction* (mod k);
Correct p_i *in the direction of shortest cyclic distance*
end

Thus if there are two choices for some p_i, q_i coordinate pair, the algorithm routes through $p_i + 1\ (mod\ k)$, $p_i + 2\ (mod\ k)$, $\ldots$, q_i. The shortcoming of having only one path between a pair of processors is the lack of fault-tolerance in the network. Specifically, if an edge over which a pair of processors communicate fails then the pair will no longer be able to exchange messages. In section 8 we look at another routing algorithm which does not suffer from this limitation.

7.1 Load Analysis for Linear Placements with ODR

Theorem 2 *Given a linear placement* $P = \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 0\ (mod\ k)\}$ *in* $T^d_k = (V, E)$*,* Ordered Dimensional Routing Algorithm *results in linear load on edges.*

Proof Since ODR algorithm ensures one path between each pair of processors, each denominator in the expression (3) for $\mathcal{E}(l)$ is 1. Thus, in order to compute $\mathcal{E}_{max}$, we need only to count the (maximum) number of pairs of processors that communicate through a specific edge. Without loss of generality, consider an edge $l \in E$, where

$$l = <(i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d)> .$$

We will count pairs of processors which communicate using l. Let $\vec{p}$ and $\vec{q} \in P$ be two processors where $\vec{p}$ sends messages to $\vec{q}$ through l. Since ODR algorithm is used, we must have

$$\vec{p} = (p_1, \ldots, p_s, i_{s+1}, \ldots, i_d)$$

and

$$\vec{q} = (i_1, \ldots, i_{s-1}, q_s, q_{s+1}, \ldots, q_d)$$

with $p_s \leq i_s$ and $q_s \geq i_s + 1$. Since $\vec{p}$ and $\vec{q}$ are both in P and P is linear, the coordinates satisfy

$$p_1 + \ldots + p_s + i_{s+1} + \ldots + i_d \equiv 0 \ (mod\ k)$$

and

$$i_1 + \ldots + i_{s-1} + q_s + \ldots + q_d \equiv 0 \ (mod\ k).$$

Therefore

$$p_1 + \ldots + p_s \equiv c_1 \ (mod\ k) \tag{8}$$

$$q_s + \ldots + q_d \equiv c_2 \ (mod\ k) \tag{9}$$

Note that if $\vec{p}$ and $\vec{q}$ are to use the edge l then, in order to ensure that messages follow shortest paths, we must also have,

$$q_s - p_s \ (mod\ k) \leq p_s - q_s \ (mod\ k)$$

by the property of cyclic distance. The number of processors satisfying equation (8) is less than or equal to k^{s-1} while those satisfying (9) is less than or equal to k^{d-s}. Thus, the total number of processor pairs cannot be more than $k^{s-1}k^{d-s} = k^{d-1}$. Therefore, $\mathcal{E}_{max} \leq k^{d-1}$ and the maximum load is linear in $|P| = k^{d-1}$.

Note that we are actually overcounting since we have not taken the restrictions $p_s \leq i_s$, $q_s \geq i_s + 1$ and $q_s - p_s \ (mod\ k) \leq p_s - q_s \ (mod\ k)$ on the s-th dimension into account when we count the solutions of (8) and (9). These conditions affect the choices of p_s and q_s. A more accurate expression (though of the same order) can be obtained by paying closer attention to these parameters: To determine the number of different ways p_s and q_s may be chosen, consider the *1-dimensional k-subtorus* (ring) on which the edge l lies. Assume first that k is even. Without loss of generality, also assume that the nodes in the ring are enumerated from 0 to $k-1$ such that $i_s = k/2 - 1$. Then, the ODR algorithm will use l to deliver messages from node 0 to only node $k/2$ on this ring. Similarly, it will use edge l for messages from node 1 to node $k/2$ and from node 1 to node $k/2+1$ and so on. Messages from node $k/2-1$ $(= i_s)$ can be sent using l to any node indexed $k/2$ to $k-1$. The total number of choices for p_s and q_s will therefore be

$$\frac{\frac{k}{2}(\frac{k}{2}+1)}{2} = \frac{k^2}{8} + \frac{k}{4}$$

Now assume that k is odd, and also that $i_s = (k-1)/2$. In this case, messages from node 1 can be delivered to only node $(k+1)/2$ through l while messages from node 2 can be routed to nodes $(k+1)/2$ and $(k+1)/2+1$, and so on. Thus there are a total of

$$\frac{\frac{k-1}{2}(\frac{k-1}{2}+1)}{2}$$

choices for p_s and q_s when k is odd.

Therefore the number of solutions to equations (8) and (9) which satisfy the conditions $p_s \leq i_s$, $q_s \geq i_s + 1$ and $q_s - p_s \ (mod\ k) \leq p_s - q_s \ (mod\ k)$ is

$$\mathcal{E}_{max} = \frac{k^{d-1}}{8} + \frac{k^{d-2}}{4}$$

when k is even, and

$$\mathcal{E}_{max} = \frac{k^{d-1}}{8} - \frac{k^{d-3}}{8}$$

when k is odd. □

Therefore regardless of the parity of k, for a linear placement P with ODR, $|P| = k^{d-1}$ and

$$\mathcal{E}_{max} = \frac{k^{d-1}}{8} + o(k^{d-1}) \ .$$

7.2 Multiple Linear Placements with ODR

Theorem 3 *Multiple linear placements along with ODR algorithm on T_k^d results in linear load on edges.*

Proof The analysis is conceptually similar to that of the previous section. Consider a multiple linear placement $P = P_1 \cup P_2 \cup \ldots \cup P_t$ in $T_k^d = (V, E)$ with ODR where for some fixed constant $t < k$

$$\begin{aligned}
P_1 &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 0 \ (mod\ k)\} \\
P_2 &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 1 \ (mod\ k)\} \\
&\vdots \\
P_t &= \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv t-1 \ (mod\ k)\}
\end{aligned}$$

Note that $|P| = tk^{d-1}$. As before, consider an edge $l \in E$ of the form

$$l = < (i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d) >$$

and a pair of processors $\vec{p}$, $\vec{q} \in P$, which communicate using l. Since *ODR* algorithm is used, we must have

$$\vec{p} = (p_1, \ldots, \ p_s, \ i_{s+1}, \ \ldots, \ i_d)$$

and

$$\vec{q} = (i_1, \ldots,\ i_{s-1},\ q_s,\ q_{s+1}, \ldots,\ q_d)$$

with $p_s \leq i_s$ and $q_s \geq i_s + 1$ as well as $q_s - p_s\ (mod\ k) \leq p_s - q_s\ (mod\ k)$. Since $\vec{p}$ and $\vec{q}$ are both in P, each must satisfy an equation among $P_1, P_2, \ldots, P_t$. Thus

$$p_1 + \ldots + p_s + i_{s+1} + \ldots + i_d \equiv 0\ or\ 1\ or\ \ldots\ or\ t-1\ (mod\ k) \tag{10}$$

and

$$i_1 + \ldots + i_{s-1} + q_s + \ldots + q_d \equiv 0\ or\ 1\ or\ \ldots\ or\ t-1\ (mod\ k) \tag{11}$$

The number of solutions to equations (10) is no more than tk^{s-1}. Similarly, the number of solutions to equations (11) is no more than tk^{d-s}. Therefore, the total number of processor pairs communicating through l is bounded by t^2k^{d-1}, which is linear in $|P|$ for constant t. □

8 Unordered Dimensional Routing (UDR)

We mentioned in section 7 that ODR algorithm suffers from lack of fault-tolerance, since there is only one path between each pair of processors. In this section, we introduce *Unordered Dimensional Routing (UDR)*, which eliminates this problem. The algorithm is as follows: To route a packet from $\vec{p} = (p_1,\ p_2, \ldots,\ p_d)$ to $\vec{q} = (q_1,\ q_2, \ldots,\ q_d)$, both in P

for *i := 1* **to** *d* **do**
  begin
    Select a number j from the set $\{1, 2, \ldots, d\}$ *that has not been used before;*
    Correct p_j *in the direction of shortest cyclic distance*
  end

As was the case in ODR, a dimension is corrected completely before another is selected. Unlike ODR, however, the order in which the dimension to be corrected next is picked is arbitrary. This algorithm thus provides multiple paths for each pair of processors and improves the fault-tolerance of the system. If $\vec{p}$ and $\vec{q}$ are two processors differing in s dimensions, then there will be $s!$ different paths from $\vec{p}$ to $\vec{q}$ in UDR, i.e. $|\mathcal{C}^{UDR}_{\vec{p}\to\vec{q}}| = s!$. Next we show that UDR algorithm results in linear load in edges.

8.1 Load Analysis for Linear Placements with UDR

For a linear placement P which uses UDR algorithm, the load on an edge l is

$$\mathcal{E}(l) = \sum_{\vec{p}\in P, \vec{q}\in P} \frac{|\mathcal{C}^{UDR}_{\vec{p}\to l\to\vec{q}}|}{|\mathcal{C}^{UDR}_{\vec{p}\to\vec{q}}|}$$

Since there exist some pairs of processors for which $|\mathcal{C}^{UDR}_{\vec{p}\to\vec{q}}| > 1$ we have,

$$\mathcal{E}(l) < \sum_{\vec{p}\in P, \vec{q}\in P} |\mathcal{C}^{UDR}_{\vec{p}\to l\to\vec{q}}|$$

The upper bound on the right hand side of this inequality specifies the number of messages sent between pairs of processors which could "potentially" route their messages through l. Such processors can also use other paths that do not include l, since UDR algorithm provides multiple routing paths.

Theorem 4 *Given a linear placement* $P = \{\vec{p} \mid p_1 + p_2 + \ldots + p_d \equiv 0 \ (mod\ k)\}$ *in* $T^d_k = (V, E)$, Unordered Dimensional Routing Algorithm *results in linear load on edges.*

Proof Without loss of generality $l \in E$ is of the form

$$l =< (i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d) > \ .$$

Suppose $\vec{p}$ and $\vec{q} \in P$ are two processors communicating through l. Our aim is to find an upper bound on the number of pairs of processors communicating through l. A moment of thought reveals that $\vec{p}$ and $\vec{q}$ must have either $p_j = i_j$ and q_j arbitrary or $q_j = i_j$ and p_j arbitrary, for $j \neq s$. This means the number of possible choices in dimension j is less than $2k$. Hence, the total number of choices for all of the coordinates of $\vec{p}$ and $\vec{q}$ excluding s is less than $2^{d-1}k^{d-1}$. Since $\vec{p}$ and $\vec{q}$ are both in P they satisfy

$$p_1 + \ldots + p_d \equiv 0 \ (mod\ k)$$

and

$$q_1 + \ldots + q_d \equiv 0 \ (mod\ k)$$

There are at most one solution pair for each one of $2^{d-1}k^{d-1}$ choices (as before, the coordinates in the s-th dimension are restricted by the conditions $p_s \leq i_s$, $q_s \geq i_s + 1$ and $q_s - p_s \ (mod\ k) \leq p_s - q_s \ (mod\ k)$). Therefore, the total number of processor pairs communicating through l is bounded by $2^{d-1}k^{d-1}$. i.e.

$$\mathcal{E}_{max} < \max_{l\in E} \ \{\sum_{\vec{p}\in P, \vec{q}\in P} |\mathcal{C}^{UDR}_{\vec{p}\to l\to\vec{q}}|\} \ < \ 2^{d-1}k^{d-1}$$

which is linear in $|P| = k^{d-1}$ for any fixed d. □

8.2 Multiple Linear Placements with UDR

Theorem 5 *Multiple linear placements along with UDR algorithm on* T^d_k *results in linear load on edges.*

Proof We have $|P| = tk^{d-1}$. As before, consider a processor pair $\vec{p}$, $\vec{q} \in P$ which communicate using l where

$$l = <(i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d)> \ .$$

The number of choices for processor pairs using l is strictly less than $2^{d-1}k^{d-1}$, as in the case of linear placements with UDR. Since there are t equations for each of $\vec{p}$ and $\vec{q}$, there are t^2 solutions for every one of $2^{d-1}k^{d-1}$ choices of pairs. Therefore the number of pairs of processors communicating through l is less than $t^2 2^{d-1}k^{d-1}$, which is linear in $|P|$ for any fixed constant $t < k$. □

9 Conclusion

Following the work of Blaum, Bruck, Pifarré, and Sanz [3, 4], we have considered communication in partially-populated torus networks in terms of placements of processors and associated routing algorithms. We have provided lower bounds for the maximum load under the all-to-all communication scenario, and found bounds on the size of an optimal placement. We have shown that arbitrary placements can be bisected by removing a set of edges of the same order as the bisection width of the torus. We then provided optimal placements of size $\Theta(k^{d-1})$ on the d-dimensional k-torus using what we call linear and multiple linear placements, and gave load analyses of each under two different routing algorithms.

There are some interesting combinatorial properties of placements still to be resolved. Among these are the characterization of optimal placements in terms of restrictions to subtori and an extensive analysis of the properties of edge separators of tori relative to optimal placements.

References

[1] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield and B. Smith. *The Tera Computer System.* ACM Supercomputing, 1990.

[2] Myung M. Bae and Bella Bose. *Resource Placement in Torus-Based Networks.* IEEE Transactions on Computers, Vol. 46 (10), pp. 1083–1092, 1997.

[3] Mario Blaum, Jehoshua Bruck, Gustavo D. Pifarré, and Jorge L. Sanz. *On Optimal Placements of Processors in Tori Networks.* Proc. of the 8^{th} IEEE Symp. on Parallel and Distributed Processing, pp. 552–555, Oct. 1996.

[4] Mario Blaum, Jehoshua Bruck, Gustavo D. Pifarré, and Jorge L. Sanz. *On Optimal Placements of Processors in Fault-Tolerant Tori Networks.* preprint, 1997.

[5] Bella Bose, Bob Broeg, Younggeun Kwon and Yaagoub Ashir. *Lee Distance and Topological Properties of k-ary n-cubes.* IEEE Transactions on Computers, Vol. 44 (8), pp. 1021–1030, August 1995.

[6] Yu-Chee Tseng, Ting-Hsien Lin, Sandeep K. S. Gupta, Dhabaleswar K. Panda. *Bandwidth-optimal Complete Exchange on Wormhole-routed 2D/3D Torus Networks : A Diagonal-Propagation Approach.* IEEE Transactions on Parallel and Distributed Systems, Vol. 8 (4), pp. 380–396, April 1997.

[7] A. Gerbessiotis and L. G. Valiant. *Direct Bulk-Synchronous Parallel Algorithms.* J. of Parallel and Dist. Computing, 22, pp. 251–267, 1994.

[8] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays· Trees· Hypercubes.* Morgan Kaufmann Publishers, San Mateo, California, 1992.

[9] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, The Netherlands, 1977.

[10] Lionel M. Ni and Philip K. McKinley. *A Survey of Wormhole Routing Techniques in Direct Networks.* Computer, Vol. 26 (2), pp. 62–76, Feb. 1993.

[11] F.Pitteli and D. Smitley. *Analysis of a 3D Toroidal Network for a Shared Memory architecture.* Proc. of Supercomputing '88, pp. 35–41, Nov. 1988.

[12] S. Rao, T. Suel, T. Tsantilas, and M. Goudreau. *Efficient Communication Using Total Exchange.* Proc. of IPPS'95, pp. 544–550, 1995.

[13] Jop F. Sibeyn. *Routing on Triangles, Tori and Honeycombs.* Int. J. of Foundations of Computer Sci., Vol. 8 (3), pp. 269–287, 1997.

[14] L. G. Valiant. *A Bridging Model for Parallel Computation.* Comm. of the ACM, Vol. 33 (8), pp. 103–111, 1990.

Appendix I

In the following proposition we treat T_k^d as an undirected graph, rather than a directed one. Thus each edge below translates into a pair of edges of the directed case.

Proposition 1 *Any subset P of nodes of the d-dimensional k-torus can be bisected by removing $O(k^{d-1})$ edges of T_k^d. In particular, any subgraph of T_k^d has bisection width $O(k^{d-1})$.*

Proof Since T_k^d has only $O(k^{d-1})$ $(= dk^{d-1})$ more edges than the d-dimensional k-ary array A_k^d, we can instead work with A_k^d.

Let $\mathbb{Z}^d$ denote the collection of all lattice points in the d-dimensional Euclidean space $\mathbb{R}^d$. The length of $\vec{x} \in \mathbb{R}^d$ is denoted by $\|\vec{x}\|$. A *hyperplane* $\mathcal{H}_t = \mathcal{H}_t(\vec{\eta})$ in $\mathbb{R}^d$ is defined by a unit vector $\vec{\eta} = \vec{\gamma}/\|\vec{\gamma}\|$ and a real number t, as the set of points $\vec{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ satisfying

$$\vec{\eta} \cdot \vec{x} = \eta_1 x_1 + \eta_2 x_2 + \cdots + \eta_d x_d = t \ .$$

As t ranges over $\mathbb{R}$, $\mathcal{H}_t$ sweeps $\mathbb{R}^d$ in the direction $(\pm)$ of $\vec{\eta}$. Suppose we pick γ to be a transcendental number (such as π or e) and define $\mathcal{H}_t$ by the unit vector $\vec{\eta}$ in the direction of $(1, \gamma, \gamma^2, \ldots, \gamma^{d-1})$. Then there can be no two distinct points $\vec{a}, \vec{b} \in \mathbb{Z}^d$ which lie on $\mathcal{H}_t$ for a fixed t. For otherwise

$$(a_1 - b_1) + (a_2 - b_2)\gamma + \cdots + (a_d - b_d)\gamma^{d-1} = 0,$$

with not all $a_i - b_i = 0$, and this would contradict the fact that γ is transcendental.

The standard embedding of A_k^d in $\mathbb{R}^d$ is obtained by mapping each vertex $(a_1, a_2, \ldots, a_d)$ of A_k^d where $0 \le a_i < k$ for each i, to the point $\vec{a} = (a_1, a_2, \ldots, a_d) \in \mathbb{Z}^d$. An edge in A_k^d between two vertices $\vec{a} = (a_1, \ldots, a_i, \ldots, a_d)$ and $\vec{b} = (a_1, \ldots, a_i + 1, \ldots, a_d)$ is represented by the unit line segment in $\mathbb{R}^d$ parallel to the i-th axis, having $\vec{a}$ and $\vec{b}$ as its endpoints. We identify P (nodes + edges) with its geometric embedding in $\mathbb{R}^d$ as a subgraph of A_k^d.

Consider the collection of hyperplanes $\mathcal{H}_t$ defined as above using a transcendental number γ in the range $1 < \gamma < \sqrt[d-1]{2}$ and $0 \le t \le (k-1)$. For t in this range, $\mathcal{H}_t$ sweeps A_k^d from the point $(0, 0, \ldots, 0)$ to the farthest point $(k-1, k-1, \ldots, k-1)$ in the direction of $\vec{\eta}$. Note that by our choice of γ,

1. $1 < \gamma < \cdots < \gamma^{d-1} < 2$,
2. $0 < \eta_1 < \eta_2 < \cdots < \eta_d < 1$,
3. For any $r \ge 2$ and $i = 1, 2, \ldots, d$, $r\gamma^{i-1} \ge 2 > \gamma^{d-1}$, and consequently $r\eta_i > \eta_d$.

For a fixed value $t = t_0$, $\mathcal{H}_t$ crosses an edge between $\vec{a}$ and $\vec{b}$ of A_k^d iff these two endpoints lie on opposite sides of the hyperplane:

$$\vec{a} \cdot \vec{\eta} < t_0 < \vec{b} \cdot \vec{\eta} = \vec{a} \cdot \vec{\eta} + \eta_i \ .$$

Without any loss of generality, assume that there is no vertex of A_k^d on $\mathcal{H}_{t_0}$. Since an edge in the embedding has unit length, the distance from $\vec{a}$ to $\mathcal{H}_{t_0}$ (and also from $\vec{b}$ to $\mathcal{H}_{t_0}$) is strictly less than 1. The distance between $\vec{x}$ and the hyperplane $\mathcal{H}_{t_0}$ is given by

$$|\vec{x} \cdot \vec{\eta} - t_0|.$$

Therefore the number S of edges crossed by $\mathcal{H}_{t_0}$ is no larger than the number of solutions $\vec{a} \in A_k^d$ of

$$t_0 - 1 < \vec{a} \cdot \vec{\eta} < t_0 \quad \textit{and} \quad t_0 < \vec{a} \cdot \vec{\eta} + \eta_i < t_0 + 1 \ , \tag{12}$$

summed over $i = 1, 2, \ldots, d$. The two sets of inequalities in (12) are in turn equivalent to

$$t_0 - \eta_i < \vec{a} \cdot \vec{\eta} < t_0 \ . \tag{13}$$

Let

$$S_i = \{\vec{a} \in A_k^d \mid t_0 - \eta_i < \vec{a} \cdot \vec{\eta} < t_0 \ \}.$$

Then

1. $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_d$, (since $\eta_1 < \eta_2 < \cdots < \eta_d$);
2. $S = |S_1| + |S_2| + \cdots + |S_d|$.

We claim that $|S_d| \leq 2k^{d-1}$, and therefore $S \leq 2dk^{d-1}$.

Let W be a subset of nodes of A_k^d. Define the *discrepancy* $D(W) = d-$ number of indices j such that the j-th coordinate of each element in W is identical. For example when $W = \{(0,0,0), (0,0,2), (1,0,0), (1,0,2)\}$, $D(W) = 2$. Let $N[d,k]$ be the maximum cardinality collection $\mathcal{N}$ of $\vec{a} \in A_k^d$ such that any subset W of 3 elements has $D(W) \geq 2$. Grouping the elements in $\mathcal{N}$ according to their first coordinates, we obtain the bound

$$N[d,k] \leq kN[d-1,k]$$

for $d > 2$. Also by the pigeonhole principle $N[2,k] \leq 2k$. Therefore $N[d,k] \leq 2k^{d-1}$. Now by way of contradiction, assume that $|S_d| > 2k^{d-1}$. We have just shown with our count of $\mathcal{N}$ that S_d must contain a set W consisting of 3 vectors such that $D(W) = 1$, i.e. three vectors which differ only in a single coordinate i. In particular, S_d has two solutions $\vec{a} = (a_1, \ldots, a_i, \ldots, a_d)$ and $(a_1, \ldots, a_i + r, \ldots, a_d)$ with $r \geq 2$ (this is guaranteed since we have 3 distinct numbers in the i-th coordinates in W). Using (13), this implies

$$t_0 - \eta_d < \vec{a} \cdot \vec{\eta} < t_0 \quad \textit{and} \quad t_0 - \eta_d < \vec{a} \cdot \vec{\eta} + r\eta_i < t_0$$

which combine to give $t_0 - \eta_d < t_0 - r\eta_i$, or equivalently $r\eta_i < \eta_d$. However, by our choice of γ, $r\eta_i \geq \eta_d$. This contradiction proves that $|S| \leq 2dk^{d-1}$. Therefore for any value of t, $0 < t < k-1$, the hyperplane $\mathcal{H}_t$ crosses no more than $2dk^{d-1}$ edges of A_d^k.

Now we can prove that P (or P minus an element, if $|P|$ is odd) can be divided into two equal size subsets by removing no more than $2dk^{d-1}$ edges of A_d^k. This division is obtained by using the fact that $\mathcal{H}_t$ contains at most one element of A_d^k, and therefore of P for any t. Thus the number of elements of P which are on the origin side of $\mathcal{H}_t$ goes up by units of one with increasing t, and is equal to $|P|/2$ for some $t = t_0$. $\square$

Appendix II

Proposition 2 *Suppose we are given a triangular array of non-negative numbers* $b_{k,i}$, $1 \leq i \leq k$ *:*

$$\begin{array}{lll} b_{1,1} & & \\ b_{2,1} & b_{2,2} & \\ b_{3,1} & b_{3,2} & b_{3,3} \\ \vdots & & \ddots \end{array}$$

Consider the row sums $S_k = b_{k,1} + b_{k,2} + \cdots + b_{k,k}$. *Suppose*

1. *As* $k \to \infty$, $b_{k,i} \to 0$ *for every* i,
2. $S_k = \Theta(1)$. [1]

Then we can write $S_k = A_k + B_k$ *with* $A_k = b_{k,1} + b_{k,2} + \cdots + b_{k,\alpha k}$ *for some* α *with* $0 < \alpha < 1$ *so that* $A_k = \Theta(1)$ *and* $B_k = \Theta(1)$.

Proof Without loss of generality assume that $c_1 = 1$. Otherwise we replace each $b_{k,i}$ with $b_{k,i}/c_1$ and c_2 by c_2/c_1. First assume that $1 \leq S_k$ for all k, as this case contains the essential argument of the proof.

Given $\epsilon > 0$, we can find a K such that for $k \geq K$, $b_{k,i} < \epsilon$ for $i = 1, 2, \ldots, k$. For a fixed $k \geq K$ put $b_i = b_{k,i}$ to simplify the notation. For ϵ sufficiently small, we can find indices l and r such that the sum of the first l, and the sum of the last r elements of the k-th row are relatively large:

$$\begin{array}{rcl} b_1 + b_2 + \cdots + b_{l-1} \leq \dfrac{1}{8} & \text{and} & b_1 + b_2 + \cdots + b_l > \dfrac{1}{8} \ , \\ b_k + b_{k-1} + \cdots + b_{k-r+2} \leq \dfrac{1}{8} & \text{and} & b_k + b_{k-1} + \cdots + b_{k-r+1} > \dfrac{1}{8} \end{array}$$

Since $b_i < \epsilon$ for all i, this implies

$$\frac{1}{8} < b_1 + b_2 + \cdots + b_l < \frac{1}{8} + \epsilon \tag{14}$$

and

$$\frac{1}{8} < b_k + b_{k-1} + \cdots + b_{k-r+1} < \frac{1}{8} + \epsilon \ . \tag{15}$$

Since $S_k \geq 1$, the middle block of b_i's of length $L_k = k - r - l$ from b_{l+1} to b_{k-r} cannot have too small a sum. More precisely

$$b_{l+1} + b_{l+2} + \cdots + b_{l+L_k} > \frac{3}{4} - 2\epsilon \ , \tag{16}$$

[1] I.e. there are constants $c_1, c_2 > 0$ such that $S_k \leq c_2$ for all k and $c_1 \leq S_k$ for infinitely many k.

for otherwise adding the b_i terms in (14), (15), and (16) would give $S_k < 1$. Using the fact that $b_i < \epsilon$ for all i in (16) we obtain

$$\epsilon L_k > \frac{3}{4} - 2\epsilon$$

and consequently

$$L_k > \frac{3}{4}\epsilon^{-1} - 2 \ .$$

We would like to guarantee that L_k is a large enough fraction of k, say

$$L_k \geq \frac{2}{3}k - 1. \tag{17}$$

For this it suffices to have

$$\frac{3}{4}\epsilon^{-1} > \frac{2}{3}k,$$

or

$$k < \frac{9}{8}\epsilon^{-1}.$$

Since $1 \leq S_k$ and S_k has k terms, we already know that $\epsilon^{-1} < k$. Let $(a, b) = \{x \mid a < x < b\}$. To guarantee that we can pick $k \in (\epsilon^{-1}, \frac{9}{8}\epsilon^{-1})$, take a sequence ϵ_n going down to 0 such

$$\epsilon_{n+1} > \frac{8}{9}\epsilon_n.$$

Then

$$\epsilon_n^{-1} < \epsilon_{n+1}^{-1} < \frac{9}{8}\epsilon_n^{-1},$$

so that

$$\bigcup_{n=1}^{\infty} (\ \epsilon_n^{-1}, \frac{9}{8}\epsilon_n^{-1}\ ) = (\epsilon_1^{-1}, \infty)$$

Now as soon as the interval $(\epsilon_n^{-1}, \frac{9}{8}\epsilon_n^{-1})$ has length larger than 2 it must contain an integer. Therefore we can construct a sequence $k_1 < k_2 < \cdots$ such that

$$L_{k_n} \geq \frac{2}{3}k_n - 1 \ .$$

i.e. the lengths of the middle intervals for this subsequence satisfy (17). But in row k of the array $b_{k,i}$, $1 \leq i \leq k$, an interval of consecutive elements whose length is 2/3 the total length, necessarily contains the central element of the row. This means that if we bisect the rows by taking $\alpha = 1/2$, then

$$\begin{aligned} \frac{1}{8} < b_1 + b_2 + \cdots + b_l &\leq b_1 + b_2 + \cdots + b_{k,\frac{1}{2}k} \\ \frac{1}{8} < b_k + b_{k-1} + \cdots + b_{k-r+1} &\leq b_k + b_{k-1} + \cdots + b_{k,\frac{1}{2}k+1} \end{aligned}$$

for $k \in \{k_1, k_2, \ldots\}$. Therefore writing $S_k = A_k + B_k$ with $A_k = b_{k,1} + b_{k,2} + \cdots + b_{k,\frac{1}{2}k}$, both $A_k, B_k > \frac{1}{8}$ for $k \in \{k_1, k_2, \ldots\}$. Since both sequences are bounded above by c_2, $A_k = \Theta(1)$ and $B_k = \Theta(1)$ as desired.

Now assume that $S_k \geq 1$ on some infinite set $k^{(1)} < k^{(2)} < \cdots$, and not necessarily for all k. The crucial point in the above argument is the fact that

$$\bigcup_{n=1}^{\infty} (\ \epsilon_n^{-1}, \frac{9}{8}\epsilon_n^{-1}\ ) = (\epsilon_1^{-1}, \infty)$$

for appropriate ϵ_n. We can extract a subsequence of $\{k^{(1)}, k^{(2)}, \ldots\}$ just as we constructed $\{k_1, k_2, \ldots\}$ from $\{1, 2, \ldots\}$. On this subsequence $\alpha = \frac{1}{2}$ again suffices to produce A_k and $B_k = S_k - A_k$ with the desired properties. $\square$

Proposition 2 is applied to generalize the final argument in the proof of theorem 1 as follows. We are given k functions $f_i(k)$ for $i = 1, 2, \ldots, k$ (these correspond to the functions $a_i(k)$, $i = 0, 1, \ldots, k-1$ in theorem 1 such that

1. $f_i(k) = o(k^{d-1})$ for all i,
2. $\sum_{i=1}^{k} f_i(k) = \Theta(k^{d-1})$.

Let $b_{k,i} = f_i(k)/k^{d-1}$. Then the triangular sequence $b_{k,i}$ satisfies the conditions of proposition 2. The conclusion of the proposition cast in terms of the functions f_i is that for some α with $0 < \alpha < 1$,

$$\begin{aligned} f_1(k) + f_2(k) + \cdots + f_{\alpha k}(k) &= \Theta(k^{d-1}), \quad \textit{and} \\ f_{\alpha k+1}(k) + f_{\alpha k+2}(k) + \cdots + f_k(k) &= \Theta(k^{d-1}). \end{aligned}$$

This provides the decomposition into subtori required for the conclusion of theorem 1.