DAM 2954
pp: 1-23 (col.fig.: NIL)
PROD. TYPE: COM
ED: GRACY
PAGN: VS – SCAN: VKUMAR

ARTICLE IN PRESS

ELSEVIER

Discrete Applied Mathematics ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

DISCRETE APPLIED MATHEMATICS

On k-ary n-cubes: theory and applications

Mao Weizhen[a,*], David M. Nicol[b]

[a] Department of Computer Science, College of William and Mary, P.O. Box 8795, Williamsburg, VA 23187-8795, USA

[b] Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA

Abstract

Many parallel processing applications have communication patterns that can be viewed as graphs called k-ary n-cubes, whose special cases include rings, hypercubes and tori. In this paper, combinatorial properties of k-ary n-cubes are examined. 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 applied to compute a lower bounding function in branch-and-bound partitioning algorithms and to establish the optimality of some irregular partitions. © 2002 Published by Elsevier Science B.V.

Keywords: Parallel processing; Communication; k-Ary n-cubes; Graph partitioning

1. Introduction

In a *k-ary n-cube* [2,7,14], each node is identified by an n-bit base-k address $b_{n-1}\dots b_i\dots b_0$, and for each *dimension* $i=0,1,\dots,n-1$, the node is connected by edges to nodes with addresses $b_{n-1}\dots b_i \pm 1 \,(\mathrm{mod}\, k)\dots b_0$.

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

Hypothesis 1. A k-ary 1-cube is a ring of k nodes. Without loss of generality, we place the k nodes on a line, and call the leftmost node the 0th position node and the rightmost node the $(k-1)$th position node.

* Corresponding author.
E-mail address: wm@cs.wm.edu (M. Weizhen).

PII: S0166-218X(02)00238-X

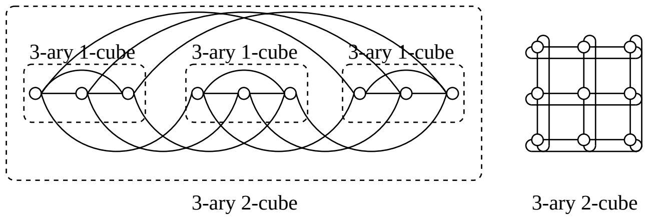


Fig. 1. A 3-ary 2-cube.

Table 1
Special cases of k-ary n-cubes

k	x		
	1	2	$\geqslant 3$
1	Point (ring)	Point (torus)	Point
2	Edge (hypercube/ring)	Square (hypercube/torus)	Hypercube
$\geqslant 3$	Ring	Torus	k-Ary n-cube

Hypothesis 2. A k-ary n-cube contains k *composite subcubes*, each of which is a k-ary $(n-1)$-cube, 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 ith position nodes into a ring.

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

The second and the third definitions of k-ary n-cubes provide two ways of drawing the graphs. See Fig. 1 for an example.

The class of k-ary n-cubes contains as special cases many topologies important to parallel processing, such as rings, hypercubes, and tori. Hence, a thorough study of k-ary n-cubes is necessary. Table 1 summarizes the special cases of k-ary n-cubes. In the table the first column contains values of k and the first row contains values of n.

In this paper, we study combinatorial properties of k-ary n-cubes and their applications to the graph partitioning problem for parallel processing. We organize the paper as follows. In Section 2, we give some simple properties of k-ary n-cubes. In Section 3, we give a visual description of *edge isoperimetric subgraphs*, which are subgraphs of a fixed node count in k-ary n-cubes that achieve the maximal internal edge count, when the wrap-around edges can be discounted. In Section 4, we give formulas that compute the maximal edge count in an edge isoperimetric subgraph for three special cases. In Section 5, we apply our theoretical results to graph partitioning. Finally in Section 6, we make our conclusions.

2. Combinatorial properties of *k*-ary *n*-cubes

The following combinatorial properties of k-ary n-cubes are easy to verify except perhaps the last one, for which we provide its proof.

Proposition 1. *A k-ary n-cube has k^n nodes.*

Proposition 2. *A k-ary n-cube contains k composite subcubes, each of which is a k-ary $(n-1)$-cube, and the number of edges with endpoints in different composite subcubes is k^{n-1} for $k = 2$ and k^n for $k \geqslant 3$.*

Proposition 3. *A k-ary n-cube 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 $k \geqslant 3$.*

Proposition 4. *The number of edges in a k-ary n-cube is nk^{n-1} for $k=2$ and nk^n for $k \geqslant 3$.*

Proposition 5. *Consider the recursive definition of a k-ary n-cube. In each ith composite subcube with $0 \leqslant i \leqslant k-1$, which is a k-ary $(n-1)$-cube, choose m_i nodes. Define $m = \sum_{i=0}^{k-1} m_i$. Then 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 no larger than $m - \max_{0 \leqslant i \leqslant k-1}\{m_i\} + \min_{0 \leqslant i \leqslant k-1}\{m_i\}$ for $k \geqslant 3$.*

Proof. We observe that if the k composite subcubes, which are k-ary $(n-1)$-cubes, 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 $k \geqslant 3$. Define $i \dot{+} 1 = i + 1 \pmod k$ and $i \dot{-} 1 = i - 1 \pmod k$. Let $m_p = \max_{0 \leqslant i \leqslant k-1}\{m_i\}$ and $m_q = \min_{0 \leqslant i \leqslant k-1}\{m_i\}$. 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,

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

and

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

We observe that the number of edges with endpoints among the m nodes but in different composite subcubes is no larger than

$$\begin{aligned}&\min\{m_0, m_1\} + \cdots + \min\{m_{k-2}, m_{k-1}\} + \min\{m_{k-1}, m_0\}\\ &= \sum_{i=0}^{k-1} \min\{m_i, m_{i \dot{+} 1}\}\end{aligned}$$

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

3. Edge isoperimetric subgraphs for *k*-ary *n*-cubes

Given a graph and an integer m, which is no larger than the total number of nodes in the graph. For any subgraph of m nodes, an *internal edge* is one with both endpoints in the subgraph while an *external edge* is one with only one endpoint in the subgraph. An *edge isoperimetric subgraph* of m nodes is one, among all subgraphs of m node, with the maximum number of internal edges. Note that the edge isoperimetric property is also studied in the context of finding a subgraph with the minimum number of external edges. However, since in this paper the object of interest is k-ary n-cubes, which are regular graphs with a fixed degree for each node, for subgraphs of m nodes minimizing the number of external edges is, in fact, equivalent to maximizing the number of internal edges. Thus, we focus on edge isoperimetric subgraphs for k-ary n-cubes with the maximum number of internal edges.

The edge isoperimetric problems on general graphs are surveyed in [3]. Although the edge isoperimetric property for k-ary n-cubes has not been studied directly, there are a few results relevant to our work. For instance, the construction, based on a lexicographic order of nodes, of edge isoperimetric subgraphs in a hypercube is given in [10], and a similar method is used to determine edge isoperimetric subgraphs in an n-D mesh (without wrap-around edges as in k-ary n-cubes) in [1,5]. Our research differs from previous work mainly in that we are interested in a visual description of edge isoperimetric subgraphs in k-ary n-cubes. That is, given a k-ary n-cube and an integer m (no larger than k^n), what does a subgraph of m nodes which achieves the maximum internal edge count look like? An important assumption we use throughout this section is that k is so large relative to m that an edge isoperimetric subgraph cannot possibly include wrap-around edges. (In Section 4, this assumption will be removed for the consideration of several special cases.) Intuition tells us that under our assumption the maximum number of internal edges, denoted as $e_{k,n}(m)$, can be obtained when the m nodes are placed as tightly as possible to form a “cubish” polyhedron in the k-ary n-cube. In the remainder of this section, we prove that our intuition turns out to be correct.

Along any dimension, a subgraph of m nodes in a k-ary n-cube can be partitioned into *layers*, each of which contains nodes with the same coordinate in the dimension.

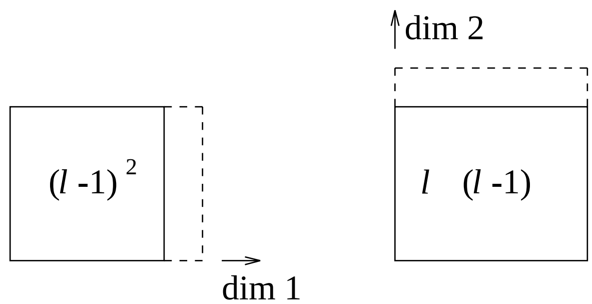


Fig. 2. Construction procedure for $\mathscr{C}_2(m)$.

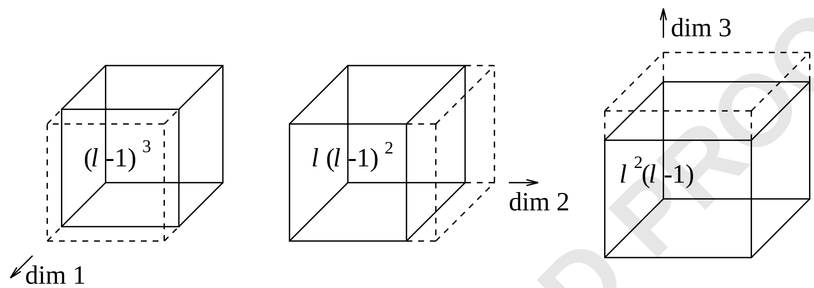


Fig. 3. Construction procedure for $\mathscr{C}_3(m)$.

Furthermore, there may be edges (legs) between adjacent layers in the subgraph. For any m, there must exist $l \geqslant 2$ and $1 \leqslant i \leqslant n$ such that $l^{i-1}(l-1)^{n-i+1} < m \leqslant l^i(l-1)^{n-i}$. Let $\delta = m - l^{i-1}(l-1)^{n-i+1}$. We give the following definition of a *cubish polyhedron*.

Definition 6. An n-D cubish polyhedron of m nodes in a k-ary n-cube, denoted as $\mathscr{C}_n(m)$, is defined recursively as follows:

- $\mathscr{C}_1(m)$ is a line of m adjacent nodes in the k-ary n-cube.
- $\mathscr{C}_n(m)$ contains an n-D mesh of size $\underbrace{l \times \cdots \times l}_{i-1} \times \underbrace{(l-1) \times \cdots \times (l-1)}_{n-i+1}$ with an $(n-1)$-D layer $\mathscr{C}_{n-1}(\delta)$ stacked on its top along dimension i. (Recall that $m = l^{i-1}(l-1)^{n-i+1} + \delta$.)

The above procedure of constructing $\mathscr{C}_n(m)$ is like making a ball of yarn. The idea is to fill in each side (dimension) with yarn (nodes), one side at a time. Fig. 2 illustrates the construction procedure for $\mathscr{C}_2(m)$, and Fig. 3 illustrates the procedure for $\mathscr{C}_3(m)$. Let $e_n(m)$ be the internal edge count in a cubish polyhedron $\mathscr{C}_n(m)$. Obviously, $e_n(m) = e_{n-1}(\delta) + \delta + e_n(m-\delta)$ according to the recursive definition of $\mathscr{C}_n(m)$. Note that the term δ in the equation is the number of legs between the $(n-1)$-D layer (with

$e_{n-1}(\delta)$ nodes) and the n-D mesh (with $e_n(m-\delta)$ nodes). The $(n-1)$-D layer stacked on top of the n-D mesh is referred to as the top layer. Obviously, the top layer has the fewest number of nodes among all layers along the chosen dimension i. For a cubish polyhedron, we can also define the bottom layer as the layer with nothing below it along the dimension. Obviously, the bottom layer (as well as the other non-top layers) has the most number of nodes.

We next prove that any cubish polyhedron $\mathscr{C}_n(m)$ is an edge isoperimetric subgraph of m nodes in a k-ary n-cube.

Theorem 7. *$\mathscr{C}_n(m)$ has the maximum internal edge count among all subgraphs S_m of m nodes in a k-ary n-cube, when the wrap-around edges can be discounted.*

Proof. We wish to prove that for any subgraph with m nodes, denoted as S_m, the number of internal edges, $e(S_m)$, is no larger than that of a cubish polyhedron, $e_n(m)$.

We prove the theorem by multiple inductions. First we induct on n, the number of dimensions. When $n=1$, The only way to achieve the maximum internal edge count is by placing all m nodes next to each other along the dimension, which is exactly the case in $\mathscr{C}_1(m)$. So for any subgraph S_m in a k-ary 1-cube,

$$e(S_m) \leqslant e_1(m).$$

In the inductive hypothesis, we assume that in any k-ary $(n-1)$-cube,

$$e(S_m) \leqslant e_{n-1}(m) \quad \text{(Hypothesis 1)}.$$

Now consider the case of n dimensions. The goal is to prove that for any S_m, $e(S_m) \leqslant e_n(m)$. We make another induction on m. When $m=1$, both $e(S_m)$ and $e_n(m)$ are 0. So

$$e(S_m) \leqslant e_n(m).$$

In the inductive hypothesis, we assume that for any $m' \leqslant m-1$,

$$e(S_{m'}) \leqslant e_n(m') \quad \text{(Hypothesis 2)}.$$

Now consider the case of m nodes. Let S_m be any subgraph in n dimensions. For any dimension, S_m can be viewed as having several $(n-1)$-D layers of nodes stacked on top of each other along the dimension. Choose the dimension with $h \geqslant l$ layers. (If all dimensions each has fewer than l layers, then $m \leqslant (l-1)^n$, which contradicts to our assumption that $l^{i-1}(l-1)^{n-i+1} < m \leqslant l^i(l-1)^{n-i}$.) To S_m and along the chosen dimension, we make the following rearrangement of the nodes:

- 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 Fig. 4 for an example of the rearrangement procedure described. (The numbers in the figure are the sizes of the layers.)

Let the subgraph obtained after the rearrangement procedure be S'_m. We know that in S'_m there are $h \geqslant l$ layers of $(n-1)$-D cubish polyhedrons. Assume that s_i is the size of the ith layer. Then $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_h$. By Hypothesis 1 and the fact that the

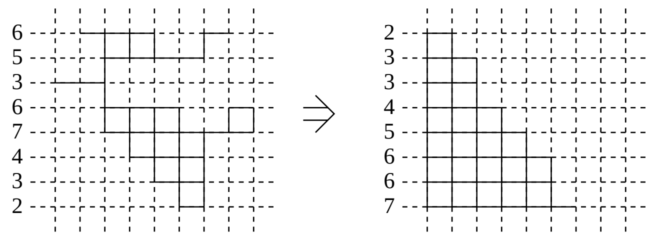


Fig. 4. Rearrangement procedure.

number of legs between adjacent layers is maximized when the layers are ordered by sizes, the rearrangement procedure does not decrease the number of internal edges in the subgraph. So we have

$$e(S_m) \leqslant e(S'_m). \tag{1}$$

Further, for S'_m, the number of internal edges is the sum of the number of edges in the first layer of s_1 nodes, the number of legs between the first and the second layers, and the number of internal edges in the subgraph of $m-s_1$ nodes containing the remaining $h-1$ layers, the last of which is bounded by $e_n(m-s_1)$ by Hypothesis 2. So we have

$$e(S'_m) \leqslant e_{n-1}(s_1) + s_1 + e_n(m-s_1). \tag{2}$$

To continue, we first consider the case when $s_1 \leqslant \delta$. Recall that $l^{i-1}(l-1)^{n-i+1} < m \leqslant l^i(l-1)^{n-i}$ and $\delta = m - l^{i-1}(l-1)^{n-i+1}$. Since $s_1 \leqslant \delta$, $l^{i-1}(l-1)^{n-i+1} \leqslant m - s_1 < l^i(l-1)^{n-i}$. Let $m - s_1 = l^{i-1}(l-1)^{n-i+1} + \delta'$. Then $\delta = \delta' + s_1$. By Definition 6, we have

$$e_n(m-s_1) = e_{n-1}(\delta') + \delta' + e_n(l^{i-1}(l-1)^{n-i+1}). \tag{3}$$

Also, let $S_{\delta'+s_1}$ be a subgraph in a k-ary $(n-1)$-cube, consisting of cubish polyhedrons $\mathscr{C}_{n-1}(\delta')$ and $\mathscr{C}_{n-1}(s_1)$ and some connecting edges in between. By Hypothesis 1, we have

$$\begin{aligned} e_{n-1}(\delta) &\geqslant e(S_{\delta'+s_1}) \\ &\geqslant e_{n-1}(\delta') + e_{n-1}(s_1). \end{aligned} \tag{4}$$

Therefore,

$$\begin{aligned} e(S_m) &\leqslant e(S'_m) \text{ by Eq. (1)} \\ &\leqslant e_{n-1}(s_1) + s_1 + e_n(m-s_1) \text{ by Eq. (2)} \\ &= e_{n-1}(s_1) + s_1 + e_{n-1}(\delta') + \delta' + e_n(l^{i-1}(l-1)^{n-i+1}) \text{ by Eq. (3)} \\ &= e_{n-1}(s_1) + e_{n-1}(\delta') + \delta + e_n(l^{i-1}(l-1)^{n-i+1}) \\ &\leqslant e_{n-1}(\delta) + \delta + e_n(l^{i-1}(l-1)^{n-i+1}) \text{ by Eq. (4)} \\ &= e_n(m) \text{ by Definition 6.} \end{aligned}$$

Next, we consider the case when $s_1 > \delta$. By Definition 6, we know that $\mathscr{C}_n(m-\delta)$ is in fact an n-D mesh with $l^{i-1}(l-1)^{n-i+1}$ nodes. $\mathscr{C}_n(m-\delta)$ can also be viewed as having l (or $l-1$ if $i=1$) layers stacked on top of each other, where each layer is an $(n-1)$-D mesh of L nodes. Clearly,

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

We can show that $s_1<L+\delta$. Suppose not. We must have $m=s_1+\cdots+s_h\geqslant hs_1\geqslant ls_1\geqslant lL+l\delta>lL+\delta\geqslant m$, which is impossible. Define $l'=l-1$ if $i=1$ and $l'=l$ if $i\geqslant 2$. Since $s_1<L+\delta$, together with the previous assumption that $s_1>\delta$ and the definition of $m=l'L+\delta$, it is easy to obtain that $(l'-1)L<m-s_1<l'L$. Let $m-s_1=(l'-1)L+\delta'$, where $\delta'<L$. Then $s_1+\delta'=L+\delta$. So by Definition 6,

$$e_n(m-s_1)=e_{n-1}(\delta')+\delta'+e_n((l'-1)L). \tag{5}$$

Therefore,

$$\begin{aligned}e(S_m)&\leqslant e(S'_m) \text{ by Eq. (1)}\\ &\leqslant e_{n-1}(s_1)+s_1+e_n(m-s_1) \text{ by Eq. (2)}\\ &= e_{n-1}(s_1)+s_1+e_{n-1}(\delta')+\delta'+e_n((l'-1)L) \text{ by Eq. (5).}\end{aligned}$$

On the other hand, we have

$$\begin{aligned}e_n(m)&=e_{n-1}(\delta)+\delta+e_n(l'L) \text{ by Definition 6}\\ &=e_{n-1}(\delta)+\delta+e_{n-1}(L)+L+e_n((l'-1)L) \text{ by Definition 6.}\end{aligned}$$

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

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

The inequality is trivially true when $s_1=L$. Next, we prove the inequality for two cases: $s_1<L$ and $s_1>L$. The ideas used for both the cases are similar. We start with two cubish polyhedrons in $n-1$ dimensions of node counts s_1 and δ', respectively. Obviously, the total number of internal edges in the two initial polyhedrons is $e_{n-1}(s_1)+e_{n-1}(\delta')$. Then we move nodes from one polyhedron to the other until the node counts of the polyhedrons become L and δ, respectively. Obviously, the total number of internal edges in the two final polyhedrons is $e_{n-1}(L)+e_{n-1}(\delta)$. If we can guarantee that the total number of internal edges of the polyhedrons does not decrease during the move. Then the inequality holds true.

Suppose $s_1<L$. We prove by yet another induction on the number of dimensions $n-1$ that $e_{n-1}(s_1)+e_{n-1}(\delta')\leqslant e_{n-1}(L)+e_{n-1}(\delta)$, 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 $n-2$ dimensions (Hypothesis 3). Now consider the case of $n-1$ dimensions. Since both s_1 and δ' are less than L, without loss of generality, assume that $s_1\geqslant\delta'$. (The case $s_1<\delta'$ is symmetric.) Next, we describe how to move nodes from $\mathscr{C}_{n-1}(\delta')$ to $\mathscr{C}_{n-1}(s_1)$. First, initialize subgraphs A and B to be $\mathscr{C}_{n-1}(s_1)$ and $\mathscr{C}_{n-1}(\delta')$, respectively. The node count in A, denoted as $|A|$, is then s_1, and the node count in B, denoted as $|B|$, is δ'.

Consider A as a cubish polyhedron of several $(n-2)$-D layers of size L' each plus one top layer of $a \leqslant L'$ nodes and a legs, and B as a cubish polyhedron of several $(n-2)$-D layers of size L'' each plus one top layer of $b \leqslant L''$ nodes and b legs. Since $|A| \geqslant |B|$, A completely contains B. So $L' \geqslant L''$. We next apply the following step to move nodes from B to A. If $b \leqslant L - |A|$, move the top layer $\mathscr{C}_{n-2}(b)$ together with its b legs and attach it to the bottom layer of A. we can do this because $b \leqslant L'' \leqslant L'$, which implies that the bottom layer of A is large enough to have the top layer of B attached without changing the remaining structure of the subgraph. After the move we rearrange the two subgraphs into cubish polyhedrons again. (Note that at this time A, B, L', L'', a, and b should be updated to reflect the resulting cubish polyhedrons.) It is clear that this step does not decrease the total edge count in the two subgraphs according to Hypothesis 1. That is,

$$e_{n-1}(s_1) + e_{n-1}(\delta') \leqslant e_{n-2}(a) + a + e_{n-1}(|A| - a) + e_{n-2}(b) + b + e_{n-1}(|B| - b). \tag{6}$$

We apply the above step until $|A| = L$ or $b > L - |A|$. If $|A| = L$ is the terminating condition, then A is already a cubish polyhedron with L nodes and B is thus a cubish polyhedron with $s_1 + \delta' - L = \delta$ nodes. So

$$\begin{aligned} &e_{n-1}(s_1) + e_{n-1}(\delta') \\ &\quad\leqslant e_{n-2}(a) + a + e_{n-1}(|A| - a) + e_{n-2}(b) + b + e_{n-1}(|B| - b) \text{ by Eq. (6)} \\ &\quad= e_{n-1}(|A|) + e_{n-1}(|B|) \text{ by Definition 6} \\ &\quad= e_{n-1}(L) + e_{n-1}(\delta). \end{aligned}$$

On the other hand, if $b > L - |A|$ is the terminating condition, which indicates that all non-top layers in B also have more than $L - |A|$ nodes, i.e., $L' > L - |A|$, we have to rearrange the nodes in the top layers of A and B. Since $|A| < L$ and $a \leqslant L'$, A needs exactly $L' - a$ nodes (the missing nodes in the top layer of A) to become an $(n-1)$-D mesh with only equal size layers. Therefore, $|A| + L' - a = L$. Combining this inequality with $b > L - |A|$, we have $a + b > (|A| + L' - L) + (L - |A|) = L'$. So a set of $a + b$ nodes can be split into a set of L' nodes and a set of $a + b - L'$ nodes. Since $a, b \leqslant L'$, $a + b = L' + (a + b - L')$, and L' is the size of a $(n-2)$-D mesh, by Hypothesis 3,

$$e_{n-2}(a) + e_{n-2}(b) \leqslant e_{n-2}(L') + e_{n-2}(a + b - L'). \tag{7}$$

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

$$\begin{aligned} &e_{n-1}(s_1) + e_{n-1}(\delta') \\ &\quad\leqslant e_{n-2}(a) + a + e_{n-1}(|A| - a) + e_{n-2}(b) + b + e_{n-1}(|B| - b) \text{ by Eq. (6)} \\ &\quad\leqslant e_{n-2}(L') + e_{n-2}(a + b - L') + a + b + e_{n-1}(|A| - a) + e_{n-1}(|B| - b) \end{aligned}$$

$$
\begin{aligned}
&\text{by Eq. (7)}\\
&= e_{n-2}(L') + L' + e_{n-1}(|A| - a) + e_{n-2}(a + b - L') + (a + b - L')\\
&\quad + e_{n-1}(|B| - b)\\
&= e_{n-1}(L) + e_{n-1}(\delta) \text{ by Definition 6.}
\end{aligned}
$$

Suppose $s_1 > L$. Let $s_1 = L + g$ for some g. Since $s_1 + \delta' = L + \delta$, then $\delta = s_1 + \delta' - L = g + \delta'$. We next show that $\delta' \geqslant (l-1)g$. Suppose not. We must have $m = l'L + \delta = l'L + g + \delta' < l'L + g + (l-1)g = l'L + lg \leqslant l(L+g) = ls_1 \leqslant hs_1 \leqslant m$, which is impossible. Consider the cubish polyhedron $\mathscr{C}_{n-1}(\delta')$ with $e_{n-1}(\delta')$ edges. If all $n-1$ dimensions of $\mathscr{C}_{n-1}(\delta')$ have l or more layers, then the number of nodes in $\mathscr{C}_{n-1}(\delta')$, $\delta' \geqslant l^{n-1} > L$, contradicting the fact that $\delta' < L$. So there must be a dimension in $\mathscr{C}_{n-1}(\delta')$ with $l-1$ or less layers. Along this dimension, we apply the rearrangement procedure described earlier in the proof. Since the procedure does not decrease the total internal edge count and the subgraph $\mathscr{C}_{n-1}(\delta')$ before the procedure is applied already has the maximum internal edge count according to Hypothesis 1, the number of edges in the resulting subgraph after the procedure is applied, which we call $S'_{\delta'}$, remains to be $e_{n-1}(\delta')$. For the bottom layer in $S'_{\delta'}$, which is the layer (also an $(n-2)$-D cubish polyhedron) with the most nodes since the layers are ordered during the rearrangement procedure, it must have $t \geqslant g$ nodes. Otherwise, δ', the numbers of nodes in $S'_{\delta'}$, would be less than $(l-1)g$, contradicting the fact that $\delta' \geqslant (l-1)g$. Now we are ready to move the nodes from $\mathscr{C}_{n-1}(s_1)$ to $S'_{\delta'}$. In $\mathscr{C}_{n-1}(s_1)$, since $s_1 = L + g$, along a certain dimension the top layer is a $\mathscr{C}_{n-2}(g)$. In $S'_{\delta'}$, along a certain dimension the bottom layer is a $\mathscr{C}_{n-2}(t)$ with $t \geqslant g$. Since $\mathscr{C}_{n-2}(g)$ is completely contained inside $\mathscr{C}_{n-2}(t)$, we move the layer $\mathscr{C}_{n-2}(g)$ together with its g legs from $\mathscr{C}_{n-1}(s_1)$ and attach it beneath the layer $\mathscr{C}_{n-2}(t)$. After the move, $\mathscr{C}_{n-1}(s_1)$ becomes $\mathscr{C}_{n-1}(L)$ with $e_{n-1}(L)$ edges and $S'_{\delta'}$ becomes a subgraph (not necessarily a cubish polyhedron) with $e_{n-1}(\delta') + e_{n-2}(g) + g$ edges. Note that the move does not change the total number of edges in the two subgraphs. So we have

$$
\begin{aligned}
e_{n-1}(s_1) + e_{n-1}(\delta') &= e_{n-1}(L) + e_{n-1}(\delta') + e_{n-2}(g) + g\\
&\leqslant e_{n-1}(L) + e_{n-1}(\delta' + g) \text{ by Hypothesis 1}\\
&= e_{n-1}(L) + e_{n-1}(\delta). \qquad \square
\end{aligned}
$$

4. Special cases

The theorem in the previous section describes what an edge isoperimetric subgraph for a k-ary n-cube looks like and how such a subgraph can be constructed. Although an algorithmic procedure can be easily designed to count the number of internal edges in an edge isoperimetric subgraph, the theorem does not tell us what the internal edge count is. Further, the theorem holds true only when the wrap-around edges in the k-ary n-cube can be discounted. In this section, we focus on formulas determining the number of internal edges of an edge isoperimetric subgraph (which may contain wrap-around edges) of m nodes in any k-ary n-cube. In particular, we first consider one important special case of k-ary n-cubes, namely, hypercubes. We then study the case of any k-ary 2-cube, which is in fact a 2-D torus. Finally, we focus on computing the number of internal edges of an edge isoperimetric subgraph in a k-ary n-cube with high dimensions.

4.1. Hypercubes

To compute $e_{2,n}(m)$, the maximum number of internal edges of a subgraph with m nodes in a hypercube (in n dimensions), we have to do some preliminary work.

Definition 8. Let $w(i)$ denote the sum of all bits in the base-2 (binary) representation of i. Let $W(i,j)$, $i \leqslant j$, denote the sum of $w(i),\ldots,w(j)$.

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

Definition 9. Define function F recursively as follows:

$$F(0) = F(1) = 0,$$
$$F(m) = F\left(\left\lceil \frac{m}{2} \right\rceil\right) + F\left(\left\lfloor \frac{m}{2} \right\rfloor\right) + \left\lfloor \frac{m}{2} \right\rfloor \quad \text{for } m \geqslant 2.$$

Lemma 10. *$F(m) = W(0, m-1)$ for $m \geqslant 1$.*

Proof. We prove the lemma by showing that $W(0,m-1)$ indeed satisfies the recursion that defines $F(m)$. This can be done by the following counting argument. By Definition 8, $W(0,m-1)$ represents the sum of all bits in the binary representations of integers $0,1,\ldots,m-1$. We display all the binary numbers $0,1,\ldots,m-1$ in two columns: one containing numbers whose last digit is 0 and the other containing numbers whose last digit is 1. The first column has $\lceil m/2 \rceil$ numbers and the second column has $\lfloor m/2 \rfloor$ numbers. We observe that the sum of the bits of the numbers in the first column is $W(0,\lceil m/2 \rceil - 1)$ and that the sum of the bits of the numbers in the second column is $W(0,\lfloor m/2 \rfloor - 1) + \lfloor m/2 \rfloor$. Therefore,

$$W(0, m-1) = W\left(0, \left\lceil \frac{m}{2} \right\rceil - 1\right) + W\left(0, \left\lfloor \frac{m}{2} \right\rfloor - 1\right) + \left\lfloor \frac{m}{2} \right\rfloor.$$

The recursive definition matches the one that defines $F(m)$. So $F(m) = W(0, m-1)$. □

Lemma 11. *$F(m) \geqslant F(m_0) + F(m_1) + \min\{m_0, m_1\}$ for $m_0 + m_1 = m$.*

Proof. We induct on m. When $m = 0, 1$, the inequality holds obviously. Suppose that the inequality holds true for cases $\leqslant m-1$. Now consider the case of m.

When $m = m_0 + m_1$ is even, we have $m/2 = \lceil m_0/2 \rceil + \lfloor m_1/2 \rfloor = \lfloor m_0/2 \rfloor + \lceil m_1/2 \rceil$:

$$
\begin{aligned}
F(m) &= F\left(\frac{m}{2}\right) + F\left(\frac{m}{2}\right) + \frac{m}{2} \quad \text{by Definition 9} \\
&\geqslant F\left(\left\lceil \frac{m_0}{2} \right\rceil\right) + F\left(\left\lfloor \frac{m_1}{2} \right\rfloor\right) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} \\
&\quad + F\left(\left\lfloor \frac{m_0}{2} \right\rfloor\right) + F\left(\left\lceil \frac{m_1}{2} \right\rceil\right) + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lceil \frac{m_1}{2} \right\rceil\right\} + \frac{m}{2} \\
&\quad \text{by the inductive hypothesis} \\
&= F(m_0) + F(m_1) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lceil \frac{m_1}{2} \right\rceil\right\} \\
&\quad + \frac{m}{2} - \left\lfloor \frac{m_0}{2} \right\rfloor - \left\lfloor \frac{m_1}{2} \right\rfloor \quad \text{by Definition 9} \\
&\geqslant F(m_0) + F(m_1) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lceil \frac{m_1}{2} \right\rceil\right\} \\
&= F(m_0) + F(m_1) + \min\{m_0, m_1\}.
\end{aligned}
$$

When $m = m_0 + m_1$ is odd, we have $\lceil m/2 \rceil = \lceil m_0/2 \rceil + \lceil m_1/2 \rceil$ and $\lfloor m/2 \rfloor = \lfloor m_0/2 \rfloor + \lfloor m_1/2 \rfloor$:

$$
\begin{aligned}
F(m) &= F\left(\left\lceil \frac{m}{2} \right\rceil\right) + F\left(\left\lfloor \frac{m}{2} \right\rfloor\right) + \left\lfloor \frac{m}{2} \right\rfloor \quad \text{by Definition 9} \\
&\geqslant F\left(\left\lceil \frac{m_0}{2} \right\rceil\right) + F\left(\left\lceil \frac{m_1}{2} \right\rceil\right) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lceil \frac{m_1}{2} \right\rceil\right\} \\
&\quad + F\left(\left\lfloor \frac{m_0}{2} \right\rfloor\right) + F\left(\left\lfloor \frac{m_1}{2} \right\rfloor\right) + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} + \left\lfloor \frac{m}{2} \right\rfloor \\
&\quad \text{by the inductive hypothesis} \\
&= F(m_0) + F(m_1) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lceil \frac{m_1}{2} \right\rceil\right\} + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} \\
&\quad + \left\lfloor \frac{m}{2} \right\rfloor - \left\lfloor \frac{m_0}{2} \right\rfloor - \left\lfloor \frac{m_1}{2} \right\rfloor \quad \text{by Definition 9} \\
&= F(m_0) + F(m_1) + \min\left\{\left\lceil \frac{m_0}{2} \right\rceil, \left\lceil \frac{m_1}{2} \right\rceil\right\} + \min\left\{\left\lfloor \frac{m_0}{2} \right\rfloor, \left\lfloor \frac{m_1}{2} \right\rfloor\right\} \\
&= F(m_0) + F(m_1) + \min\{m_0, m_1\}. \qquad \square
\end{aligned}
$$

Lemma 12. $F(m) = \frac{1}{2} m \log_2 m$ *if* $m = 2^l$ *for some* l.

Proof. Use Definition 9 and induct on m. $\square$

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

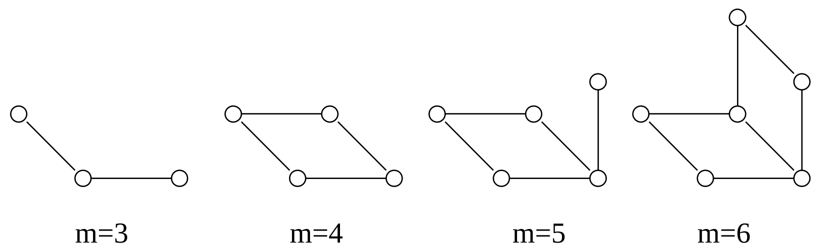


Fig. 5. Subgraphs of a hypercube achieving internal edge count $F(m)$.

Theorem 13. $e_{2,n}(m) = F(m)$.

Proof. Since a hypercube of n dimensions contains two composite subcubes, each of which is a hypercube of $n-1$ dimensions, assume that m_0 and m_1 nodes are chosen in the 0th and 1st composite subcubes, respectively. By Proposition 5,

$$e_{2,n}(0) = e_{2,n}(1) = 0,$$
$$e_{2,n}(m) \leqslant \max_{\forall \sum m_i = m} \{e_{2,n-1}(m_0) + e_{2,n-1}(m_1) + \min\{m_0, m_1\}\}.$$

First, we prove by induction on m that $e_{2,n}(m) \leqslant F(m)$. When $m = 0, 1$, $e_{2,n}(m) = F(m) = 0$. Assume that the inequality holds for cases $\leqslant m-1$. Now consider the case of m:

$$\begin{aligned} e_{2,n}(m) &\leqslant \max_{\forall \sum m_i = m} \{e_{2,n-1}(m_0) + e_{2,n-1}(m_1) + \min\{m_0, m_1\}\} \\ &\leqslant \max_{\forall \sum m_i = m} \{F(m_0) + F(m_1) + \min\{m_0, m_1\}\} \\ &\quad \text{by the inductive hypothesis} \\ &\leqslant F(m) \text{ by Lemma 11.} \end{aligned}$$

Next, we prove that there exists 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 $\lceil m/2 \rceil$ nodes into the 0th composite subcube and $\lfloor m/2 \rfloor$ nodes into the 1st 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 $\lfloor m/2 \rfloor$ edges between the two pieces, which are themselves optimal with respect to their sizes. Fig. 5 illustrates optimal subgraphs of a hypercube for $m = 3$–6. Note that these subgraphs are also cubish polyhedrons as defined in the previous section, matching the result proved in Theorem 7.

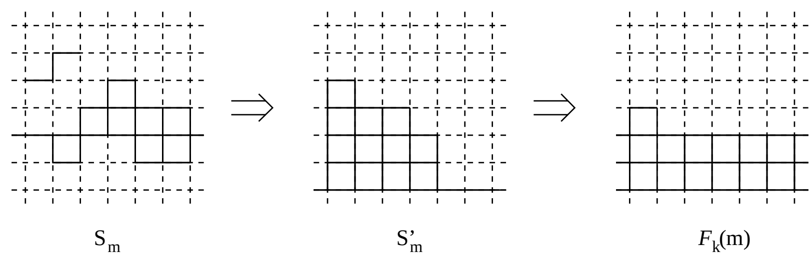


Fig. 6. Rearrangement procedure from S_m to $\mathscr{F}_k(m)$.

4.2. 2-D tori

The edge isoperimetric subgraph in a 2-D torus (k-ary 2-cube) may or may not contain wrap-around edges. If it has no wrap-around edges, by Theorem 7 it must be a cubish polyhedron, which in the 2-D plane becomes a squarish polygon that grows like a swirl as nodes are added. If the edge isoperimetric subgraph contains wrap-around edges, it must be a flat polygon containing as many wrap-around edges as possible. We explain what this means.

Let m be the number of nodes in a subgraph S_m with at least one wrap-around edge. If we divide the node set into layers along any one of the two dimensions, rearrange the layers of nodes so that they are ordered by sizes (node counts in the layers), and place nodes in each layer as closely to each other as possible, we obtain a new subgraph S'_m of m nodes. This rearrangement procedure can be depicted by the first step in Fig. 6. Clearly, $e(S_m) \leqslant e(S'_m)$. In subgraph S'_m, the bottommost layer is full, i.e., it contains all k nodes and one wrap-around edge. We continue moving nodes from the topmost layer to the bottommost layer that is not full, one by one and without decreasing the total number of internal edges in the subgraph, until there is at most one layer at the top that is not full. We call the subgraph obtained a *flat polyhedron*, denoted as $\mathscr{F}_k(m)$. If $m = xk + y$ for $0 \leqslant y \leqslant m-1$, then $\mathscr{F}_k(m)$ has x full layers of k nodes each and an additional layer of y nodes stacked at the top. This rearrangement procedure can be depicted by the second step in Fig. 6. Clearly, $e(S'_m) \leqslant e(\mathscr{F}_k(m))$, where $e(\mathscr{F}_k(m))$ is the number of internal edges in $\mathscr{F}_k(m)$. So we have the following lemma.

Lemma 14. *$\mathscr{F}_k(m)$ has the maximum internal edge count among all subgraphs S_m of m nodes with at least one wrap-around edge in a 2-D torus.*

Proof. By the rearrangement procedure illustrated in Fig. 6, any subgraph S_m with at least one wrap-around edge can be transformed, without decreasing the internal edge count, to a subgraph S'_m with all the layers ordered by sizes and nodes put next to each other within each layer. The subgraph S'_m can then be transformed, again without decreasing the internal edge count, to a 2-D flat polyhedron which contains the most full layers for the given m. So we have

$$e(S_m) \leqslant e(\mathscr{F}_k(m)). \qquad \square$$

Since the edge isoperimetric subgraph of m nodes in a 2-D torus may be a cubish polyhedron $\mathscr{C}_2(m)$ or a 2-D flat polyhedron $\mathscr{F}_k(m)$, depending on which has more internal edge count, we next focus on the exact internal edge counts of the two structures.

Lemma 15. *$e(\mathscr{C}_2(m)) = \lfloor 2m - 2\sqrt{m} \rfloor$, where $e(\mathscr{C}_2(m))$ (equivalent to $e_2(m)$ used in Section 3) is the number of internal edges in $\mathscr{C}_2(m)$.*

Proof. Note that a cubish polyhedron is defined in Section 3 under the assumption that all wrap-around edges are discounted. Therefore, $\mathscr{C}_2(m)$ does not include any wrap-around edges. We consider four cases.

If $m = l^2$ for some integer l, then $\mathscr{C}_2(m)$ is a square mesh of size $l \times l$. So $e(\mathscr{C}_2(m)) = 2l(l-1) = 2l^2 - 2l = 2m - 2\sqrt{m} = \lfloor 2m - 2\sqrt{m} \rfloor$.

If $m = l(l-1)$ some integer l, then $\mathscr{C}_2(m)$ is a mesh of size $l \times (l-1)$. So $e(\mathscr{C}_2(m)) = (l-1)^2 + l(l-2) = 2l^2 - 4l + 1 = 2m - (2l-1) = \lfloor 2m - 2\sqrt{m} \rfloor$ since $2l - 2 < 2\sqrt{m} < 2l - 1$.

If $l(l-1) < m < l^2$ for some integer l, then $\mathscr{C}_2(m)$ is a mesh of size $l \times (l-1)$ plus one layer of $m - l(l-1)$ nodes. So $e(\mathscr{C}_2(m)) = (l-1)^2 + l(l-2) + 2(m - l(l-1)) - 1 = 2m - 2l = \lfloor 2m - 2\sqrt{m} \rfloor$ since $2l - 1 < 2\sqrt{m} < 2l$.

If $l^2 < m < l(l+1)$ for some integer l, then $\mathscr{C}_2(m)$ is a square mesh of size $l \times l$ plus one layer of $m - l^2$ nodes. So $e(\mathscr{C}_2(m)) = 2l(l-1) + 2(m - l^2) - 1 = 2m - 2l - 1 = \lfloor 2m - 2\sqrt{m} \rfloor$ since $2l < 2\sqrt{m} < 2l + 1$. □

We want to point out that Lemma 15 above matches the result in [9], which gives the exactly same formula of $\lfloor 2m - 2\sqrt{m} \rfloor$ for infinite 2-D meshes.

Lemma 16.

$$e(\mathscr{F}_k(m)) = \begin{cases} m - 1 & if\ m < k, \\ 2m - k & if\ m = xk\ for\ 1 \leqslant x \leqslant k-1, \\ 2m - k - 1 & if\ m = xk + y\ for\ 1 \leqslant x \leqslant k-2 \\ & \ 1 \leqslant y \leqslant k-1, \\ 2m - k - 1 + (m \bmod k) & if\ m = xk + y\ for\ x = k-1 \\ & \ 1 \leqslant y \leqslant k-1, \\ 2m & if\ m = k^2\ (maximum\ node\ count). \end{cases}$$

Proof. The proof contains a simple count of edges in $\mathscr{F}_k(m)$ for each of the five cases defined in the lemma. See Fig. 7 for the example of $k = 4$. □

Combining the lemmas above, we have the following theorem that gives the exact internal edge count for the edge isoperimetric subgraph of m nodes in a 2-D torus.

Theorem 17. *$e_{k,2}(m) = \max\{e(\mathscr{C}_2(m)), e(\mathscr{F}_k(m))\}$, where the formulas for $e(\mathscr{C}_2(m))$ and $e(\mathscr{F}_k(m))$ are given in Lemmas 15 and 16.*

DAM 2954
ARTICLE IN PRESS

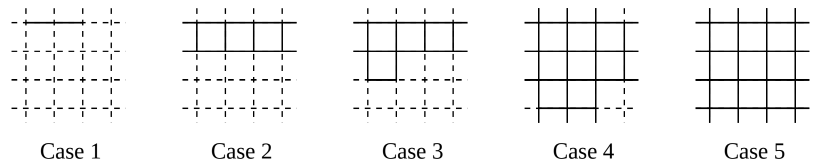


Fig. 7. Counting edges in $\mathscr{F}_4(m)$.

Proof. The theorem follows immediately from the discussion in this subsection. $\square$

4.3. Subgraphs in k-ary n-cubes of high dimensions

We now consider another special case of k-ary n-cubes and show how to compute the maximum internal edge count of a subgraph. We make the following assumptions: $k \geqslant 4$ and $n \geqslant \log m$ (or equivalently, $m \leqslant 2^n$). We observe that the special case under the assumptions is rather general since it includes a large class of different k-ary n-cubes. In what follows, we show that $F(m)$, the function defined in Section 4.1, again captures the quantity of interest.

Lemma 18. *$F(m) \geqslant \sum_{i=0}^{k-1} F(m_i) + m - \max_{0\leqslant i\leqslant k-1}\{m_i\} + \min_{0\leqslant i\leqslant k-1}\{m_i\}$ for $\sum_{i=0}^{k-1} m_i = m$ and $k \geqslant 4$.*

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

$$F(m) \geqslant F\left(\sum_{i=0}^{l} m_i\right) + F\left(\sum_{i=l+1}^{k-1} m_i\right) + \min\left\{\sum_{i=0}^{l} m_i, \sum_{i=l+1}^{k-1} m_i\right\} \text{ by Lemma 11}$$

$$\geqslant \sum_{i=0}^{k-1} F(m_i) + A + B + C \text{ by Lemma 11 repeatedly,}$$

where

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

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

and

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

Next, we wish to prove that $A+B+C \geqslant m-m_0+m_{k-1}$. Since $\sum_{i=0}^{l} m_i \geqslant m/2$, $A=\sum_{i=l+1}^{k-1} m_i$. Since $l<k/2$ and $k \geqslant 4$, $l+1 \leqslant k-2$. So there is at least one term in C. Therefore, $C \geqslant m_{k-1}$. How large is B? If $l=0$, then $B=0$ and $A+B+C \geqslant \sum_{i=1}^{k-1} m_i + m_{k-1} = m-m_0+m_{k-1}$. If $l=1$, then $B=m_1$ and $A+B+C \geqslant \sum_{i=2}^{k-1} m_i + m_1 + m_{k-1} = m-m_0+m_{k-1}$. Now assume that $l \geqslant 2$. B must have at least two terms. If $m_h \leqslant \sum_{i=h+1}^{l} m_i$ for all $h=0,\ldots,l-2$, then $B=\sum_{i=0}^{l-2} m_i + m_l$ and $A+B+C \geqslant \sum_{i=l+1}^{k-1} m_i + \sum_{i=0}^{l-2} m_i + m_l + m_{k-1} \geqslant m-m_0+m_{k-1}$. 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 $B \geqslant \sum_{i=0}^{h-1} m_i + \sum_{i=h+1}^{l} m_i$ and $A+B+C \geqslant \sum_{i=l+1}^{k-1} m_i + \sum_{i=0}^{h-1} m_i + \sum_{i=h+1}^{l} m_i + m_{k-1} \geqslant m-m_0+m_{k-1}$. □

Theorem 19. *$e_{k,n}(m)=F(m)$ for $k \geqslant 4$ and $n \geqslant \log m$.*

Proof. Since a k-ary n-cube contains k composite subcubes, each of which is a k-ary $(n-1)$-cube, assume that m_i nodes are chosen in the ith composite subcube for $0 \leqslant i \leqslant k-1$. By Proposition 5,

$$e_{k,n}(0)=e_{k,n}(1)=0,$$

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

Similar to Theorem 13, we can prove by induction on m that $e_{k,n}(m) \leqslant F(m)$, using the above recursive definition of $e_{k,n}(m)$, the inductive hypothesis, and Lemma 18.

Also similar to Theorem 13, a subgraph S_m of $m \leqslant 2^n$ nodes with $F(m)$ internal edges can be constructed by allocating $\lceil m/2 \rceil$ nodes into the 0th composite subcube and $\lfloor m/2 \rfloor$ nodes into the 1st composite subcube; the same method is then used recursively to allocate the nodes in each composite subcube. □

5. Applications to graph partitioning

The problem of partitioning graphs for parallel processing is studied extensively [4,12,15,16]. In a k-ary n-cube that represents a parallel program, nodes are tasks with node weights representing computation costs, and edges are message-passing links between tasks with edge weights representing communication costs [8,11]. Recall that for any subgraph, an internal edge is one with two endpoints in the subgraph and an external edge is one with one endpoint in the subgraph. Viewing the subgraph as the set of nodes (tasks) assigned to a processor, the sum of weights on external edges is a measure of the communication cost between processors. (Note that the internal edge weights are usually discounted since the communications they represent all happen within one processor and are considered free.) The *load* of a subgraph is defined to be the sum of the weights of its nodes and its external edges. If a k-ary n-cube is partitioned into P subgraphs, then the *bottleneck cost* of the partition is the maximum load among all subgraphs in the partition. The graph partitioning problem is to find a partition that minimizes the bottleneck cost.

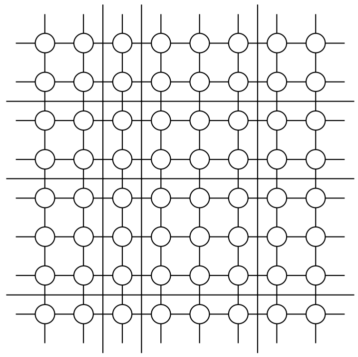


Fig. 8. Rectilinear partition of an 8×8 torus.

The applications of the edge isoperimetric property to graph partition are summarized in [3]. Here, we add two more applications. First, our results on edge isoperimetric subgraphs for k-ary n-cubes may be used in the context of branch-and-bound algorithms for graph partitioning. Our object here is neither to propose the specifics of such an algorithm neither to 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 illustrate its use in one specific case. Second, our theoretical results can also be used to show the optimality of some curiously shaped partitions. An example of this application is shown.

5.1. Lower bounding in branch-and-bound algorithms

Let us consider the *rectilinear* partitioning [13] in a k-ary n-cube graph, where the separating cuts that define the partition are all hyperplanes of the form $x_i = c_{ij}$, a constant. A rectilinear partition of an 8×8 torus is illustrated in Fig. 8.

Since the problem of rectilinear partitioning is generally intractable, branch-and-bound algorithms [6] may be used to find an optimal partitioning within a reasonable amount of time. The key in a branch-and-bound algorithm is the construction of a search tree. For rectilinear partitioning, a node in the branch-and-bound search tree reflects a set of cuts already made, where the root reflects an empty cut set. 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 the 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. Function $bnd(N)$ 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 $bnd(N)$ which provides a tight lower bound.

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 (node counts) of subregions $m_1, m_2, \ldots, m_s$, for $\sum_{i=1}^{s} m_i = r$, such that $\sum_{i=1}^{s} C(m_i)$ is minimized, where $C(m_i)$ is the external edge count (cost) of an edge isoperimetric 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 $k \geqslant 3$, we have that $C(m_i) = dm_i - 2e_{k,n}(m_i)$. Solution to this minimization problem even when modified to include a constraint $m_i \leqslant B$ for all i, is straightforward using dynamic programming.

The 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 ith component w_i is the sum of the weights of the first $C(m_i)$ edges in E. Vector $\mathbf{w}$ reflects lower bounds on communication costs under assignment $\mathbf{m}$. Without loss of generality, suppose that w_1 is the largest component. We define the *slack* of $\mathbf{w}$ as

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

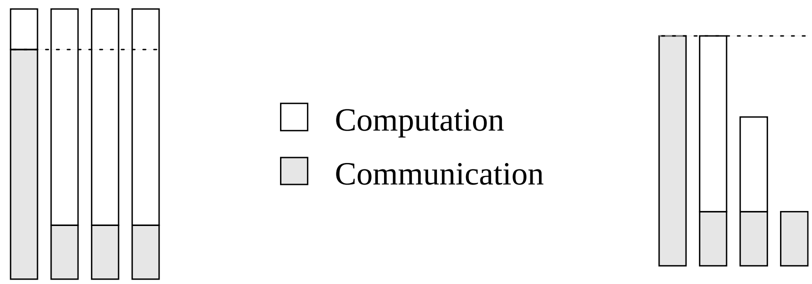
Third, we consider the following two cases.

The first case of interest is when $slack(\mathbf{w}) \leqslant W_R$. 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 remnant may be divided evenly among the s processors. This is illustrated in Fig. 9(a). So

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

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 of interest occurs when $slack(\mathbf{w}) > W_R$, as illustrated by Fig. 9(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 also decrease the bottleneck cost. To reduce the bottleneck cost we constrain the assignment $m_i \leqslant B$ 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



(a) (b)

Fig. 9. Computation of lower bound on bottleneck cost. (a) Slack is less than total computation and (b) Slack exceeds total computation.

the corresponding slack exceeds W_R. Let $\mathbf{w} = (w_1, \dots, w_s)$ and $\mathbf{w}' = (w'_1, \dots, w'_s)$ be the weight vectors derived from using $B^* - 1$ and B^* as constraints, respectively. Then we make the lower bound to be

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

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

The procedure above shows how to bound from below the potential least bottleneck cost for each region reflected by node N in the branch-and-bound search tree. 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 partition whose bottleneck cost is the 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, with a bottleneck cost of $4 \times 32 + 8 = 136$. However, the graph that weights one node by 128 and all other nodes by $\frac{128}{63}$ is optimally partitioned by isolating the heavy node, with a bottleneck 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.

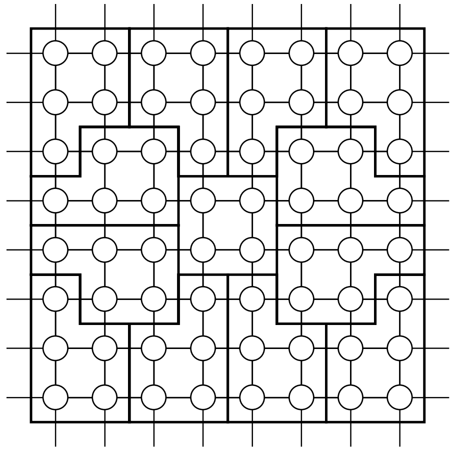


Fig. 10. Optimal partition of an 8-ary 2-cube into 13 subgraphs.

5.2. *Identifying 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 an 8-ary 2-cube (an 8×8 torus) into 13 subgraphs, assuming that all nodes have common computation weight w_1 and all edges have common communication weight w_2. The problem clearly does not divide evenly. The minimal load of a subgraph of m nodes assigned to a processor is $w_1 m + w_2 C(m)$, where $C(m)$ is the minimum external edge count of a subgraph with m nodes. Since the 8-ary 2-cube is a regular graph with degree 4 for each node and the maximum internal edge count of a subgraph with m nodes is $e_{8,2}(m)$, which can be computed constructively according to our main theorem in Section 3, we then have $C(m) = 4m - 2e_{8,2}(m)$. Note that the C 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 (which is an edge isoperimetric subgraph) of the 8-ary 2-cube with 5 nodes is a square of 4 nodes, with an attached singleton node. As illustrated in Fig. 10, it is possible to nearly tessellate the 8-ary 2-cube with this optimal subgraph, the only exception being one subgraph (the center square) which is itself an optimal subgraph of 4 nodes. The optimality of this partition derives from the fact that $w_1 m + w_2 C(m)$ is monotonically non-decreasing in m, so that the bottleneck cost $\max\{w_1 m_1 + w_2 C(m_1), \ldots, w_1 m_{13} + w_2 C(m_{13})\}$ is minimized when the m_i's are nearly equal. The partition shown achieves the lower bound of $5w_1 + C(5)w_2 = 5w_1 + 10w_2$.

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 monotonically non-decreasing as a function of the number of nodes assigned to it, $w_1 m + w_2 C(m)$ 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 $w_1 m + w_2 C(m)$, then that partition is optimal.

6. Conclusions

In this paper, we have studied combinatorial properties of k-ary n-cube graphs. The class of k-ary n-cubes reflect common parallel processing architectures as well as communication patterns. In addition, its special cases includes some widely used topologies such as rings, hypercubes, and tori. Because of its importance to parallel processing and communication, our study of the class has been not only necessary but also rewarding.

One combinatorial aspect we have examined in detail is the edge isoperimetric property of k-ary n-cubes. We are particularly interested in the structure of the subgraph of a fixed node count with the maximum number of internal edges. In the paper, we have proved that any subgraph whose structure is that of a “cubish polyhedron” achieves such maximum under the assumption that wrap-around edges are discounted. For three special cases of k-ary n-cubes, we also have given simple formulas that compute easily the maximum.

While the above results have combinatorial interest, they also have serious applications to problems in parallel processing. We have shown, 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 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 have also shown how our results can be used to demonstrate the optimality of certain irregular partitions.

The 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.

Acknowledgements

The authors wish to thank the referees for their helpful comments and constructive suggestions.

References

[1] R. Ahlswede, S.L. Bezrukov, Edge isoperimetric theorems for integer point arrays, Appl. Math. Lett. 8 (1995) 75–80.

[2] Y. Ashir, I. Stewart, A. Ahmed, Communication algorithms in k-ary n-cube interconnection networks, Inform. Process. Lett. 61 (1997) 43–48.

[3] S.L. Bezrukov, Edge isoperimetric problems on graphs, in: L. Lovasz, A. Gyarfas, G.O.H. Katona, A. Recski, L. Szekely (Eds.), Graph Theory and Combinatorial Biology, American Mathematical Society, Providence, RI, 1999, pp. 157–197.

[4] S.H. Bokhari, Partitioning problems in parallel, pipelined, and distributed computing, IEEE Trans. Comput. 37 (1988) 48–57.
[5] B. Bollobás, I. Leader, Edge-isoperimetric inequalities in the grid, Combinatorica 11 (1991) 299–314.
[6] G. Brassard, P. Bratley, Algorithms: Theory and Practice, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[7] W.J. Dally, Performance analysis of k-ary n-cube interconnection networks, IEEE Trans. Comput. 39 (1990) 775–785.
[8] P. Dickens, P. Heidelberger, D.M. Nicol, A distributed memory LAPSE: parallel simulation of message-passing programs, in: Proceedings of the Eighth Workshop on Parallel and Distributed Simulation, 1994, pp. 32–38.
[9] F. Harary, H. Harborth, Extremal animals, J. Combin. Inform. System Sci. 1 (1976) 1–8.
[10] L.H. Harper, Optimal assignment of numbers to vertices, SIAM J. Appl. Math. 12 (1964) 131–135.
[11] C. McCann, J. Zahorjan, Processor allocation policies for message-passing parallel computers, in: Proceedings of the 1994 SIGMETRICS Conference, 1994, pp. 19–32.
[12] G. Miller, S.-H. Teng, W. Thurston, S.A. Vavasis, Automatic mesh partitioning, in: Graph Theory and Sparse Matrix Computation, Springer, Berlin, 1993.
[13] D.M. Nicol, Rectilinear partitioning of irregular data parallel computations, J. Parallel Distributed Comput. 23 (1994) 119–134.
[14] D.M. Nicol, W. Mao, On bottleneck partitioning of k-ary n-cubes, Parallel Process. Lett. 6 (1996) 389–399.
[15] A. Pothen, H.D. Simon, K.P. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl. 11 (1990) 430–452.
[16] D.A. Reed, L.M. Adams, M.L. Patrick, Stencils and problem partitionings: their influence on the performance of multiple processor systems, IEEE Trans. Comput. 36 (1987) 845–858.