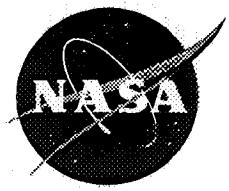
NASA Contractor Report 194996

ICASE Report No. 94-88



ICASE

ON *K*-ARY *N*-CUBES: THEORY AND APPLICATIONS

Weizhen Mao
David M. Nicol

Contract NAS1-19480
October 1994

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

Operated by Universities Space Research Association

On k-ary n-cubes: Theory and Applications [1]

Weizhen Mao [2] *and* David M. Nicol [3]

Department of Computer Science
The College of William and Mary
Williamsburg, VA 23187-8795
{wm, nicol}@cs.wm.edu

Abstract

Many parallel processing networks can be viewed as graphs called k-ary n-cubes, whose special cases include rings, hypercubes and toruses. In this paper, combinatorial properties of k-ary n-cubes are explored. In particular, the problem of characterizing the subgraph of a given number of nodes with the maximum edge count is studied. These theoretical results are then used to compute a lower bounding function in branch-and-bound partitioning algorithms and to establish the optimality of some irregular partitions.

[1] An extended abstract of this paper (without any proofs and missing some theorems) has been submitted to the 1995 International Parallel Processing Symposium, with the permission of its Program Chair to simultaneously submit this full paper to an archival journal.

[2] This work was supported in part by NSF grant CCR-9210372.

[3] This work was supported in part by NASA under NAS1-19480 while the author was on sabbatical at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681. It was also supported in part by NSF grant CCR-9201195.

1 Introduction

In a *k-ary n-cube* $G_{k,n}$, each node is identified by an n-bit base-k address $b_{n-1} \ldots b_i \ldots b_0$, and for every *dimension* $i = 0, 1, \ldots, n-1$, it is connected by edges to nodes with addresses $b_{n-1} \ldots b_i \pm 1 (\mathrm{mod}\ k) \ldots b_0$.

We can also define $G_{k,n}$ recursively. First, we define a *ring* of k nodes $0, 1, \ldots, k-1$ to be a graph with edges between i and $i+1 (\mathrm{mod}\ k)$ for $i = 0, 1, \ldots, k-1$. When $k = 1$, a ring is a point. When $k = 2$, a ring is two nodes sharing an edge. When $k \geq 3$, a ring is a conventional ring. The recursive definition of $G_{k,n}$ is as follows.

- $G_{k,1}$ is a ring of k nodes. Without loss of generality, we place the k nodes on a line, and call the leftmost node the 0^{th} position node and the rightmost node the $(k-1)^{st}$ position node.

- $G_{k,n}$ contains k *composite subcubes* of type $G_{k,n-1}$ placed from left to right. For each position $i = 0, \ldots, k^{n-1} - 1$, edges between composite subcubes are defined by connecting all k i^{th} position nodes in a ring.

Further, $G_{k,n}$ can also be viewed as an n-dimensional (n-D) torus, which is a $\underbrace{k \times \cdots \times k}_{n}$ cube of grids with wrap-around edges.

The second and the third definitions of $G_{k,n}$ provide two ways of drawing $G_{k,n}$. See Figure 1 for an example.

Table 1 shows special cases of $G_{k,n}$. The first column contains the values of k, and the first row contains the values of n. We notice that the class $G_{k,n}$ contains many topologies important to parallel processing, such as rings, hypercubes and toruses; hence a thorough study of $G_{k,n}$ is worthwhile.

The following combinatorial properties of $G_{k,n}$ are easy to verify except perhaps the last one, for which we provide its proof in Appendix.

PROPERTY 1.1 *$G_{k,n}$ has k^n nodes.*

PROPERTY 1.2 *$G_{k,n}$ contains k composite subcubes of type $G_{k,n-1}$, and the number of edges with endpoints in different composite subcubes is k^{n-1} for $k = 2$ and k^n for $k \geq 3$.*

PROPERTY 1.3 *$G_{k,n}$ is a regular graph, meaning that each node has the same degree. The degree of each node is n for $k = 2$ and $2n$ for $k \geq 3$.*

PROPERTY 1.4 *The number of edges in $G_{k,n}$ is nk^{n-1} for $k = 2$ and nk^n for $k \geq 3$.*

PROPERTY 1.5 *In each i^{th} composite subcube ($0 \leq i \leq k-1$) of type $G_{k,n-1}$ in $G_{k,n}$, choose m_i nodes, and define $m = \sum_{i=0}^{k-1} m_i$. The number of edges with endpoints among these m nodes but in different composite subcubes is no larger than $\min\{m_0, m_1\}$ for $k = 2$, and is no larger than $m - \max_{0 \leq i \leq k-1}\{m_i\} + \min_{0 \leq i \leq k-1}\{m_i\}$ for $k \geq 3$.*

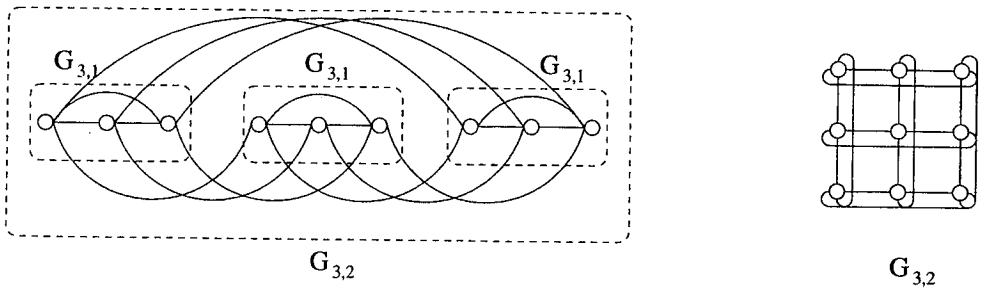


Figure 1: A 3-ary 2-cube $G_{3,2}$

	1	2	≥ 3
1	point (ring)	point (torus)	point
2	edge (hypercube/ring)	square (hypercube/torus)	hypercube
≥ 3	ring	torus	$G_{k,n}$

Table 1: Special cases of $G_{k,n}$

Properties of k-ary n-cubes related to VLSI concerns have been explored by Dally [4]. One property that is related to our study is the *bisection-width* of k-ary n-cubes, the minimum number of edges one must cut when partitioning the graph into two subgraphs with the equal numbers of nodes. Our work considers a generalization of this notion: given that the partition may contain P subgraphs, what is the minimum number of edges between the subgraphs?

The problem of partitioning graphs for parallel processing includes rigorous treatments in [8, 11], where algorithms are developed that partition graphs with guarantees on the load-imbalance and number of edges cut. Our work is similar in the sense of its rigor, restricted to k-ary n-cubes we give achievable lower bounds on partitioning costs.

We have previously studied properties of k-ary n-cubes in the context of load balancing [10]. Here graph nodes typically represent computation and edges represent communication. For any subgraph, define an *internal edge* to be one with two endpoints in the subgraph and an *external edge* to be one with one endpoint in the subgraph; viewing the subgraph as the set of nodes assigned to a processor, the number of external edges is a measure of the communication cost. Allowing nodes and edges to be weighted (reflecting relative computation and communication volumes, respectively), the "load" of a subgraph is taken to be the sum of the weights of its nodes and its external edges. If $G_{k,n}$ is partitioned into P subgraphs, the *bottleneck* cost of the partition is the maximum load among all partition subgraphs [2, 12]. The bottleneck cost reflects that of one phase of a data parallel computation where computation and communication are not overlapped, and a global synchronization occurs at the end of the phase. The communication that occurs is needed for the subsequent phase; there are no data dependencies among the computations performed in a given phase. In a previous paper [10] we showed that certain equi-partitions are optimal in the sense of

minimizing the bottleneck cost, but that, surprisingly, there exist cases where the optimal partition is *not* an equi-partition. These results are based on a lower bound on a processor's communication cost, a bound that is achieved for selected subgraph sizes. The current paper completes that work by identifying an achievable bound for general subgraph sizes.

The problem of identifying the minimal communication cost (assuming unit edge weight) of a subgraph of size m is the same as maximizing the number of internal edges in a subgraph with m nodes, since each node in $G_{k,n}$ has the same degree. That is, we study the following combinatorial problem.

> Consider any subgraph S_m of $m \leq k^n$ nodes in $G_{k,n}$. Let $e(S_m)$ be the number of internal edges in S_m. Define
>
> $$e_k(m,n) = \max_{\forall S_m}\{e(S_m)\}.$$
>
> For any $m = 1, 2, \ldots, k^n$, determine $e_k(m,n)$, the maximum number of internal edges in any subgraph S_m in a k-ary n-cube.

We will say that a subgraph of $G_{k,n}$ with m nodes is *optimal* if it has $e_k(m,n)$ internal edges.

The case $k = 1$ is trivial: $e_1(m,n) = 0$ for $m \leq 1^n = 1$. In Sections 2, 3, and 4, we will determine $e_2(m,n)$, $e_3(m,n)$, and $e_4(m,n)$, respectively. In Section 5, we will study the case $k \geq 5$. In Section 6, we present two applications of the results developed; one uses these results in the context of branch-and-bound algorithms for partitioning k-ary n-cubes with generally weighted nodes and edges. Finally, we summarize our contributions in Section 7. Appendix contains the proofs of Property 1.5 and lemmas contributing to the main results.

2 The case $k = 2$

To determine $e_2(m,n)$, the maximum number of internal edges of a subgraph with m nodes in a hypercube, we will have to do some preliminary work.

DEFINITION 2.1

$w(i)$ denotes the sum of all bits in the base-2 (binary) representation of i.
$W(i,j)$, $i \leq j$, denotes the sum of $w(i), \ldots, w(j)$.

The following three lemmas concern properties of function W. Their proofs can be found in Appendix.

LEMMA 2.1 *$W(i, 2i-1) = W(0, i-1) + i$ for $i \geq 1$.*

LEMMA 2.2 *$W(i+1, 2i) = W(0, i-1) + i$ for $i \geq 1$.*

LEMMA 2.3 *$W(j, j+i-1) \geq W(0, i-1) + i$ for $j \geq i \geq 1$.*

We next define a recursive function F and give its closed form in terms of W.

DEFINITION **2.2**

$F(0) = F(1) = 0$;

$F(m) = F(\lceil \frac{m}{2} \rceil) + F(\lfloor \frac{m}{2} \rfloor) + \lfloor \frac{m}{2} \rfloor$ *for* $m \geq 2$.

THEOREM **2.1** $F(m) = W(0, m-1)$ *for* $m \geq 1$.

Proof We induct on m. When $m = 1$, $F(1) = W(0,0) = 0$. Assume that the equation holds for $\leq m-1$. Now consider m.

Case 1. $m = 2i$ for some $i \geq 1$.

$$\begin{aligned} F(m) &= F(i) + F(i) + i \ \text{(Definition 2.2)} \\ &= W(0, i-1) + W(0, i-1) + i \ \text{(Inductive hypothesis)} \\ &= W(0, i-1) + W(i, 2i-1) \ \text{(Lemma 2.1)} \\ &= W(0, 2i-1) \\ &= W(0, m-1). \end{aligned}$$

Case 2. $m = 2i + 1$ for some $i \geq 1$.

$$\begin{aligned} F(m) &= F(i+1) + F(i) + i \ \text{(Definition 2.2)} \\ &= W(0, i) + W(0, i-1) + i \ \text{(Inductive hypothesis)} \\ &= W(0, i) + W(i+1, 2i) \ \text{(Lemma 2.2)} \\ &= W(0, 2i) \\ &= W(0, m-1). \blacksquare \end{aligned}$$

COROLLARY **2.1** $F(m) \geq F(m_0) + F(m_1) + \min\{m_0, m_1\}$ *for* $m_0 + m_1 = m$.

Proof If at least one of m_0 and m_1 is 0, the inequality holds trivially. Now assume that $m_0 \geq m_1 \geq 1$.

$$\begin{aligned} F(m) &= W(0, m-1) \ \text{(Theorem 2.1)} \\ &= W(0, m_0 - 1) + W(m_0, m-1) \\ &\geq W(0, m_0 - 1) + W(0, m_1 - 1) + m_1 \ \text{(Lemma 2.3)} \\ &= F(m_0) + F(m_1) + m_1 \ \text{(Theorem 2.1)}. \blacksquare \end{aligned}$$

COROLLARY **2.2** $F(m) = \frac{1}{2} m \log_2 m$ *if* $m = 2^l$ *for some* l.

Proof Use Definition 2.2 and inductive proof on m. ■

It turns out that $F(m)$ exactly captures the quantity of interest.

THEOREM **2.2** $e_2(m, n) = F(m)$ *for* $m \leq 2^n$.

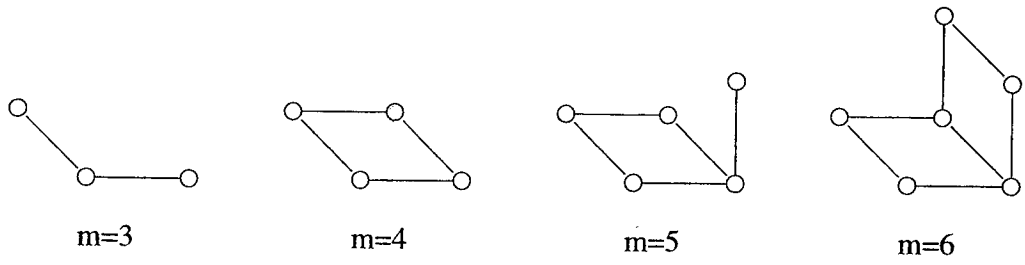


Figure 2: Subgraphs of $G_{2,n}$ achieving internal edge count $F(m)$

Proof Since $G_{2,n}$ contains two composite subcubes of type $G_{2,n-1}$, assume that m_0 and m_1 nodes are chosen in the 0^{th} and 1^{st} composite subcubes, respectively. By Property 1.5,

$$e_2(0,n) = e_2(1,n) = 0;$$
$$e_2(m,n) \leq \max_{\forall \sum m_i = m} \{e_2(m_0, n-1) + e_2(m_1, n-1) + \min\{m_0, m_1\}\}.$$

First we prove by induction on m that $e_2(m,n) \leq F(m)$. When $m = 0, 1$, $e_2(m,n) = F(m) = 0$. Assume that the inequality holds for $\leq m-1$. Now consider m.

$$\begin{aligned} e_2(m,n) &\leq \max_{\forall \sum m_i = m} \{e_2(m_0, n-1) + e_2(m_1, n-1) + \min\{m_0, m_1\}\} \\ &\leq \max_{\forall \sum m_i = m} \{F(m_0) + F(m_1) + \min\{m_0, m_1\}\} \text{ (Inductive hypothesis)} \\ &\leq F(m) \text{ (Corollary 2.1).} \end{aligned}$$

Next we prove that there is a subgraph S_m^* of m nodes such that the number of internal edges in S_m^* is $F(m)$. Here is how we can allocate the m nodes for S_m^*: Allocate $\lceil \frac{m}{2} \rceil$ nodes into the 0^{th} composite subcube and $\lfloor \frac{m}{2} \rfloor$ nodes into the 1^{st} composite subcube; use the same method recursively to allocate the nodes in each composite subcube. It is obvious that the number of internal edges in S_m^* is exactly $F(m)$. ■

This theorem tells us about the structure of a subgraph with exactly $F(m)$ internal edges—it is possible to bisect this subgraph "evenly" with exactly $\lfloor \frac{m}{2} \rfloor$ edges between the two pieces, which are themselves optimal with respect to their sizes. Figure 2 illustrates optimal subgraphs of $G_{2,n}$ for $m = 3, 4, 5, 6$.

3 The case $k = 3$

Similar to the previous section, to determine $e_3(m,n)$ for $G_{3,n}$ we will have to do some preliminary work.

DEFINITION 3.1

$z(i)$ denotes the sum of all bits in the base-3 representation of i.

$Z(i,j)$, $i \leq j$, denotes the sum of $z(i), \ldots, z(j)$.

The following five lemmas concern the properties of function Z. Their proofs can be found in Appendix.

LEMMA 3.1 $Z(0, 3i-1) = 3Z(0, i-1) + 3i$ *for* $i \geq 1$.

LEMMA 3.2 $Z(0, 3i) = Z(0, i) + 2Z(0, i-1) + 3i$ *for* $i \geq 1$.

LEMMA 3.3 $Z(0, 3i+1) = 2Z(0, i) + Z(0, i-1) + 3i + 1$ *for* $i \geq 1$.

LEMMA 3.4 $Z(j, j+i-1) \geq Z(0, i-1) + i$ *for* $j \geq i \geq 1$.

LEMMA 3.5 $Z(j, j+i_1+i_2-1) \geq Z(0, i_1-1) + Z(0, i_2-1) + i_1 + 2i_2$ *for* $j \geq i_1 \geq i_2 \geq 1$.

We next define a recursive function G and give its closed form in terms of Z.

DEFINITION 3.2
$G(0) = G(1) = 0$;
$G(m) = (m \bmod 3)G(\lceil \frac{m}{3} \rceil) + (3 - m \bmod 3)G(\lfloor \frac{m}{3} \rfloor) + m - \lceil \frac{m}{3} \rceil + \lfloor \frac{m}{3} \rfloor$ *for* $m \geq 2$.

THEOREM 3.1 $G(m) = Z(0, m-1)$ *for* $m \geq 1$.

Proof Similar to the proof of Theorem 2.1. In the inductive step, we consider three cases: $m = 3i$, $m = 3i+1$, and $m = 3i+2$, and use Lemmas 3.1, 3.2, and 3.3 in the three cases, respectively. ■

COROLLARY 3.1 $G(m) \geq G(m_0)+G(m_1)+G(m_2)+m-\max\{m_0, m_1, m_2\}+\min\{m_0, m_1, m_2\}$ *for* $m_0 + m_1 + m_2 = m$.

Proof If at least two of m_0, m_1 and m_2 are 0, the inequality holds trivially. If only one, say m_2, is 0, assuming that $m_0 \geq m_1 \geq 1$, the derivation is almost identical to the same case in the proof of Corollary 2.1 except here we use G instead of F and Z instead of W. If none of m_0, m_1 and m_2 is 0, assuming that $m_0 \geq m_1 \geq m_2 \geq 1$, we have

$$\begin{aligned} G(m) &= Z(0, m-1) \text{ (Theorem 3.1)} \\ &= Z(0, m_0 - 1) + Z(m_0, m-1) \\ &\geq Z(0, m_0-1) + Z(0, m_1-1) + Z(0, m_2-1) + m_1 + 2m_2 \text{ (Lemma 3.5)} \\ &= G(m_0) + G(m_1) + G(m_2) + m - m_0 + m_2 \text{ (Theorem 3.1).} \blacksquare \end{aligned}$$

COROLLARY 3.2 $G(m) = m \log_3 m$ *if* $m = 3^l$ *for some* l.

Proof Use Definition 3.2 and inductive proof on m. ■

It turns out that $G(m)$ exactly captures the quantity of interest.

THEOREM 3.2 $e_3(m, n) = G(m)$ *for* $m \leq 3^n$.

Proof Since $G_{3,n}$ contains three composite subcubes of type $G_{3,n-1}$, assume that m_0, m_1 and m_2 nodes are chosen in the 0^{th}, 1^{st} and 2^{nd} composite subcubes, respectively. By Property 1.5,

$$\begin{aligned} e_3(0,n) &= e_3(1,n) = 0 \\ e_3(m,n) &\leq \max_{\forall \sum m_i = m} \{e_3(m_0,n-1) + e_3(m_1,n-1) + e_3(m_2,n-1) \\ &\qquad + m - \max\{m_0,m_1,m_2\} + \min\{m_0,m_1,m_2\}\}. \end{aligned}$$

Similar to Theorem 2.2, we can prove by induction on m that $e_3(m,n) \leq G(m)$, using the above recursive definition of $e_3(m,n)$, inductive hypothesis, and Corollary 3.1.

Also similar to Theorem 2.2, a subgraph S_m^* of m nodes with $G(m)$ internal edges can be constructed by allocating $\lceil \frac{m}{3} \rceil$ nodes into each of the first m mod 3 composite subcubes and $\lfloor \frac{m}{3} \rfloor$ nodes into each of the remaining composite subcubes; the same method is then used recursively to allocate the nodes in each composite subcube. ∎

4 The case $k = 4$

Similar to the previous two sections, to determine $e_4(m,n)$ for $G_{4,n}$ we will have to do some preliminary work. The following four lemmas concern additional properties of function W. Their proofs can be found in Appendix.

LEMMA **4.1** $W(0,4i-1) = 4W(0,i-1) + 4i$ *for* $i \geq 1$.

LEMMA **4.2** $W(0,4i) = W(0,i) + 3W(0,i-1) + 4i$ *for* $i \geq 1$.

LEMMA **4.3** $W(0,4i+1) = 2W(0,i) + 2W(0,i-1) + 4i + 1$ *for* $i \geq 1$.

LEMMA **4.4** $W(0,4i+2) = 3W(0,i) + W(0,i-1) + 4i + 2$ *for* $i \geq 1$.

We next define a recursive function H and show that it is the same function as F defined in Section 2.

DEFINITION **4.1**

$H(0) = H(1) = 0$;

$H(m) = (m \bmod 4)H(\lceil \frac{m}{4} \rceil) + (4 - m \bmod 4)H(\lfloor \frac{m}{4} \rfloor) + m - \lceil \frac{m}{4} \rceil + \lfloor \frac{m}{4} \rfloor$ *for* $m \geq 2$.

THEOREM **4.1** $H(m) = W(0,m-1)$ *for* $m \geq 1$.

Proof Similar to the proof of Theorem 2.1. In the inductive step, we consider four cases: $m = 4i$, $m = 4i+1$, $m = 4i+2$, and $m = 4i+3$, and use Lemmas 4.1, 4.2, 4.3, and 4.4 in the four cases, respectively. ∎

COROLLARY **4.1** $H(m) \geq H(m_0) + H(m_1) + H(m_2) + H(m_3) + m - \max\{m_0,m_1,m_2,m_3\} + \min\{m_0,m_1,m_2,m_3\}$ *for* $m_0 + m_1 + m_2 + m_3 = m$.

Proof If at least three of m_0, m_1, m_2 and m_3 are 0, the inequality holds trivially. If only two, say m_2 and m_3, are 0, assuming that $m_0 \geq m_1 \geq 1$, the derivation is almost identical to the same case in the proof of Corollary 2.1 except here we use H instead of F. If at most one of m_0, m_1, m_2 and m_3 is 0, assuming that $m_0 \geq m_1 \geq m_2 \geq m_3$, we have

$$\begin{aligned} H(m) &= F(m) \text{ (Theorems 2.1 and 4.1)} \\ &\geq F(m_0 + m_1) + F(m_2 + m_3) + m_2 + m_3 \text{ (Corollary 2.1)} \\ &\geq F(m_0) + F(m_1) + F(m_2) + F(m_3) + m_1 + m_2 + 2m_3 \text{ (Corollary 2.1)} \\ &= H(m_0) + H(m_1) + H(m_2) + H(m_3) + m_1 + m_2 + 2m_3 \text{ (Theorem 4.1).} \blacksquare \end{aligned}$$

COROLLARY **4.2** *$H(m) = m \log_4 m$ if $m = 4^l$ for some l.*

Proof Use Definition 4.1 and inductive proof on m. ■

THEOREM **4.2** *$e_4(m, n) = H(m)$ for $m \leq 4^n$.*

Proof Since $G_{4,n}$ contains four composite subcubes of type $G_{4,n-1}$, assume that m_0, m_1, m_2 and m_3 nodes are chosen in the 0^{th}, 1^{st}, 2^{nd} and 3^{rd} composite subcubes, respectively. By Property 1.5,

$$\begin{aligned} & e_4(0, n) = e_4(1, n) = 0 \\ & e_4(m, n) \leq \max_{\forall \sum m_i = m} \{e_4(m_0, n-1) + e_4(m_1, n-1) + e_4(m_2, n-1) + e_4(m_3, n-1) \\ & \qquad + m - \max\{m_0, m_1, m_2, m_3\} + \min\{m_0, m_1, m_2, m_3\}\}. \end{aligned}$$

Similar to Theorem 2.2, we can prove by induction on m that $e_4(m, n) \leq H(m)$, using the above recursive definition of $e_4(m, n)$, inductive hypothesis, and Corollary 4.1.

Also similar to Theorem 2.2, a subgraph S_m^* of m nodes with $H(m)$ internal edges can be constructed by allocating $\lceil \frac{m}{4} \rceil$ nodes into each of the first m mod 4 composite subcubes and $\lfloor \frac{m}{4} \rfloor$ nodes into each of the remaining composite subcubes; the same method is then used recursively to allocate the nodes in each composite subcube. ■

5 The case $k \geq 5$

Given that essentially the same approach defines the structure of optimal subgraphs for three successive values of k, one might suspect a general pattern for all k. It turns out that this is *not* the case and that for $k \geq 5$ the decomposition that once defined optimal subgraphs now defines suboptimal ones. Consider the example of $k = 5$, $m = 6$. If we partition in one dimension into one subgraph of two nodes and four subgraphs of one node each we achieve six internal edges (a ring of five nodes, with one extra node hanging off the ring). However, it is possible to embed the six node graph illustrated in Figure 2 into $G_{5,n}$, and achieve seven internal edges. An ability to embed subgraphs of $G_{2,n}$ into $G_{k,n}$ turns out to be what is needed to characterize the optimal subgraphs of $G_{k,n}$ with m nodes, when $k \geq 5$ and $m \leq 2^n$. To prove this, we need the following theorem.

THEOREM **5.1** *$F(m) \geq \sum_{i=0}^{k-1} F(m_i)+m-\max_{0\leq i\leq k-1}\{m_i\}+\min_{0\leq i\leq k-1}\{m_i\}$ for $\sum_{i=0}^{k-1} m_i = m$.*

Proof Assume that $m_0 \geq m_1 \geq \cdots \geq m_{k-1} \geq 0$. Let l be the smallest index such that $\sum_{i=0}^{l} m_i \geq \frac{m}{2}$. Clearly, $\sum_{i=0}^{l-1} m_i < \frac{m}{2}$ and $\sum_{i=l}^{k-1} m_i > \frac{m}{2}$. This also implies that $l < k-l$. So $l < \frac{k}{2}$.

$$\begin{aligned} F(m) &\geq F(\sum_{i=0}^{l} m_i) + F(\sum_{i=l+1}^{k-1} m_i) + \min\{\sum_{i=0}^{l} m_i, \sum_{i=l+1}^{k-1} m_i\} \text{ (Corollary 2.1)} \\ &\geq \sum_{i=0}^{k-1} F(m_i) + A + B + C \text{ (Corollary 2.1 repeatedly)}, \end{aligned}$$

where

$$A = \min\{\sum_{i=0}^{l} m_i, \sum_{i=l+1}^{k-1} m_i\},$$

$$B = \sum_{i=0}^{l-1} \min\{m_i, \sum_{j=i+1}^{l} m_i\},$$

and

$$C = \sum_{i=l+1}^{k-2} \min\{m_i, \sum_{j=i+1}^{k-1} m_i\}.$$

Next, we wish to prove that $A+B+C \geq m-m_0+m_{k-1}$. Since $\sum_{i=0}^{l} m_i \geq \frac{m}{2}$, $A = \sum_{i=l+1}^{k-1} m_i$. Since $l < \frac{k}{2}$ and $k \geq 5$, $l+1 \leq k-2$. So there is at least one term in C. Therefore, $C \geq m_{k-1}$. How large is B? If $l = 0$, then $B = 0$ and $A+B+C \geq \sum_{i=1}^{k-1} m_i + m_{k-1} = m-m_0+m_{k-1}$. If $l = 1$, then $B = m_1$ and $A+B+C \geq \sum_{i=2}^{k-1} m_i + m_1 + m_{k-1} = m-m_0+m_{k-1}$. Now assume that $l \geq 2$. B must have at least two terms. If $m_h \leq \sum_{i=h+1}^{l} m_i$ for all $h = 0, \ldots, l-2$, then $B = \sum_{i=0}^{l-2} m_i + m_l$ and $A+B+C \geq \sum_{i=l+1}^{k-1} m_i + \sum_{i=0}^{l-2} m_i + m_l + m_{k-1} \geq m-m_0+m_{k-1}$. If there is h in $[0, l-2]$ such that $m_h > \sum_{i=h+1}^{l} m_i$ (choose the smallest h if there is more than one), then $B \geq \sum_{i=0}^{h-1} m_i + \sum_{i=h+1}^{l} m_i$ and $A+B+C \geq \sum_{i=l+1}^{k-1} m_i + \sum_{i=0}^{h-1} m_i + \sum_{i=h+1}^{l} m_i + m_{k-1} \geq m-m_0+m_{k-1}$. ∎

THEOREM **5.2** *$e_k(m,n) = F(m)$ for $m \leq 2^n$ and $k \geq 5$.*

Proof Since $G_{k,n}$ contains k composite subcubes of type $G_{k,n-1}$, assume that m_i nodes are chosen in the i^{th} composite subcube for $0 \leq i \leq k-1$. By Property 1.5,

$$e_k(0,n) = e_k(1,n) = 0;$$

$$e_k(m,n) \leq \max_{\forall \sum m_i = m} \{\sum_{i=0}^{k-1} e_k(m_i, n-1) + m - \max_{0\leq i\leq k-1}\{m_i\} + \min_{0\leq i\leq k-1}\{m_i\}\}.$$

Similar to Theorem 2.2, we can prove by induction on m that $e_k(m,n) \leq F(m)$, using the above recursive definition of $e_k(m,n)$, inductive hypothesis, and Theorem 5.1.

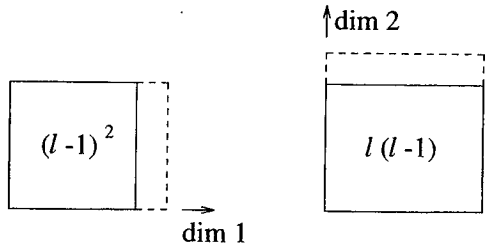


Figure 3: Construction procedure for $C_2(m)$

Also similar to Theorem 2.2, a subgraph S_m^* of $m \leq 2^n$ nodes with $F(m)$ internal edges can be constructed by allocating $\lceil \frac{m}{2} \rceil$ nodes into the 0^{th} composite subcube and $\lfloor \frac{m}{2} \rfloor$ nodes into the 1^{st} composite subcube; the same method is then used recursively to allocate the nodes in each composite subcube. ■

What then of subgraphs of size $m > 2^n$? For this case we assume that either k is so large relative to m that an optimal subgraph cannot include wrap-around edges, or that the graph of interest is a mesh (without wrap-around edges) whose local structure is like that of $G_{k,n}$. In other words, we now also consider multi-dimensional rectangular meshes, structures we will call n-D meshes. Intuition tells us that the maximum number of internal edges $e_k(m,n)$ can be reached when the m nodes are placed as tightly as possible to form a "cubish" polyhedron. In the remainder of this section, we shall prove that our intuition turns out to be correct.

In any dimension i, a subgraph of m nodes can be partitioned into *layers*, each of which contains nodes with the same coordinate in dimension i. Furthermore, there may be edges (legs) between adjacent layers. We give the following definition of a *cubish polyhedron.*

DEFINITION **5.1**

For any $m \geq 2$, there exist $l \geq 2$ and $1 \leq i \leq n$ such that $l^{i-1}(l-1)^{n-i+1} < m \leq l^i(l-1)^{n-i}$. Let $\delta = m - l^{i-1}(l-1)^{n-i+1}$. The n-D cubish polyhedron of m nodes in $G_{k,n}$, denoted as $C_n(m)$, is defined recursively as follows.

- *$C_1(m)$ is a line of m nodes.*
- *To construct $C_n(m)$, we start with an $\underbrace{l \times \cdots \times l}_{i-1} \times \underbrace{(l-1) \times \cdots \times (l-1)}_{n-i+1}$ n-D mesh. For the remaining δ nodes, we construct an $(n-1)$-D layer $C_{n-1}(\delta)$ and add it on the top of the n-D mesh in dimension i.*

The above procedure of constructing $C_n(m)$ is very much like making a ball of yarn. The idea is to fill in each side (dimension) with yarn (nodes), one side (dimension) at a time. Figure 3 illustrates the construction procedure for $C_2(m)$, and Figure 4 illustrates the procedure for $C_3(m)$. Let $e_n(m)$ be the internal edge count in a cubish polyhedron $C_n(m)$. Obviously, $e_n(m) = [e_{n-1}(\delta) + \delta] + e_n(m - \delta)$.

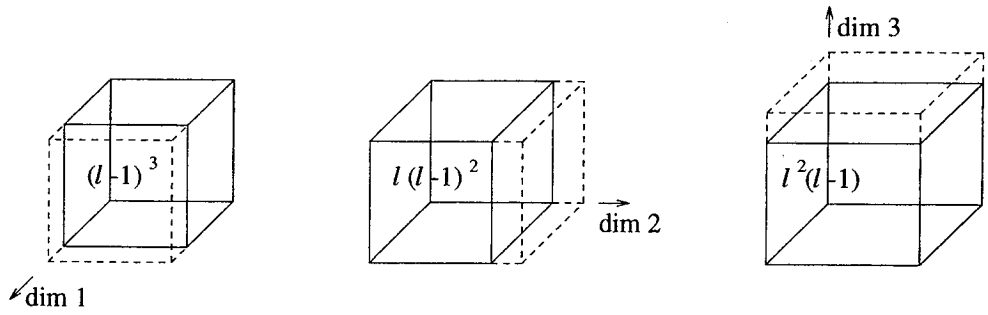


Figure 4: Construction procedure for $\mathcal{C}_3(m)$

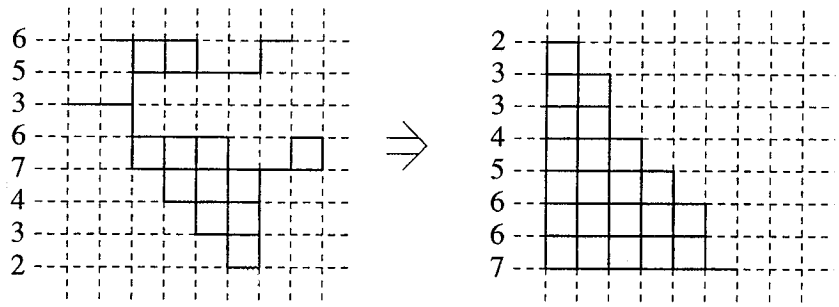


Figure 5: Rearrange S_m (in $G_{k,2}$) without decreasing $e(S_m)$

THEOREM **5.3** *$\mathcal{C}_n(m)$ has the maximum internal edge count among all subgraphs S_m of m nodes in $G_{k,n}$ (or in n-D meshes), when the warp-around edges can be discounted.*

Proof We prove by induction on n. When $n = 1$, the claim is trivially true. Assume that the claim holds true for $n-1$. Now consider the case of n. Let S_m be any subgraph of m nodes with $e(S_m)$ internal edges in $G_{k,n}$. We wish to prove that $e(S_m) \leq e_n(m)$.

We can view S_m as having several $(n-1)$-D layers of nodes stacked on each other in a certain dimension. Rearrange the order of the layers by sizes (node counts) and within each layer rearrange the nodes into an $(n-1)$-D cubish polyhedron. See Figure 5 for an example (The numbers in the figure are the sizes of the layers). If after this rearrangement there are h layers and s_i is the size of the i^{th} layer with $s_1 \leq s_2 \leq \cdots \leq s_h$, then by the inductive hypothesis we have

$$e(S_m) \leq [e_{n-1}(s_1) + s_1] + [e_{n-1}(s_2) + s_2] + \cdots + [e_{n-1}(s_{h-1}) + s_{h-1}] + e_{n-1}(s_h).$$

Note that $s_1 + s_2 + \cdots + s_{h-1}$ is the number of edges (legs) between adjacent layers.

We have a few observations about the new subgraph obtained. First, layers in each dimension (not just the dimension chosen in the rearrangement) are stacked on each other by sizes. Second, $h \geq l$. Assume that $h \leq l-1$ for all dimensions. We must have $m \leq (l-1)^n$, which is impossible. Third, $s_1 \leq l^{i-1}(l-1)^{n-i}$. Suppose not. We must have $m = s_1 + \cdots + s_h \geq hs_1 \geq ls_1 > l^i(l-1)^{n-i}$, which is impossible.

Let us go back to the induction step, in which we assume that $e_{n-1}(m)$ is maximum and wish to prove that $e_n(m)$ is maximum. We need another induction on m to prove this. When $m = 1, 2$, $e_n(m)$ is obviously maximum. Assume that $e_n(j)$ is maximum for $j \leq m-1$. Now consider the case $j = m$. We know by the inductive hypothesis that

$$e(S_m) \leq [e_{n-1}(s_1) + s_1] + e_n(m - s_1).$$

By Definition 5.1, we know that $\mathcal{C}_n(m-\delta)$ is in fact an n-D mesh with $l^{i-1}(l-1)^{n-i+1}$ nodes. $\mathcal{C}_n(m-\delta)$ can also be viewed as having l (or $l-1$ if $i=1$) layers stacked on each other, where each layer is an $(n-1)$-D mesh and has L nodes. Clearly,

$$L = \begin{cases} (l-1)^{n-1} & \text{if } i = 1; \\ l^{i-2}(l-1)^{n-i+1} & \text{if } i \geq 2. \end{cases}$$

We can show that $s_1 < L + \delta$. Suppose not. We must have $m \geq hs_1 \geq ls_1 \geq lL + l\delta > lL + \delta \geq m$, which is impossible. To continue, we consider two cases.

Case 1. $s_1 \leq \delta$. We must have $l^{i-1}(l-1)^{n-i+1} \leq m - s_1 < l^i(l-1)^{n-i}$. Let $m - s_1 = l^{i-1}(l-1)^{n-i+1} + \delta'$. Then $s_1 + \delta' = \delta$. So

$$e_n(m - s_1) = [e_{n-1}(\delta') + \delta'] + e_n(l^{i-1}(l-1)^{n-i+1})$$

and

$$e_{n-1}(s_1) + e_{n-1}(\delta') \leq e_{n-1}(\delta).$$

Therefore,

$$\begin{aligned} e(S_m) &\leq [e_{n-1}(s_1) + s_1] + e_n(m - s_1) \\ &= [e_{n-1}(s_1) + s_1] + [e_{n-1}(\delta') + \delta'] + e_n(l^{i-1}(l-1)^{n-i+1}) \\ &\leq [e_{n-1}(\delta) + \delta] + e_n(l^{i-1}(l-1)^{n-i+1}) \\ &= e_n(m). \end{aligned}$$

Case 2. $s_1 > \delta$. Since $s_1 < L + \delta$, we must have $(l'-1)L < m - s_1 < l'L$, where $l' = l-1$ if $i = 1$ and $l' = l$ if $i \geq 2$. Let $m - s_1 = (l'-1)L + \delta'$, where $\delta' < L$. Then $s_1 + \delta' = L + \delta$. So

$$e_n(m - s_1) = [e_{n-1}(\delta') + \delta'] + e_n((l'-1)L).$$

Therefore,

$$\begin{aligned} e(S_m) &\leq [e_{n-1}(s_1) + s_1] + e_n(m - s_1) \\ &= [e_{n-1}(s_1) + s_1] + [e_{n-1}(\delta') + \delta'] + e_n((l'-1)L). \end{aligned}$$

On the other hand, we have

$$e_n(m) = [e_{n-1}(\delta) + \delta] + [e_{n-1}(L) + L] + e_n((l'-1)L).$$

To show that $e(S_m) \leq e_n(m)$, all we need to prove is that for $s_1 + \delta' = L + \delta$,

$$e_{n-1}(s_1) + e_{n-1}(\delta') \leq e_{n-1}(L) + e_{n-1}(\delta).$$

The inequality is trivially true when $s_1 = L$. Let us consider the following subcases.

Subcase 2.1. $s_1 < L$. We will prove by yet another induction on dimension $n-1$ that $e_{n-1}(s_1)+e_{n-1}(\delta') \leq e_{n-1}(L)+e_{n-1}(\delta)$, where $s_1, \delta' < L$ and $s_1+\delta' = L+\delta$. When $n-1 = 1$, it is a trivial case. Assume that the inequality holds for dimension $n-2$. Now consider the case of $n-1$. Without loss of generality, assume that $s_1 \geq \delta'$ (The case $s_1 < \delta'$ is symmetric). Initialize A and B to be $\mathcal{C}_{n-1}(s_1)$ and $\mathcal{C}_{n-1}(\delta')$, respectively. The node count in A, denoted as $|A|$, is then s_1, and $|B|$ is δ'. Consider A as a cubish polyhedron of several $(n-2)$-D layers of size L' each plus one more layer of $a \leq L'$ nodes and a legs on the top, and B as a cubish polyhedron of several $(n-2)$-D layers of size L'' each plus one more layer of $b \leq L''$ nodes and b legs on the top. Since $|A| \geq |B|$, A completely includes B. So $L' \geq L''$. We next apply the following step to move nodes from B to A. If there is a layer in B with size no greater than $L - |A|$, move the layer together with its legs to A and rearrange two polyhedrons into cubish polyhedrons again (Note that after the move A, B, L', L'', a, and b are updated). It is clear that this step does not decrease the total edge count in the two polyhedrons. Apply the above step until for any layer in B its size is larger than $L - |A|$. We must have $L - |A| < L'$ and $a + b > L'$. Since $a, b \leq L'$, by the inductive hypothesis,

$$e_{n-2}(a) + e_{n-2}(b) \leq e_{n-2}(L') + e_{n-2}(a + b - L').$$

Removing the top layer of a nodes and the top layer of b nodes from A and B, respectively, and adding a layer of L' nodes and a layer of $a + b - L'$ nodes to A and B, respectively, we get $|A| = L$ and $|B| = \delta$. So

$$e_{n-1}(s_1) + e_{n-1}(\delta') \leq e_{n-1}(L) + e_{n-1}(\delta).$$

Subcase 2.2. $s_1 > L$. Assume that $s_1 = L + g$, then $\delta = \delta' + g$. We have

$$e_{n-1}(s_1) = [e_{n-2}(g) + g] + e_{n-1}(L).$$

We can show that $\delta' \geq (l-1)g$. Suppose not. We must have $m = (l'-1)L + \delta' + s_1 < (l'-1)L + (l-1)g + L + g \leq l(L+g) = ls_1 \leq m$, which is impossible. We know that $\delta' < L$. If all dimensions in $\mathcal{C}_{n-1}(\delta')$ have at least l layers, then $\delta' > l^{n-2}(l-1) \geq L$, which is impossible. So there must be a dimension in $\mathcal{C}_{n-1}(\delta')$ which has fewer than l layers. Since $\delta' \geq (l-1)g$, there must be a layer with at least g nodes. So we can move the layer of g nodes together with its legs from $\mathcal{C}_{n-1}(s_1)$ to $\mathcal{C}_{n-1}(\delta')$ safely and get

$$[e_{n-2}(g) + g] + e_{n-1}(\delta') \leq e_{n-1}(\delta' + g).$$

Therefore,

$$\begin{aligned} e_{n-1}(s_1) + e_{n-1}(\delta') &= e_{n-1}(L) + [e_{n-2}(g) + g] + e_{n-1}(\delta') \\ &\leq e_{n-1}(L) + e_{n-1}(\delta' + g) \\ &= e_{n-1}(L) + e_{n-1}(\delta). \blacksquare \end{aligned}$$

6 Applications to Partitioning

The results so far, besides having theoretical interest, have practical applications to partitioning. There are different ways in which k-ary n-cubes are appropriate descriptions of parallel computations. One way is when at the lowest level the communication pattern of the computation is that of a k-ary n-cube, e.g., some mesh-oriented computation with periodic boundary conditions. Another is when the communication patterns reflect a k-ary n-cube because the computation is *about* a k-ary n-cube. For instance, the computation may be a direct-execution simulation of an application running on an architecture whose communication network is $G_{k,n}$ [1, 5]. We partition the simulation in order to balance the simulation workload and minimize communication overheads. Another instance is when the computation is written as though it executes on all nodes of an k-ary n-cube architecture, but the program is to be "folded" onto fewer processors, with subgraphs defined by the folding reflecting a set of tasks that are multi-tasked on one node of an actual machine [7].

To illustrate these points we show how our results may be used in the context of branch-and-bound algorithms for partitioning. Our object here is not to propose the specifics of such an algorithm nor study its performance. The ability to construct lower bounds on communication costs based only on subgraph node size is one that can be used in a variety of branch-and-bound formulations, and for a variety of partitioning problem formulations. We will illustrate its use in one specific case.

The results can also be used to show the optimality of some curiously shaped partitions, an example of this application is shown.

6.1 Lower Bounding in Branch-and-Bound

Consider a data parallel computation whose communication structure can be viewed as a k-ary n-cube, or related structure. The nodes of the graph are weighted individually to reflect computation costs, the edges of the graph are also weighted to reflect communication costs. It is assumed that communication between co-resident nodes is free, alternatively, with minor modifications one could model such internal communication with smaller—but nonzero—costs. We wish to find a *rectilinear* partitioning [9] of the graph into P subgraphs such that the bottleneck cost (the maximum, among all subgraphs, sum of the total node weights and the total external edge weights of any subgraph) is minimized. A rectilinear partition is one in which the separating cuts are all hyperplanes of the form $x_i = c_{ij}$, a constant. A rectilinear partition of an 8×8 mesh is illustrated in Figure 6. Rectilinear partitions preserve the nearest-neighbor communication structure of mesh-like communication patterns, as well as having other desirable properties [9].

Our earlier work on rectilinear partitioning established that for dimensions larger than two, the problem of finding an optimal partition is intractable. Furthermore, that work did not explicitly include communication costs. The results in this paper can be used in branch-and-bound algorithms [3] for finding rectilinear partitions, as we now show.

A node in the branch-and-bound search tree reflects a set of cuts already made, the initial

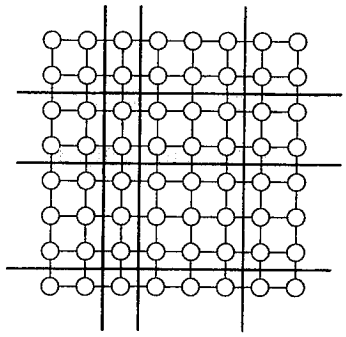


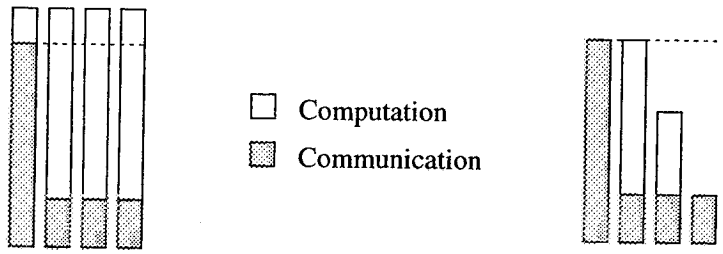
Figure 6: Rectilinear partition of an 8×8 mesh

node is empty. The children of a node reflect various ways of choosing one additional cut. If there are c cuts to be made, the search tree has depth $c + 1$. Every solution is a leaf of the search tree. We assume that the relative positioning of the cut associated with a level is known a priori, e.g., the cut in the third dimension whose cut coordinate is fifth smallest. Selecting the cut order is part of the branch-and-bound solution, but our focus here is on the lower bounding function needed for the branch-and-bound approach.

For every node N in the search tree we associate a function $bnd(N)$, that provides a lower bound on the bottleneck cost of any solution rooted at that node. $bnd(N)$ can be used to direct the search in different ways, e.g., in choosing the next node to explore or in pruning the search beyond that node because a known solution is better than any solution rooted at N. We are interested in defining an easily computed function $bnd(N)$. Each node N reflects the partitioning of the graph into some number of regions; furthermore, under our assumptions we know how many further divisions will be applied to each region. Consider a region R, to be further divided into s subregions, suppose that the number of nodes in region R is r, that the sum of all node weights in R is W_R, and that the edge weights of all edges with at least one node in R are sorted in list E in non-decreasing order.

We wish to construct a lower bound $lb(R)$ on the minimal bottleneck cost due to any possible subdivision of R into s subregions. The method we use relies on an ability to compute sizes of subregions $m_1, m_2, \ldots, m_s$, $m_i \geq 1$ for all i, and $\sum_{i=1}^{s} m_i = r$, such that $\sum_{i=1}^{s} C(m_i)$ is minimized, where $C(m_i)$ is the cost (external edge count) of an optimal subgraph with m_i nodes. Note that since all nodes in a k-ary n-cube have the same degree d, which is n for $k = 2$ and $2n$ for $k \geq 3$, we have that $C(m_i) = dm_i - 2e_k(m_i, n)$. Solution to this minimization problem—even when modified to include a constraint $m_i \leq B$ for all i, is straightforward using dynamic programming.

The bound construction of $lb(R)$ has three phases. First, we compute the vector $\mathbf{m} = (m_1, \ldots, m_s)$ that minimizes $\sum_{i=1}^{s} C(m_i)$; this reflects an idealized assignment of numbers of graph nodes to processors in such a way that the total number of edges cut (summed over all processors) is minimized. Second, we compute a vector $\mathbf{w}$ whose i^{th} component (w_i) is the sum of the weights of the first $C(m_i)$ edges in E. $\mathbf{w}$ reflects lower bounds on communication costs under assignment $\mathbf{m}$. Without loss of generality suppose that w_1 is the



(a) Slack is less than total computation

(b) Slack exceeds total computation

Figure 7: Computation of lower bound on bottleneck cost

largest component. We define the *slack* of $\mathbf{w}$ as

$$\text{slack}(\mathbf{w}) = \sum_{i=2}^{s}(w_1 - w_i).$$

Third, we consider the following two cases.

The first case of interest is when slack($\mathbf{w}$)$\leq W_R$. This means that if we treat the total computational workload W_R as divisible into arbitrary pieces, we can give each processor except the first enough workload to bring its total cost up to w_1, and still have workload remaining. The remanent may be divided evenly among the s processors. This is illustrated in Figure 7(a). So

$$lb(R) = \frac{W_R + \sum_{i=1}^{s} w_i}{s}.$$

The correctness of the bound is evident by the fact that the total load (sum of computation and communication) is minimized, and that no processor is ever idle.

The second case occurs when slack($\mathbf{w}$)$> W_R$, as illustrated by Figure 7(b). In this case the bottleneck is entirely communication induced, and the maximum number of nodes assigned to a processor must be driven down. This may increase the total communication cost, but will decrease the bottleneck cost. To reduce the bottleneck cost we constrain the assignment $m_i \leq B$ for all i; for each B considered we may compute the slack of the corresponding weight vector, and determine whether it exceeds W_R. Using a binary search on B we may find the least value B^* such that the corresponding slack exceeds W_R. Let $\mathbf{w} = (w_1, \ldots, w_s)$ and $\mathbf{w}' = (w'_1, \ldots, w'_s)$ be the weight vectors derived from using $B^* - 1$ and B^* as constraints, respectively. Then we make the lower bound to be

$$lb(R) = \min\{\frac{W_R + \sum_{i=1}^{s} w_i}{s}, w'_1\}.$$

We need not consider any bottleneck derived from using $B > B^*$, since the bottleneck cost is monotone non-decreasing in $\max\{m_i\}$, which is monotone non-decreasing in B. We need not consider any bottleneck derived from using $B < B^* - 1$, since in this case no processor is idle, and the total communication cost is at least as large as that derived from using $B^* - 1$.

Clearly the solution of dynamic programming equations is the most expensive part of this bound construction. It may be avoided by using lower bounds on external edge count

function C that have concave closed form expression. Such bounds have been developed in [10]:

$$B_k(m_i, n) = \begin{cases} m_i n - m_i \log m_i & \text{for } k = 2,\ m_i \leq 2^n; \\ 2m_i n - m_i \log m_i & \text{for } k > 2,\ m_i \leq 2^n; \\ 2m_i n - n(m_i - m_i^{(n-1)/n}) & \text{for } k > 2,\ m_i > 2^n. \end{cases}$$

Since $B_k(m_i, n)$ is concave in m_i, the theory of majorization [6] tells us that to minimize $\sum_{i=1}^{s} B_k(m_i, n)$ subject to $1 \leq m_i \leq B$ and $\sum_{i=1}^{s} m_i = r$ we assign $m_i = B$ for $i = 1, 2, \ldots, \lfloor (r-s)/(B-1) \rfloor$, with $m_{\lfloor (r-s)/(B-1) \rfloor + 1} = (r-s) \bmod (B-1) + 1$, and $m_i = 1$ for the remainder.

The procedure above shows how to bound from below the potential least bottleneck cost for each region reflected by node N. Applying this method to each such region, we define $bnd(N)$ as the greatest of these lower bounds, i.e.,

$$bnd(N) = \max_{\forall R \in N} \{lb(R)\}.$$

It should be noted that for a given number of processors P, and a given total workload W_R, the assignment problem whose minimized bottleneck cost is least is not necessarily one where the workload is spread evenly. For instance, consider an 8×8 torus to be partitioned into two regions. If each node has weight 4 and each edge has weight 1, then the optimal solution is to bisect the graph into two equal pieces, at a cost of $4 \times 32 + 8 = 136$. However, the graph that weights one node by 128 and all other nodes by 128/63 is optimally partitioned by isolating the heavy node, at a cost of $128 + 4 = 132$. Realization that minimized bottleneck costs need not be associated with evenly spread workload (and equi-partitions) leads us to the careful construction of $bnd(N)$ given.

6.2 Identification of Optimal Partitions

Another application of our results is to identify optimal partitions (with respect to the bottleneck metric), even when those partitions are not entirely regular. Consider the problem of partitioning $G_{8,2}$ (an 8×8 torus) into 13 subgraphs, assuming that all nodes have common computation weight w and all edges have unit communication cost. The problem clearly does not divide evenly. The minimal cost to a processor of having m nodes is $wm + C(m)$, where $C(m)$, the external edge count of the optimal subgraph with m nodes, is $4m - 2e_8(m, 2)$; note that the cost function increases monotonically in m.

The processor with the most nodes assigned will have at least $\lceil 64/13 \rceil = 5$ nodes. The optimal subgraph of $G_{8,2}$ with 5 nodes is a square, with an attached singleton node. As illustrated in Figure 8, it is possible to nearly tessellate $G_{8,2}$ with this optimal subgraph, the only exception being one subgraph (the center square) which is a subgraph itself of the optimal subgraph. The optimality of this partition derives from the fact that $wm + C(m)$ is monotone non-decreasing in m, so that the bottleneck cost $\max\{wm_1 + C(m_1), \ldots, wm_{13} + C(m_{13})\}$ is minimized when the m_i's are nearly equal. The partition shown achieves the lower bound of $5w + C(5) = 5w + 10$.

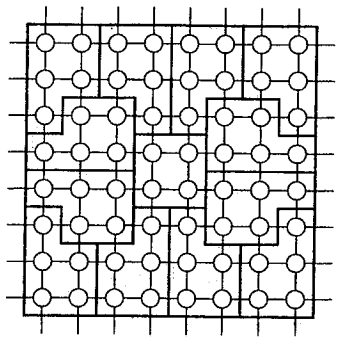


Figure 8: Optimal partition of $G_{8,2}$ into 13 subgraphs

There is clearly a general principle at work here, for uniformly weighted graphs. If there are M nodes to be assigned to P processors, then at least one processor will receive $m = \lceil M/P \rceil$ nodes. When the processor cost function is monotone non-decreasing as a function of the number of nodes assigned to it, $wm + C(m)$ is a lower bound on the optimal bottleneck cost, C being the appropriate minimized function for communication cost. If it is possible to partition the graph so that no processor has cost greater than $wm + C(m)$, then that partition is optimal.

7 Conclusions

A subgraph of a k-ary n-cube can be viewed as having internal edges and external edges. This paper describes how to construct subgraphs that are optimal in the sense of maximizing the number of internal edges, thus minimizing the number of external edges, given m nodes in the subgraph. While these results have combinatorial interest, they also have serious applications to problems in parallel processing. We show, for instance, how to apply these results in the context of branch-and-bound algorithms for partitioning a k-ary n-cube whose nodes and edges have general (positive) weights. Lower bounds lie at the heart of any branch-and-bound algorithm, and our results provide the critical means needed to compute sharper bounds than those that ignore communication overheads. We also show how our results can be used to demonstrate the optimality of certain irregular partitions. k-ary n-cubes arise frequently in studies of parallel processing. The results and applications developed here help us to better understand these important graphs.

Appendix[4]

PROPERTY **1.5** *In each i^{th} composite subcube ($0 \leq i \leq k-1$) of type $G_{k,n-1}$ in $G_{k,n}$, choose m_i nodes, and define $m = \sum_{i=0}^{k-1} m_i$. The number of edges with endpoints among these m nodes but in different composite subcubes is no larger than $\min\{m_0, m_1\}$ for $k = 2$, and is no larger than $m - \max_{0 \leq i \leq k-1}\{m_i\} + \min_{0 \leq i \leq k-1}\{m_i\}$ for $k \geq 3$.*

[4]To referees: Proofs in this section have all been verified by programs.

Proof We observe that if the k composite subcubes of type $G_{k,n-1}$ are placed from left to right, any node in one composite subcube is connected to exactly one node in its neighboring composite subcubes. When $k = 2$, it is trivial that the number of edges with endpoints among the m nodes but in different composite subcubes is no larger than $\min\{m_0, m_1\}$. Now consider $k \geq 3$. Clearly, the number of edges with endpoints among the m nodes but in different composite subcubes is no larger than

$$\min\{m_0, m_1\} + \min\{m_1, m_2\} + \cdots + \min\{m_{k-2}, m_{k-1}\} + \min\{m_{k-1}, m_0\}.$$

Define $i \dot{+} 1 = i + 1 (\text{mod } k)$ and $i \dot{-} 1 = i - 1 (\text{mod } k)$. Let $m_p = \max_{0 \leq i \leq k-1}\{m_i\}$ and $m_q = \min_{0 \leq i \leq k-1}\{m_i\}$. Place k pairs $(m_0, m_1), (m_1, m_2), \ldots, (m_{k-2}, m_{k-1}), (m_{k-1}, m_0)$ in a circle clockwise. Cut the circle into two chains C_1 and C_2 such that $C_1 = \{(m_p, m_{p\dot{+}1}), \ldots, (m_{q\dot{-}1}, m_q)\}$ and $C_2 = \{(m_q, m_{q\dot{+}1}), \ldots, (m_{p\dot{-}1}, m_p)\}$. Clearly,

$$\sum_{(m_i, m_{i\dot{+}1}) \in C_1} \min\{m_i, m_{i\dot{+}1}\} \leq \sum_{i=p\dot{+}1}^{q} m_i$$

and

$$\sum_{(m_i, m_{i\dot{+}1}) \in C_2} \min\{m_i, m_{i\dot{+}1}\} \leq \sum_{i=q}^{p\dot{-}1} m_i.$$

Consequently,

$$\begin{aligned}
& \sum_{i=0}^{k-1} \min\{m_i, m_{i\dot{+}1}\} \\
= & \sum_{(m_i, m_{i\dot{+}1}) \in C_1} \min\{m_i, m_{i\dot{+}1}\} + \sum_{(m_i, m_{i\dot{+}1}) \in C_2} \min\{m_i, m_{i\dot{+}1}\} \\
\leq & \sum_{i=p\dot{+}1}^{q} m_i + \sum_{i=q}^{p\dot{-}1} m_i \\
= & \sum_{i=0}^{k-1} m_i - m_p + m_q \\
= & m - \max_{0 \leq i \leq k-1}\{m_i\} + \min_{0 \leq i \leq k-1}\{m_i\}. \blacksquare
\end{aligned}$$

LEMMA 2.1 *$W(i, 2i-1) = W(0, i-1) + i$ for $i \geq 1$.*

Proof We induct on i. When $i = 1$, it is trivial that $W(1,1) = W(0,0) + 1$. Assume that the equation holds for $\leq i - 1$. Now consider i.

$$\begin{aligned}
W(i, 2i-1) &= W(i-1, 2i-3) + w(2i-2) + w(2i-1) - w(i-1) \\
&= W(0, i-2) + (i-1) + w(2i-2) + w(2i-1) - w(i-1) \\
&\quad \text{(Inductive hypothesis)}
\end{aligned}$$

$$
\begin{aligned}
&= W(0, i-2) + (i-1) + w(2i-1) \ \text{(Since } w(2i-2) = w(i-1)\text{)} \\
&= W(0, i-2) + (i-1) + w(i-1) + 1 \ \text{(Since } w(2i-1) = w(i-1) + 1\text{)} \\
&= W(0, i-1) + i. \blacksquare
\end{aligned}
$$

LEMMA **2.2** *$W(i+1, 2i) = W(0, i-1) + i$ for $i \geq 1$.*

Proof Straightforward by using Lemma 2.1. ■

LEMMA **2.3** *$W(j, j+i-1) \geq W(0, i-1) + i$ for $j \geq i \geq 1$.*

Proof We induct on i. When $i = 1$, it is obvious that $W(j, j) \geq W(0, 0) + 1$. Assume that the inequality holds for $\leq i-1$. That is, for $j' \geq i' \geq 1$ and $i' \leq i-1$,

$$W(j', j'+i'-1) \ \geq \ W(0, i'-1) + i'. \tag{1}$$

An important implication of the inductive hypothesis is that when $j' + i' \leq 2^b$ for some b, if we replace all parameters of W in (1) by their $(2^b - 1)$-complements, we have

$$W(2^b - i', 2^b - 1) \ \geq \ W(2^b - j' - i', 2^b - j' - 1) + i'. \tag{2}$$

Now consider i.

Case 1. There exists 2^b in $(j, j+i-1]$ for some b, and $j+i-1$ also has $b+1$ bits and starts with 1 in its base-2 representation. By (1) and (2),

$$W(j, 2^b - 1) \ \geq \ W(j + i - 2^b, i-1) + (2^b - j). \tag{3}$$

Removing the highest bit 1 from $2^b, \ldots, j+i-1$,

$$W(2^b, j+i-1) \ = \ W(0, j+i-2^b-1) + (j+i-2^b). \tag{4}$$

Adding (3) and (4),

$$W(j, j+i-1) \geq W(0, i-1) + i.$$

Case 2. There is no number equal to 2^b in $(j, j+i-1]$ for any b. We then know that $j, \ldots, j+i-1$ must all have the same number of bits, say $b+1$, and the same highest bit 1. Let $p_1 \cdots p_t$ be the longest common prefix of the base-2 representations of $j, \ldots, j+i-1$. Let $p = p_1 \cdot 2^b + \cdots + p_t \cdot 2^{b-t+1}$. Clearly, $p \leq j$ and $p_1 \geq 1$. Removing the highest t bits $p_1 \cdots p_t$ from $j, \ldots, j+i-1$,

$$W(j, j+i-1) \geq W(j-p, j+i-p-1) + i.$$

Now we wish to show that $W(j-p, j+i-p-1) \geq W(0, i-1)$, or equivalently $W(j-p, j+i-p-1) - W(0, i-1) \geq 0$. If $j - p < i$,

$$
\begin{aligned}
W(j-p, j+i-p-1) - W(0, i-1) \ &= \ W(i, j+i-p-1) - W(0, j-p-1) \\
&\geq \ j - p \ \text{(By (1))} \\
&\geq \ 0.
\end{aligned}
$$

If $j-p \geq i$, there must exist $2^{b'}$ in $(j-p, j+i-p-1]$ for some $b' < b$ since the highest bit in $j-p$ is not the same as the highest bit in $j+i-p-1$. By Case 1,

$$W(j-p, j+i-p-1) - W(0, i-1) \geq i > 0. \blacksquare$$

LEMMA **3.1** *$Z(0, 3i-1) = 3Z(0, i-1) + 3i$ for $i \geq 1$.*

Proof We induct on i. When $i = 1$, $Z(0,2) = 3Z(0,0) + 3 = 3$. Assume that the equation holds for $\leq i-1$. Now consider i.

$$\begin{aligned}
Z(0,3i-1) &= Z(0,3i-4) + z(3i-3) + z(3i-2) + z(3i-1) \\
&= 3Z(0,i-2) + 3(i-1) + z(3i-3) + z(3i-2) + z(3i-1) \\
&\quad \text{(Inductive hypothesis)} \\
&= 3Z(0,i-2) + 3(i-1) + z(i-1) + z(3i-2) + z(3i-1) \\
&\quad \text{(Since } z(3i-3) = z(i-1)\text{)} \\
&= 3Z(0,i-2) + 3(i-1) + z(i-1) + z(i-1) + 1 + z(3i-1) \\
&\quad \text{(Since } z(3i-2) = z(i-1) + 1\text{)} \\
&= 3Z(0,i-2) + 3(i-1) + z(i-1) + z(i-1) + 1 + z(i-1) + 2 \\
&\quad \text{(Since } z(3i-1) = z(i-1) + 2\text{)} \\
&= 3Z(0,i-1) + 3i. \blacksquare
\end{aligned}$$

LEMMA **3.2** *$Z(0, 3i) = Z(0, i) + 2Z(0, i-1) + 3i$ for $i \geq 1$.*

Proof Straightforward by using Lemma 3.1. ■

LEMMA **3.3** *$Z(0, 3i+1) = 2Z(0, i) + Z(0, i-1) + 3i + 1$ for $i \geq 1$.*

Proof Straightforward by using Lemma 3.2. ■

LEMMA **3.4** *$Z(j, j+i-1) \geq Z(0, i-1) + i$ for $j \geq i \geq 1$.*

Proof Use the proof of Lemma 2.3, but change W to Z and base-2 to base-3. To be more specific, in the inductive step, consider the following two cases.

Case 1. There exists 3^b or $2 \cdot 3^b$ in $(j, j+i-1]$ for some b, and $j+i-1$ also has $b+1$ bits and starts with 1 or 2, respectively, in its base-3 representation.

Case 2. There is no number equal to 3^b or $2 \cdot 3^b$ in $(j, j+i-1]$ for any b. ■

Before we go to prove Lemma 3.5, we need following claims.

CLAIM **1** *$Z(j, j+i-1) \geq Z(0, i-1)$ for $j \geq 0$ and $i \geq 1$.*

Proof Trivial by Lemma 3.4. ■

CLAIM **2** *$Z(l-i, l-1) \geq Z(l-j-i, l-j-1) + i$ for $j \geq i \geq 1$, $j+i \leq l$ and $l = 3^b$ or $2 \cdot 3^b$ for some b.*

Proof Replace all parameters of Z in Lemma 3.4 by their $(l-1)$-complements. ■

CLAIM 3 *$Z(l-i,l-1) \geq Z(l-j-i,l-j-1)$ for $j \geq 0$, $i \geq 1$, $j+i \leq l$ and $l = 3^b$ or $2 \cdot 3^b$ for some b.*

Proof Replace all parameters of Z in Claim 1 by their $(l-1)$-complements. ■

LEMMA **3.5** *$Z(j, j+i_1+i_2-1) \geq Z(0,i_1-1)+Z(0,i_2-1)+i_1+2i_2$ for $j \geq i_1 \geq i_2 \geq 1$.*

Proof We induct on i_1+i_2. When $i_1+i_2=2$, we must have $i_1=i_2=1$. It is obvious that $Z(j,j+1) \geq Z(0,0)+Z(0,0)+3$. Assume that the inequality holds for $\leq i_1+i_2-1$. That is, for $j' \geq i_1' \geq i_2' \geq 1$ and $i_1'+i_2' \leq i_1+i_2-1$,

$$Z(j', j'+i_1'+i_2'-1) \quad \geq \quad Z(0,i_1'-1)+Z(0,i_2'-1)+i_1'+2i_2'. \tag{5}$$

An important implication of the inductive hypothesis is that when $j'+i_1'+i_2' \leq l$ and $l = 3^b$ or $2 \cdot 3^b$ for some b, if we replace all parameters of Z in (5) by their $(l-1)$-complements, we have

$$Z(l-i_1', l-1)+Z(l-i_2', l-1) \quad \geq \quad Z(l-j'-i_1'-i_2', l-j'-1)+i_1'+2i_2'. \tag{6}$$

Now consider i_1+i_2.

Case 1. There exists $2 \cdot 3^b$ in $(j, j+i_1+i_2-1]$ for some b, and $j+i_1+i_2-1$ also has $b+1$ bits and starts with 2 in its base-3 representation.

Subcase 1.1. There is 3^b in $(j, 2 \cdot 3^b)$. Removing the highest bit 1 from $3^b, \ldots, 3^b+i_1-1$,

$$Z(3^b, 3^b+i_1-1) = Z(0,i_1-1)+i_1. \tag{7}$$

Removing the highest bit 2 from $2 \cdot 3^b, \ldots, j+i_1+i_2-1$,

$$Z(2 \cdot 3^b, j+i_1+i_2-1) = Z(0, j+i_1+i_2-2 \cdot 3^b-1)+2(j+i_1+i_2-2 \cdot 3^b). \tag{8}$$

Removing the highest bit 1 from $3^b+i_1, \ldots, 2 \cdot 3^b-1$,

$$Z(3^b+i_1, 2 \cdot 3^b-1) = Z(i_1, 3^b-1)+(3^b-i_1). \tag{9}$$

Next,

$$\begin{aligned} & Z(j,3^b-1)+Z(3^b+i_1, 2 \cdot 3^b-1) \\ = \; & Z(j,3^b-1)+Z(i_1,3^b-1)+(3^b-i_1) \quad \text{(By (9))} \\ \geq \; & Z(j+i_1+i_2-2 \cdot 3^b, i_2-1)+(3^b-i_1)+2(3^b-j)+(3^b-i_1) \quad \text{(By (5) (6))}. \end{aligned} \tag{10}$$

Adding (7), (8) and (10),

$$Z(j, j+i_1+i_2-1) \geq Z(0,i_1-1)+Z(0,i_2-1)+i_1+2i_2.$$

Subcase 1.2. There is no number equal to 3^b in $(j, 2\cdot 3^b)$. We then know that j must have $b+1$ bits and be at least 3^b.

Subsubcase 1.2.1. Assume that $j + i_1 + i_2 - 2\cdot 3^b \geq i_2$. Removing the highest bit 2 from $2\cdot 3^b, \ldots, 2\cdot 3^b + i_2 - 1$,

$$Z(2\cdot 3^b, 2\cdot 3^b + i_2 - 1) = Z(0, i_2 - 1) + 2i_2. \tag{11}$$

Removing the highest bit 2 from $2\cdot 3^b + i_2, \ldots, j + i_1 + i_2 - 1$,

$$\begin{aligned} Z(2\cdot 3^b + i_2, j + i_1 + i_2 - 1) &= Z(i_2, j + i_1 + i_2 - 2\cdot 3^b - 1) + 2(j + i_1 - 2\cdot 3^b) \\ &\geq Z(0, j + i_1 - 2\cdot 3^b - 1) + 2(j + i_1 - 2\cdot 3^b) \text{ (Claim 1)} \\ &= Z(2\cdot 3^b, j + i_1 - 1). \end{aligned} \tag{12}$$

Therefore,

$$\begin{aligned} & Z(j, j + i_1 + i_2 - 1) \\ = \; & Z(j, 2\cdot 3^b - 1) + Z(2\cdot 3^b, 2\cdot 3^b + i_2 - 1) + Z(2\cdot 3^b + i_2, j + i_1 + i_2 - 1) \\ \geq \; & Z(j, 2\cdot 3^b - 1) + Z(0, i_2 - 1) + Z(2\cdot 3^b, j + i_1 - 1) + 2i_2 \text{ (By (11) (12))} \\ = \; & Z(j, j + i_1 - 1) + Z(0, i_2 - 1) + 2i_2 \\ \geq \; & Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2 \text{ (Lemma 3.4)} \end{aligned}$$

Subsubcase 1.2.2. Assume that $j + i_1 + i_2 - 2\cdot 3^b < i_2$. Removing the highest bit 2 from $2\cdot 3^b, \ldots, j + i_1 + i_2 - 1$,

$$Z(2\cdot 3^b, j + i_1 + i_2 - 1) = Z(0, j + i_1 + i_2 - 2\cdot 3^b - 1) + 2(j + i_1 + i_2 - 2\cdot 3^b). \tag{13}$$

By Lemma 3.4,

$$Z(j, j + i_1 - 1) \geq Z(0, i_1 - 1) + i_1. \tag{14}$$

Removing the highest bit 1 from $j + i_1, \ldots, 2\cdot 3^b - 1$,

$$\begin{aligned} Z(j + i_1, 2\cdot 3^b - 1) &= Z(j + i_1 - 3^b, 3^b - 1) + (2\cdot 3^b - j - i_1) \\ &\geq Z(j + i_1 + i_2 - 2\cdot 3^b, i_2 - 1) + 2(2\cdot 3^b - j - i_1) \text{ (Claim 2).} \end{aligned} \tag{15}$$

Adding (13), (14) and (15),

$$Z(j, j + i_1 + i_2 - 1) \geq Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2.$$

Case 2. There exists 3^b in $(j, j + i_1 + i_2 - 1]$ for some b, and $j + i_1 + i_2 - 1$ also has $b+1$ bits and starts with 1 in its base-3 representation.

Subcase 2.1. There is $2\cdot 3^{b-1}$ in $(j, 3^b)$.

Subsubcase 2.1.1. Assume that $i_2 \leq 3^{b-1}$. Removing the highest bit 2 from $2\cdot 3^{b-1}, \ldots, 2\cdot 3^{b-1} + i_2 - 1$,

$$Z(2\cdot 3^{b-1}, 2\cdot 3^{b-1} + i_2 - 1) = Z(0, i_2 - 1) + 2i_2. \tag{16}$$

Removing the highest bit 1 from $3^b, \ldots, j + i_1 + i_2 - 1$,

$$Z(3^b, j + i_1 + i_2 - 1) = Z(0, j + i_1 + i_2 - 3^b - 1) + (j + i_1 + i_2 - 3^b). \tag{17}$$

By Claim 2,

$$Z(j, 2 \cdot 3^{b-1} - 1) \geq Z(j + i_1 - 2 \cdot 3^{b-1}, i_1 - 1) + (2 \cdot 3^{b-1} - j) \tag{18}$$

and

$$Z(2 \cdot 3^{b-1} + i_2, l - 1) \geq Z(j + i_1 + i_2 - 3^b, j + i_1 - 2 \cdot 3^{b-1} - 1) + (3^{b-1} - i_2). \tag{19}$$

Adding (16), (17), (18) and (19),

$$Z(j, j + i_1 + i_2 - 1) \geq Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2.$$

Subsubcase 2.1.2. Assume that $i_2 > 3^{b-1}$. Then $j \geq i_1 \geq i_2 > 3^{b-1}$.

$$\begin{aligned} & Z(j - 3^{b-1}, j + i_1 + i_2 - 3^b - 1)) \\ \geq\ & Z(0, i_1 - 3^{b-1} - 1) + Z(0, i_2 - 3^{b-1} - 1) + (i_1 - 3^{b-1}) + 2(i_2 - 3^{b-1}). \text{(By (5))} \end{aligned} \tag{20}$$

Let $h = \min\{3^b + 2 \cdot 3^{b-1}, j + i_1 + i_2\}$.

$$\begin{aligned} & Z(h - 2 \cdot 3^{b-1}, h - 1) \\ \geq\ & Z(0, 3^{b-1} - 1) + Z(0, 3^{b-1} - 1) + 3^{b-1} + 2(3^{b-1}). \text{ (By (5))} \end{aligned} \tag{21}$$

Subtracting 3^{b-1} from $j, \ldots, h - 2 \cdot 3^{b-1} - 1$,

$$Z(j, h - 2 \cdot 3^{b-1} - 1) = Z(j - 3^{b-1}, h - 3^b - 1) + (h - j - 2 \cdot 3^{b-1}). \tag{22}$$

In the case of $h = 3^b + 2 \cdot 3^{b-1}$, removing the highest bit 1 from $3^b + 2 \cdot 3^{b-1}, \ldots, j + i_1 + i_2 - 1$,

$$\begin{aligned} & Z(h, j + i_1 + i_2 - 1) \\ =\ & Z(2 \cdot 3^{b-1}, j + i_1 + i_2 - 3^b - 1) + (j + i_1 + i_2 - 3^b - 2 \cdot 3^{b-1}). \end{aligned} \tag{23}$$

Adding (22) and (23),

$$\begin{aligned} & Z(j, h - 2 \cdot 3^{b-1} - 1) + Z(h, j + i_1 + i_2 - 1) \\ =\ & Z(j - 3^{b-1}, j + i_1 + i_2 - 3^b - 1) + (i_1 + i_2 - 2 \cdot 3^{b-1}) \\ \geq\ & Z(0, i_1 - 3^{b-1} - 1) + Z(0, i_2 - 3^{b-1} - 1) + (i_1 - 3^{b-1}) + 2(i_2 - 3^{b-1}) \\ & + (i_1 + i_2 - 2 \cdot 3^{b-1}) \text{ (By (20))} \\ =\ & Z(3^{b-1}, i_1 - 1) + Z(3^{b-1}, i_2 - 1) + (i_1 - 3^{b-1}) + 2(i_2 - 3^{b-1}). \end{aligned} \tag{24}$$

Adding (21) and (24),

$$Z(j, j + i_1 + i_2 - 1) \geq Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2.$$

Subcase 2.2. There is no number equal to $2 \cdot 3^{b-1}$ in $(j, 3^b)$. We then know that j must be at least $2 \cdot 3^{b-1}$.

Subsubcase 2.2.1. Assume that $j + i_1 + i_2 - 3^b \geq i_1$. Removing the highest bit 1 from $3^b, \ldots, 3^b + i_1 - 1$,

$$Z(3^b, 3^b + i_1 - 1) = Z(0, i_1 - 1) + i_1. \tag{25}$$

Removing the highest bit 1 from $3^b + i_1, \ldots, j + i_1 + i_2 - 1$,

$$\begin{aligned} Z(3^b + i_1, j + i_1 + i_2 - 1) &= Z(i_1, j + i_1 + i_2 - 3^b - 1) + (j + i_2 - 3^b) \\ &\geq Z(0, j + i_2 - 3^b - 1) + 2(j + i_2 - 3^b) \text{ (Lemma 3.4).} \end{aligned} \tag{26}$$

Removing the highest bit 2 from $j, \ldots, 3^b - 1$,

$$\begin{aligned} Z(j, 3^b - 1) &= Z(j - 2 \cdot 3^{b-1}, 3^{b-1} - 1) + 2(3^b - j) \\ &\geq Z(j + i_2 - 3^b, i_2 - 1) + 2(3^b - j) \text{ (Claim 3).} \end{aligned} \tag{27}$$

Adding (25), (26) and (27),

$$Z(j, j + i_1 + i_2 - 1) \geq Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2.$$

Subsubcase 2.2.2. Assume that $j + i_1 + i_2 - 3^b < i_1$. Removing the highest bit 1 from $3^b, \ldots, j + i_1 + i_2 - 1$,

$$Z(3^b, j + i_1 + i_2 - 1) = Z(0, j + i_1 + i_2 - 3^b - 1) + (j + i_1 + i_2 - 3^b). \tag{28}$$

Removing the highest bit 2 from $j, j + i_2 - 1$,

$$\begin{aligned} Z(j, j + i_2 - 1) &= Z(j - 2 \cdot 3^{b-1}, j + i_2 - 2 \cdot 3^{b-1} - 1) + 2i_2 \\ &\geq Z(0, i_2 - 1) + 2i_2 \text{ (Claim 1).} \end{aligned} \tag{29}$$

By Claim 2,

$$Z(j + i_2, 3^b - 1) \geq Z(j + i_1 + i_2 - 3^b, i_1 - 1) + (3^b - j - i_2). \tag{30}$$

Adding (28), (29) and (30),

$$Z(j + j + i_1 + i_2 - 1) \geq Z(0, i_1 - 1) + Z(0, i_2 - 1) + i_1 + 2i_2.$$

Case 3. There is no number equal to 3^b or $2 \cdot 3^b$ in $(j, j + i_1 + i_2 - 1]$ for any b. We then know that $j, \ldots, j + i_1 + i_2 - 1$ must all have the same number of bits, say $b+1$, and the same highest bit 1 or 2. Let $p_1 \cdots p_t$ be the longest common prefix of the base-3 representations of $j, \ldots, j + i_1 + i_2 - 1$. Let $p = p_1 \cdot 3^b + \cdots + p_t \cdot 3^{b-t+1}$. Clearly, $p \leq j$ and $p_1 \geq 1$. Removing the highest t bits $p_1 \cdots p_t$ from $j, \ldots, j + i_1 + i_2 - 1$,

$$Z(j, j + i_1 + i_2 - 1) \geq Z(j - p, j + i_1 + i_2 - p - 1) + (i_1 + i_2).$$

Now we wish to show that $Z(j-p, j+i_1+i_2-p-1) \geq Z(0, i_1-1) + Z(0, i_2-1) + i_2$, or equivalently, $Z(j-p, j+i_1+i_2-p-1) - Z(0, i_1-1) - Z(0, i_2-1) \geq i_2$. If $j-p < i_1$,

$$\begin{aligned} & Z(j-p, j+i_1+i_2-p-1) - Z(0, i_1-1) - Z(0, i_2-1) \\ = \; & Z(i_1, j+i_1+i_2-p-1) - Z(0, j-p-1) - Z(0, i_2-1) \\ \geq \; & \max\{j-p, i_2\} + 2\min\{j-p, i_2\} \;\; \text{(By (5))}. \\ \geq \; & i_2. \end{aligned}$$

If $j-p \geq i_1$, there must exist $3^{b'}$ or $2 \cdot 3^{b'}$ in $(j-p, j+i_1+i_2-p-1]$ for some $b' < b$ since the highest bit in $j-p$ is not the same as the highest bit in $j+i_1+i_2-p-1$. By Cases 1 and 2,

$$Z(j-p, j+i_1+i_2-p-1) - Z(0, i_1-1) - Z(0, i_2-1) \geq i_1 + 2i_2 > i_2. \blacksquare$$

LEMMA **4.1** *$W(0, 4i-1) = 4W(0, i-1) + 4i$ for $i \geq 1$.*

Proof We induct on i. When $i = 1$, $W(0,3) = 4W(0,0) + 4 = 4$. Assume that the equation holds for $\leq i-1$. Now consider i.

$$\begin{aligned} W(0, 4i-1) &= W(0, 4i-5) + w(4i-4) + w(4i-3) + w(4i-2) + w(4i-1) \\ &= 4W(0, i-2) + (4i-4) + w(4i-4) + w(4i-3) + w(4i-2) + w(4i-1) \\ &\quad \text{(Inductive hypothesis)} \\ &= 4W(0, i-2) + (4i-4) + w(i-1) + w(4i-3) + w(4i-2) + w(4i-1) \\ &\quad \text{(Since } w(4i-4) = w(i-1)\text{)} \\ &= 4W(0, i-2) + (4i-4) + w(i-1) + w(i-1) + 1 + w(4i-2) \\ &\quad + w(4i-1) \;\; \text{(Since } w(4i-3) = w(i-1) + 1\text{)} \\ &= 4W(0, i-2) + (4i-4) + w(i-1) + w(i-1) + 1 + w(i-1) + 1 \\ &\quad + w(4i-1) \;\; \text{(Since } w(4i-2) = w(i-1) + 1\text{)} \\ &= 4W(0, i-2) + (4i-4) + w(i-1) + w(i-1) + 1 + w(i-1) + 1 \\ &\quad + w(i-1) + 2 \;\; \text{(Since } w(4i-1) = w(i-1) + 2\text{)} \\ &= 4W(0, i-1) + 4i. \blacksquare \end{aligned}$$

LEMMA **4.2** *$W(0, 4i) = W(0, i) + 3W(0, i-1) + 4i$ for $i \geq 1$.*

Proof Straightforward by using Lemma 4.1. ■

LEMMA **4.3** *$W(0, 4i+1) = 2W(0, i) + 2W(0, i-1) + 4i + 1$ for $i \geq 1$.*

Proof Straightforward by using Lemma 4.2. ■

LEMMA **4.4** *$W(0, 4i+2) = 3W(0, i) + W(0, i-1) + 4i + 2$ for $i \geq 1$.*

Proof Straightforward by using Lemma 4.3. ■

References

[1] D. Agrawal, M. Choy, H. V. Leong, and A. Singh, *Maya: A simulation platform for distributed shared memories*, Proc. 8^{th} Workshop on Parallel and Distributed Simulation, 1994, pp. 151–155.

[2] S. H. Bokhari, *Partitioning problems in parallel, pipelined, and distributed computing*, IEEE Trans. on Comput., 37 (1988), pp. 48–57.

[3] G. Brassard and P. Bratley, *Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[4] W. J. Dally, *Performance analysis of k-ary n-cube interconnection networks*, IEEE Trans. on Comput., 39 (1990), pp. 775–785.

[5] P. Dickens, P. Heidelberger, and D. M. Nicol, *A distributed memory LAPSE: Parallel simulation of message-passing programs*, Proc. 8^{th} Workshop on Parallel and Distributed Simulation, 1994, pp. 32–38.

[6] A. Marshal and I. Olkin, *Inequalities : Theory of Majorization and Its Application*, Academic Press, New York, 1979.

[7] C. McCann and J. Zahorjan, *Processor allocation policies for message-passing parallel computers*, Proc. the 1994 SIGMETRICS Conference, 1994, pp. 19–32.

[8] G. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis, *Automatic mesh partitioning*, in Graph Theory and Sparse Matrix Computation, Springer-Verlag, 1993.

[9] D. M. Nicol, *Rectilinear partitioning of irregular data parallel computations*, J. of Parallel and Distributed Computing, to appear.

[10] D. M. Nicol and W. Mao, *On bottleneck partitioning of k-ary n-cubes*, ICASE Technical Report 94-72, NASA CR-194966, 1994.

[11] A. Pothen, H. D. Simon, and K. P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. on Matrix Analysis and Applications, 11 (1990), pp. 430–452.

[12] D. A. Reed, L. M. Adams, and M. L. Patrick, *Stencils and problem partitionings: Their influence on the performance of multiple processor systems*, IEEE Trans. on Comput., 36 (1987), pp. 845–858.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1994

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
ON kARY n-CUBES: THEORY AND APPLICATIONS

5. FUNDING NUMBERS
C NAS1-19480
WU 505-90-52-01

6. AUTHOR(S)
Weizhen Mao
David M. Nicol

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science
and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 94-88

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-194996
ICASE Report No. 94-88

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report
Submitted to SIAM Journal of Computing

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 60, 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many parallel processing networks can be viewed as graphs called k-ary n-cubes, whose special cases include rings, hypercubes and toruses. In this paper, combinatorial properties of k-ary n-cubes are explored. In particular, the problem of characterizing the subgraph of a given number of nodes with the maximum edge count is studied. These theoretical results are then used to compute a lower bounding function in branch-and-bound partitioning algorithms and to establish the optimality of some irregular partitions.

14. SUBJECT TERMS
k-ary n-cube, combinatorics, partitioning, load balance

15. NUMBER OF PAGES
29

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		