Journal of Graph Algorithms and Applications
http://jgaa.info/
vol. 7, no. 1, pp. 79-98 (2003)

Hamilton Decompositions and $(n/2)$-Factorizations of Hypercubes

Douglas W. Bass
Graduate Programs in Software
University of St. Thomas
http://gps.stthomas.edu
dbass@stthomas.edu

I. Hal Sudborough
Department of Computer Science
University of Texas at Dallas
http://www.utdallas.edu/dept/cs
hal@utdallas.edu

Abstract

Since Q_n, the hypercube of dimension n, is known to have n link-disjoint paths between any two nodes, the links of Q_n can be partitioned into multiple link-disjoint spanning subnetworks, or factors. We seek to identify factors which efficiently simulate Q_n, while using only a portion of the links of Q_n. We seek to identify $(n/2)$-factorizations, of Q_n where 1) the factors have as small a diameter as possible, and 2) mappings (embeddings) of Q_n to each of the factors exist, such that the maximum number of links in a factor corresponding to one link in Q_n (dilation), is as small as possible. In this paper we consider two algorithms for generating Hamilton decompositions of Q_n, and three methods for constructing $(n/2)$-factorizations of Q_n for specific values of n. The most notable $(n/2)$-factorization of Q_n results in two mutually isomorphic factors, each with diameter $n+2$, where an embedding exists which maps Q_n to each of the factors with constant dilation.

Communicated by Balaji Raghavachari: submitted November 2001;
revised September 2002 and February 2003.

1 Introduction

1.1 Traditional Partitioning of Hypercubes

In traditional computing environments with a single processor, a large number of memory locations and multiple users, memory is a partitionable resource. Processes make requests for certain amounts of contiguous memory locations, and various allocation and collection strategies are used to minimize memory fragmentation.

In an environment with 2^n processors, connected in the form of the hypercube Q_n, and multiple users, the body of processors is also a partitionable resource. However, the hypercube is not partitioned in the same way as memory, as Q_n has a recursive substructure. Q_n consists of 2 copies of Q_{n-1}, with links connecting corresponding nodes in the two copies. When processes make requests for some of the processors, they traditionally request a complete hypercube of dimension smaller than n, known as a subcube. Significant research has been conducted into identifying, allocating, and recollecting subcubes in order to minimize subcube fragmentation [13, 19].

In this environment, if two or more processes require Q_n, only one of them can run at any given time. We seek to take advantage of the node-connectivity of Q_n, to increase the effective capacity of a hypercube-based computing environment, so that two processes requiring Q_n can run concurrently.

1.2 Definitions

A *network* G, is a pair (N, L), where N is a set of distinct *nodes*, and L is a set of *links*. L is a set of two element subsets of N. In a network, the *degree* of a node n is the number of elements of L containing n as an element. A network is *regular* if every node $n \in N$ has the same degree. The *degree* of a regular network is the degree of any node $n \in N$. A *path* is a sequence of nodes $n_1, n_2, \ldots, n_k$, such that $\forall\ i,\ 1 \leq i \leq k$ - 1, $\{n_i, n_{i+1}\} \in L$. A network is *connected* if for all pairs of nodes u and v, there exists a path from u to v. The *node-connectivity* of a network G is the minimal number of nodes which must be removed from G, in order to make G no longer connected.

The *hypercube of dimension n*, or Q_n, is a network of 2^n nodes where each node is labeled by a bit string $b_0\ b_1 \ldots b_{n-1}$ of length n, and there is a link between two nodes in Q_n if and only if their labels differ in exactly one bit. Q_n is regular with degree n, and has $n2^{n-1}$ links. A *dimension k link* is a link in Q_n which connects two nodes whose labels differ in the k^{th} bit.

A *Hamilton cycle* of a network is a path $n_1, n_2, \ldots, n_k, n_1$, which visits each node in the network exactly once, and returns to its starting point n_1. A *Hamilton decomposition* of a network is a partitioning of the links of the network into link-disjoint Hamilton cycles [2]. A *matching* in a network is a set of node-disjoint links. A matching is *orthogonal* to a set of Hamilton cycles if it contains one and only one link from each Hamilton cycle. The *Cartesian product* $N_1 \times N_2$ of two networks N_1 and N_2 is the network where the nodes are ordered pairs of the nodes of N_1 and N_2, and there are links between $\{u, v\}$ and $\{w, x\}$ in $N_1 \times N_2$ if $\{u, w\}$ is a link in N_1 or $\{v, x\}$ is a link in N_2. C_n represents the simple cycle of n nodes.

A *spanning subnetwork S of a network N* is a connected network constructed from all the nodes of N and a proper subset of the links of N, such that for every pair of nodes u and v in N, there is a path in S between u and v. A *perfect matching* in a network is a matching where every node in the network is incident to exactly one link. A *k-factorization* of a network is a partitioning of the links of the network into disjoint regular spanning subnetworks, or factors, of degree k [3]. The *distance* between two vertices u and v in a network G, denoted by $distance_G(u, v)$, is the length of the shortest path between u and v in G. The *diameter* of a network G, denoted by $diameter(G)$, is the maximum value of $\text{distance}_G(u, v)\ \forall\ u, v \in N$.

An *embedding* of a network G (commonly called the *guest*) into a network H (commonly called the *host*) is a 1-1 function f mapping the nodes of G to the nodes of H. When G and H have the same set of nodes, the *identity embedding* I is the embedding $I(u) = u\ \forall\ u$ in G. The *dilation* of an embedding f is the largest value of $\text{distance}_H(f(u), f(v))$, $\forall$ edges $\{u, v\}$ in G.

1.3 An Alternate Partitioning of Hypercubes

We propose a different method for partitioning Q_n, with a number of potential benefits. Currently, when a process asks to use a subcube of Q_n, it gets full control of all the processors assigned to it. We propose that each node u of Q_n be divided into $\frac{n}{k}$ virtual nodes, where $2 \leq k \leq \frac{n}{2}$, and $n \bmod k = 0$. We propose that a k-factorization of Q_n be constructed. In other words, let the links of Q_n be divided into $\frac{n}{k}$ link-disjoint regular spanning subnetworks or factors $F_1, F_2, \ldots, F_{n/k}$ of Q_n with degree k. The j^{th} virtual node ($1 \leq \text{j} \leq \frac{n}{k}$) of some node u in Q_n is connected to the j^{th} virtual node of another node v in Q_n, by the links of F_j only.

Figure 1 shows an example of this arrangement, where $n = 4$ and $k = 2$. The numbers on the nodes of Q_4 are numeric representations of their bit string labels. Each node of Q_4 has been divided into 2 virtual nodes, which are colored black and gray. The links of Q_4 have been divided into two factors F_1 and F_2 of degree $\frac{4}{2} = 2$. F_1 is the set of gray links, connecting the gray virtual nodes, and F_2 is the set of black links, connecting the black virtual nodes. F_1 and F_2 are a 2-factorization of Q_4.

Under this arrangement, up to $\frac{n}{k}$ different processes can access a virtual copy of Q_n at the same time with no link interference between computations; since communication between virtual nodes is taking place over disjoint link sets. Another potential application for k-factorizations of Q_n is in the area of fault-tolerant computing. If a factor could efficiently simulate Q_n, then Q_n could tolerate the failure of all links not part of the factor. k-factorizations of Q_n could also be used in the construction of adaptive routing algorithms for Q_n, which make routing decisions based on the traffic for a particular node [11, 12].

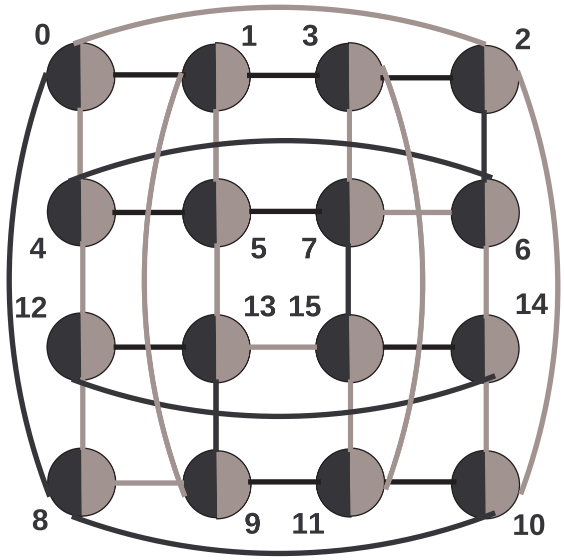

Figure 1: A 2-factorization of Q_4

n	2	3	4	5	6	7	8
Links in Q_n	4	12	32	80	192	448	1024
Removable Links	0	3	14	45	123	311	752

Table 1: The number of links removeable from Q_n without increasing diameter.

1.4 Progress in Finding Factors of Small Diameter

Q_n is known to have node-connectivity of n [17]. Menger's Theorem states if a network has node-connectivity k, then k node-disjoint paths connect any two distinct nodes [15]. Therefore, perhaps some, or even most of the links can be removed from Q_n without increasing its diameter. Discovering the number of removable links has been a subject of recent research [16, 10, 14]. It has been shown that $(n-2)2^{n-1}+1-\lceil\frac{2^n-1}{2n-1}\rceil$ links are removable from Q_n without increasing its diameter [10]. Table 1 shows the number of links removable from Q_n without increasing its diameter, for small values of n.

However, the resulting spanning subnetwork of Q_n is not regular, and therefore cannot be part of any k-factorization. Regular spanning subnetworks of Q_n are known to exist for specific values of n. These spanning subnetworks are described in Table 2. For example, the cube-connected cycles network of dimension n CCC_n [23] is known to be a spanning subnetwork of Q_{n+lgn}, where

Network	Spans	Degree	Diameter
CCC_n	Q_{n+lgn}	3	$\frac{5n}{2} - 2$
$ACCC_n^1$	Q_{n+lgn}	4	$2n$ - 2
$ACCC_n^2$	Q_{n+lgn}	6	$\frac{3n}{2}$
$Subcube_n$	Q_{n-1}	lg n	$\frac{3n}{2}$ - 2
$Q_{n,2,1}$	Q_n	$\lfloor \frac{n}{2} \rfloor + 1$	n

Table 2: Known regular spanning subnetworks of Q_n.

$n = 2^k$ and lg $n = \log_2 n$ [20]. The augmented cube-connected cycles networks $ACCC_n^1$ and $ACCC_n^2$ [8] are derived from adding links to CCC_n. The subcube network $Subcube_n$ [6] is both a spanning subnetwork of Q_{n-1} and a subnetwork of the pancake network of dimension n. The spanning subnetwork $Q_{n,2,1}$ [7] contains all the links for dimensions 0 and 1, and uses the value of the first two bits of the label of each node to determine the dimensions of links incident to that node. $Q_{n,2,1}$ is the first regular spanning subnetwork of Q_n with degree less than n and diameter n.

However, none of the networks of Table 2 are part of any k-factorization of the hypercubes they span, as they use all of the links for a particular dimension of Q_n. We therefore seek to identify k-factorizations of Q_n, where the factors have certain properties. The factors should have minimal degree (preferably $\Theta(1)$), so as to maximize the number of factors. The factors should have minimal diameter (preferably n, the diameter of Q_n). The factors in this paper all have degree $\frac{n}{2}$. It is an open question as to whether factors exist with degree smaller than $\frac{n}{2}$ and diameter n. Finally, the factors should be constructed so that there exists an embedding f of Q_n into each of the factors, with minimal dilation (preferably $\Theta(1)$). An embedding can be considered as a high-level description of how one network simulates another [22]. The dilation of an embedding is a commonly used measure of the efficiency of the simulation. Since parallel algorithms on hypercubes involve communication between adjacent nodes, the path in each factor between $f(u)$ and $f(v)$, where u and v are adjacent nodes in Q_n, should have a length of at most a constant in order for each factor to efficiently simulate Q_n.

2 Creating Hamilton Decompositions of Q_n

2.1 Creating Hamilton Decompositions of Q_{2n} from Hamilton Decompositions of Q_n

Q_n has been known to have a Hamilton decomposition for some time [4]. That is, it is known that the links of Q_n can be partitioned into disjoint Hamilton cycles. However, the proof did not readily result in an algorithm for producing the actual decomposition [1] [26].

Two algorithms are known for generating Hamilton decompositions of Q_n.

```
Algorithm HAMILTONCOMP1(n, input, output)
begin
    outCycle = 0
    for inCycle = 0 to n − 1 do
    begin
        for outCycleElement = 0 to 2^{2n} − 1 do
        begin
            Set the first n bits of output[outCycle][outCycleElement] to
                input[inCycle][outCycleElement div 2^n]
            Set the last n bits of output[outCycle][outCycleElement] to
                input[inCycle][(outCycleElement mod 2^n) −
                (outCycleElement div 2^n)]
        end
        outCycle = outCycle + 1
        for outCycleElement = 0 to 2^{2n} - 1 do
        begin
            Set the first n bits of output[outCycle][outCycleElement] to
                input[inCycle][(outCycleElement mod 2^n) −
                 (outCycleElement div 2^n)]
            Set the last n bits of output[outCycle][outCycleElement] to
                input[inCycle][outCycleElement div 2^n]
        end
        outCycle = outCycle + 1
    end
end
```

Figure 2: Creating a Hamilton decomposition of Q_{2n}

The first, discovered by Ringel and given in [24], yields a Hamilton decomposition of Q_{2n} from a Hamilton decomposition of Q_n. Each Hamilton cycle of the Hamilton decomposition of Q_n is used to form two disjoint Hamilton cycles of the Hamilton decomposition of Q_{2n}. Let the Hamilton decomposition of Q_n be stored in the two-dimensional array input$[n-1][2^n-1]$, where the first element for both dimensions (and for all dimensions of all arrays in this paper) is 0. The Hamilton decomposition of Q_{2n} will be stored in the array output$[2n-1][2^{2n}-1]$. The algorithm is shown in Figure 2.

Example: The cycle 00, 01, 11, 10 is a Hamilton decomposition of Q_2. The algorithm yields the Hamilton decomposition of Q_4, {{0000, 0001, 0011, 0010, 0110, 0100, 0101, 0111, 1111, 1110, 1100, 1101, 1001, 1011, 1010, 1000}, {0000, 0100, 1100, 1000, 1001, 0001, 0101, 1101, 1111, 1011, 0011, 0111, 0110, 1110, 1010, 0010}}. This is the Hamilton decomposition of Q_4 shown in Figure 1.

2.2 Creating Hamilton Decompositions of Q_{2n+2} from Hamilton Decompositions of Q_{2n}

The second Hamilton decomposition algorithm, based on [26] and given in [5], yields a Hamilton decomposition of Q_{2n+2} from a Hamilton decomposition of Q_{2n}, and an orthogonal matching to the Hamilton decomposition of Q_{2n}. This algorithm is based on the following result:

Theorem 1 *If a network N can be decomposed into n - 1 Hamilton cycles, and there exists a matching orthogonal to the set of Hamilton cycles, then $N \times C_{2k}$, $k \geq 2$, can be decomposed into n Hamilton cycles* [*26*].

C_4, the cycle of 4 nodes, is but another way of describing Q_2. It is known that $Q_i \times Q_j = Q_{i+j}$ for all nonnegative integers i and j. We take advantage of these facts to arrive at the following corollary:

Corollary 1: If Q_{2n} can be decomposed into n Hamilton cycles, and there exists a matching in Q_{2n} orthogonal to the set of Hamilton cycles, then $Q_{2n} \times C_4 = Q_{2n} \times Q_2 = Q_{2n+2}$ can be decomposed into $n + 1$ Hamilton cycles.

The algorithm [25] generates two Hamilton cycles for Q_{2n+2} from a selected Hamilton cycle for Q_{2n}, and one Hamilton cycle for Q_{2n+2} from each of the remaining Hamilton cycles for Q_{2n}. Let $N = 2^{2n}$. Assume the nodes of Q_{2n} are labeled by the integers 0, 1, ..., N - 1, where two nodes are adjacent if they differ in exactly one bit in their binary representations. The n Hamilton cycles of Q_{2n} are stored in the array in[n][N - 1]. The $n + 1$ Hamilton cycles of Q_{2n+2} are stored in the array out[n + 1][4N - 1]. We arrange the cycles so that the links of the orthogonal matching are {{in[0][0], in[0][N - 1]}, {in[1][0], in[1][N - 1]}, ..., {in[n - 1][0], in[n - 1][N - 1]}}. Furthermore, flipping cycles if necessary, we arrange that for 1 $\leq$ i $\leq$ n - 1, in[i][0] occurs before in[i][N - 1] in the list in[0][0], in[0][1], ..., in[0][N - 1]. The purpose of arranging the edges in the orthogonal matching in this manner is to simplify the algorithm. We also use an array flag[n] to keep track of the paths taken through nodes of Q_{2n}, which are related to endpoints of links in the matching. The algorithm is shown in Figures 3, 4 and 5 and 6.

Example Let C_1 and C_2 be the Hamilton cycles of Q_4 shown in Figure 1. Let C_1 be the black links, and C_2 be the gray links. In this example, $n = 2$ and N = 16. C_1 can be expressed as {0, 1, 3, 2, 6, 4, 5, 7, 15, 14, 12, 13, 9, 11, 10, 8}, and C_2 can be expressed as {4, 0, 2, 10, 14, 6, 7, 3, 11, 15, 13, 5, 1, 9, 8, 12}. The orthogonal matching we will use will be the two links {0, 8} and {4, 12}. The cycles have been arranged so that the links of the orthogonal matching are in the proper position.

The algorithm FIRST-OUTPUT-CYCLE produces the Hamilton cycle for Q_6 {0, 1, 3, 2, 6, 4, 5, 7, 15, 14, 12, 13, 9, 11, 10, 8, 40, 56, 48, 16, 24, 26, 58, 42, 43, 59, 27, 25, 57, 41, 45, 61, 29, 28, 30, 62, 46, 47, 63, 31, 23, 55, 39, 37, 53, 21, 20, 52, 60, 44, 36, 38, 54, 22, 18, 50, 34, 35, 51, 19, 17, 49, 33, 32}.

```
Algorithm FIRST-OUTPUT-CYCLE(n, N, in, out)
begin
    Set out[0][0] through out[0][N - 1] to
         in[0][0] through in[0][N - 1], respectively
    if n is even then begin
        Set out[0][N] through out[0][N + 4] to
            in[0][N - 1] + 2N, in[0][N - 1] + 3N,
            in[0][0] + 3N, in[0][0] + N, in[0][N - 1] + N, respectively
        count = N + 5
    end
    else begin
        out[0][N] = in[0][N - 1] + N
        count = N + 1
    end
    for j = N - 2 downto 1 do
    begin
        if out[0][count - 1] is of the form in[0][j + 1] + N and
            in[0][j] is not an endpoint of a link in the matching then
        begin
            Set out[0][count] through out[0][count + 2] to in[0][j] + N,
                in[0][j] + 3N, in[0][j] + 2N, respectively
            count = count + 3
        end
        else if out[0][count - 1] is of the form in[0][j + 1] + 2N and
            in[0][j] is not an endpoint of an edge in the matching then
        begin
            Set out[0][count] through out[0][count + 2] to in[0][j] + 2N,
                in[0][j] + 3N, in[0][j] + N, respectively
            count = count + 3
        end
        else if out[0][count - 1] is of the form in[0][j + 1] + N and
            in[0][j] = in[k][N - 1] for some k then
        begin
            out[0][count] = in[0][j] + N
            count = count + 1
            flag[k] = 0
        end
        else if out[0][count - 1] is in[0][j + 1] + 2N and
            in[0][j] = in[k][N-1] for some k then
        begin
            out[0][count] = in[0][j] + 2N
            count = count + 1
            flag[k] = 1
        end
```

Figure 3: Creating the first cycle of the Hamilton decomposition.

```
        else if out[0][count - 1] is of the form in[0][j + 1] + N and
            in[0][j] = in[k][0] for some k then
            if flag[k] = 0 then
            begin
                Set out[0][count] to out[0][count + 4] to
                    in[0][j]+ N, in[0][j] + 3N,
                    in[k][N - 1] + 3N, in[k][N - 1] + 2N, in[0][j] + 2N,
                    respectively
                count = count + 5
            end
            else
            begin
                Set out[0][count] to
                    out[0][count + 4] to in[0][j] + N, in[k][N - 1] + N,
                    in[k][N - 1] + 3N, in[0][j] + 3N, in[0][j] + 2N, respectively
                count = count + 5
            end
        else if out[0][count - 1] is of the form in[0][j + 1] + 2N and
            in[0][j] = in[k][0] for some k then
            if flag[k] = 1 then
            begin
                Set out[0][count] to
                    out[0][count + 4] to in[0][j] + 2N, in[0][j]+ 3N,
                    in[k][N - 1] + 3N, in[k][N - 1] + N, in[0][j] + N, respectively
                count = count + 5
            end
            else
            begin
                Set out[0][count] to
                    out[0][count + 4] to in[0][j] + 2N, in[k][N - 1] + 2N,
                    in[k][N - 1] + 3N, in[0][j] + 3N, in[0][j] + N, respectively
                count = count + 5
            end
    end
    if n is even then
        out[0][count] = in[0][0] + 2N
    else
        Set out[0][4N - 5] through out[0][4N - 1] to in[0][0] + N, in[0][0] + 3N,
            in[0][N - 1] + 3N, in[0][N - 1] + 2N, in[0][0] + 2N, respectively
end
```

Figure 4: Creating the first cycle (Continued).

```
Algorithm SECOND-OUTPUT-CYCLE(n, N, in, out)
begin
    Set out[1][0] through out[1][N - 1] to in[0][0] + 3N
        through in[0][N - 1] + 3N, respectively
    if n is even then
    begin
        Set out[1][N] through out [1][N + 3] to in[0][N - 1] + N,
            in[0][N - 1], in[0][0], in[0][0] + N, respectively
        for count = N + 4 to 4N - 2 do
            output[1][count] = output[0][5N + 2 - count] XOR 3N
        output[1][4N - 1] = input[0][0] + 2N
    end
    else
    begin
        Set out[1][N] through out[1] to in[0][N - 1] + N, in[0][0] + N, in[0][0],
            in[0][N - 1], in[0][N - 1] + 2N
        for count = N + 5 to 4N - 1 do
            out[1][count] = out[0][count - 4] XOR 3N
    end
end
```

Figure 5: Creating the second cycle of the Hamilton decomposition

```
Algorithm REMAINING-CYCLES(n, in, out, N)
begin
    for cycle = 1 to n - 1 do
        for node = 0 to N - 1 do
        begin
            out[cycle + 1][node] = in[cycle][node]
            out[cycle + 1][node + 2N] = in[cycle][node] + 3N
            out[cycle + 1][node + N] = in[cycle][N - 1 - node] +
                ((2 - flag[cycle]) * N)
            out[cycle + 1][node + 3N] = in[cycle][N - 1 - node] +
                ((1 + flag[cycle]) * N)
        end
end
```

Figure 6: Creating the remaining cycles of the Hamilton decomposition.

The algorithm SECOND-OUTPUT-CYCLE produces the Hamilton cycle for Q_6 {48, 49, 51, 50, 54, 52, 53, 55, 63, 62, 60, 61, 57, 59, 58, 56, 24, 8, 0, 16, 17, 1, 33, 35, 3, 19, 18, 2, 34, 38, 6, 22, 20, 28, 12, 4, 36, 37, 5, 21, 23, 7, 39, 47, 15, 31, 30, 14, 46, 44, 45, 13, 29, 25, 9, 41, 43, 11, 27, 26, 10, 42, 40, 32}. Taking elements of the first output cycle, and performing an exclusive or with $3N$ has the effect of toggling (changing 0 to 1 and 1 to 0) the first two bits of those elements.

Finally, the algorithm REMAINING-CYCLES produces the Hamilton cycle for Q_6 {4, 0, 2, 10, 14, 6, 7, 3, 11, 15, 13, 5, 1, 9, 8, 12, 44, 40, 41, 33, 37, 45, 47, 43, 35, 39, 38, 46, 42, 34, 32, 36, 52, 48, 50, 58, 62, 54, 55, 51, 59, 63, 61, 53, 49, 57, 56, 60, 28, 24, 25, 17, 21, 29, 31, 27, 19, 23, 22, 30, 26, 18, 16, 20}.

Variable flag[1] was set to 0 in the course of executing algorithm FIRST-OUTPUT-CYCLE. REMAINING-CYCLES creates four copies of C_2 with one edge removed, within Q_6, by adding either 0, N, $2N$ or $3N$. The Hamilton paths where N and $3N$ are added are traversed in the opposite direction of the Hamilton paths where 0 and $2N$ are added. Since REMAINING-CYCLES uses $n-1$ disjoint Hamilton cycles of Q_{2n} as input, it creates $n-1$ disjoint Hamilton cycles for Q_{2n+2}.

The Hamilton decomposition of Q_{2n+2} generated by this algorithm is partially determined by the Hamilton cycle of Q_{2n}, which is selected as input[0], the input to FIRST-OUTPUT-CYCLE and SECOND-OUTPUT-CYCLE. It is also partially determined by the edges selected for the orthogonal matching required by the algorithm. It is therefore possible that a large number of distinct Hamilton decompositions of Q_{2n+2} can be generated using this algorithm. For example, four distinct Hamilton decompositions of Q_4 were generated using this algorithm.

3 Constructing (n/2)-Factorizations of Q_n

3.1 Constructing Factorizations from Perfect Matchings Derived From Hamilton Decompositions

In Section 1.3, we proposed creating k-factorizations of Q_n. In this section and the next, we use Hamilton decompositions of Q_n to create $(n/2)$-factorizations of Q_n for certain values of n. One method of constructing $(n/2)$-factorizations from Hamilton decompositions is uniting perfect matchings derived from the cycles of the Hamilton decomposition.

Suppose $C_1, C_2, \ldots, C_{n/2}$ is a Hamilton decomposition of Q_n. If the links of any cycle were numbered, the even-numbered links would form a perfect matching of Q_n, as would the odd-numbered links. n link-disjoint perfect matchings of Q_n can be constructed in this manner. When n mod k = 0, and $\frac{n}{k}$ unions of k perfect matchings are selected, the result is a k-factorization of Q_n. However, not all unions of k perfect matchings are connected. Table 3 shows the results of a computer search for the $(n/2)$-factorizations of Q_n, whose factors had the smallest diameter.

Network	Degree of Factors	Diameter
Q_4	2	8, 8
Q_6	3	8, 10
Q_8	4	8, 8
Q_{12}	6	12, 12

Table 3: Degrees and diameters of factors in factorization of Q_n

The 2-factorization of Q_4 is the same as a Hamilton decomposition of Q_4, where the cycles of 16 nodes have diameter 8. The difference in the diameters of the factors in the 3-factorization of Q_6 reflects the fact that the algorithm of Section 2.2 generates two Hamilton cycles for Q_6 from one of the Hamilton cycles for Q_4, and one Hamilton cycle for Q_6 from the other Hamilton cycle for Q_4. Table 3 shows that half the links incident to each node of both Q_8 and Q_{12} can be removed without increasing their diameters. Furthermore, the removed links themselves form a spanning subnetwork with the same diameter as the original hypercube.

3.2 Constructing Factorizations from Hamilton Cycles of Hamilton Decompositions

Observation of Algorithm HAMILTONCOMP1 reveals that the algorithm creates two disjoint Hamilton cycles of Q_{2n} for each Hamilton cycle of Q_n. If C is a Hamilton cycle of Q_n, then let these Hamilton cycles of Q_{2n} be called the *children* of C. Let the *descendants of C at level k* be the 2^k disjoint Hamilton cycles of Q_{n2^k}, obtained by repeatedly applying the algorithm. Let *D(C, k)* represent the union of the descendants of C at level k.

We observe the following regarding the children of C. One of the children has a pattern of changing the first n bits 2^n - 1 times, then changing the last n bits once. This pattern is repeated 2^n times. The other child has a pattern of changing the last n bits 2^n - 1 times, then changing the first n bits once, a pattern which is repeated 2^n times. In general, each descendant of C at level k has a pattern of changing a unique block of n bits 2^n - 1 times, then changing some other bit once.

Lemma 1 *If C is a Hamilton cycle in Q_n, then $D(C, k)$ is a spanning subnetwork of Q_{n2^k}, with degree 2^{k+1} and diameter 2^{n-1+k}.*

Proof: The diameter of C is 2^{n-1}. Since $D(C, k)$ is the union of $2k$ Hamilton cycles of Q_{n2^k} , $D(C, k)$ is a spanning subnetwork of Q_{n2^k}. Since $D(C, k)$ is the union of 2^k disjoint Hamilton cycles of Q_{n2^k}, $D(C, k)$ is of degree 2^{k+1}. Let w and w' be the labels of two nodes in $D(C, k)$. Algorithm ROUTE provides a route in $D(C, k)$ between w and w', and is shown in Figure 7. It takes at most 2^{n-1} nodes to arrange the bits of each of the 2^k blocks of n bits, therefore the diameter is 2^{n-1+k}. □

```
Algorithm ROUTE(w, w')
begin
    for i = 0 to 2k - 1 do
    begin
        Select the descendant of C at level k which changes the i^th block of n
            bits 2n - 1 times
        Set the i^th block of n bits of w to the i^th block of n bits of w' by
            traversing that descendant, using as few nodes as possible
    end
end
```

Figure 7: A routing algorithm for $D(C_0, k)$

Theorem 2 *Let C be the Hamilton cycle for Q_2. Then $D(L(C), k)$ and $D(R(C), k)$ are mutually isomorphic.*

Proof: Consider any of the cycles in $D(L(C), k)$. If the labels of the nodes of this cycle are reversed, then the labels for one of the cycles in $D(R(C), k)$ are obtained. This is because in the cycles of $D(L(C), k)$, some portion of the first half of the 2^{k+2} bits of the labels are changed most rapidly while traversing the cycle, while in the cycles of $D(R(C), k)$, some portion of the second half of the bits of the labels are changed most rapidly. □

Theorem 3 *For $j \geq 2$, there exists an 2^{j-1}-factorization of Q_{2^j} where the two factors have diameter 2^{j+1}.*

Proof: Let A and B be any Hamilton decomposition of Q_4. A and B are a 2-factorization of Q_4, where each of the factors have diameter 8. $D(A, k)$ and $D(B, k)$, k $\geq$ 1, form a 2^{k+1}-factorization of $Q_{4*2^k} = Q_{2^{k+2}}$, because they are comprised of all the Hamilton cycles of the Hamilton decomposition of $Q_{2^{k+2}}$. $D(A, k)$ and $D(B, k)$ have degree 2^{k+1} and diameter $2^{4-1+k} = 2^{k+3}$ by Lemma 1, which is twice the diameter of $Q_{2^{k+2}}$. □

In Section 1.4, we mentioned that in order for a factor to effectively simulate Q_n, an embedding of Q_n into the factor must exist with no more than constant dilation.

Theorem 4 *Let A and B represent the Hamilton cycles of any Hamilton decomposition of Q_4. For $j > 2$, The identity embedding embeds Q_{2^j} into $D(A, j - 2)$ and $D(B, j - 2)$ with $\Theta(1)$ dilation.*

Proof: Suppose we wish to route in either $D(A, j$ - 2) or $D(B, j$ - 2) between adjacent nodes in u and v differ in some bit in some block of 4 bits. Without loss of generality, we select $D(A, j$ - 2). There exists a descendant of A at level k - 2, which changes the block of 4 bits, containing the bit in which u and v

differ, 2^4 - 1 times before changing another bit. By using that descendant, we can route from u to v in at most 2^4 - 1 = 15 = $\Theta(1)$ links. □

In summary, it is possible to create $(n/2)$-factorizations from Hamilton decompositions of Q_n in two ways; by uniting perfect matchings derived from Hamilton cycles, and by uniting the Hamilton cycles themselves. It is currently unknown whether the factors in the factorizations of Section 3.1 have a Hamilton cycle. Since the factors mentioned in this section are composed of the union of Hamilton cycles, they have Hamilton cycles of their own. Furthermore, Hamilton decompositions exist for the factors as well.

3.3 Constructing Factorizations of Q_n from Variations on Reduced and Thin Hypercubes

Many reduced-degree variations on hypercubes have been proposed. Some of these variants use the values of portions of the labels of nodes, to determine the dimensions of the links incident to those nodes. Examples include the reduced hypercube [27], and the thin hypercube [7, 18]. The motivation for these networks was to construct a subnetwork of a hypercube with a smaller degree than the original hypercube, and a diameter which is either the same (thin hypercube) or only slightly larger (reduced hypercube) than the original hypercube. However, neither the reduced hypercube nor the thin hypercube are part of a k-factorization. We use the idea behind reduced and thin hypercubes, to construct an $(n/2)$-factorization of Q_n, where n is even, where the factors have diameter $n + \Theta(1)$. This factorization was first given in [9].

Consider Q_n, where n is even. Recall that the label of each node is a bit string $b_0 b_1 \ldots b_{n-1}$. Let the substring $b_0 b_1$ represent the first two bits of the label of a node. Let the parity of a bit string signify the number of 1's in the bit string. Let F_1 be a degree $\frac{n}{2}$ spanning subnetwork, defined as follows:

- If a label of a node has the value 00 in $b_0 b_1$, then that node is incident to links in dimensions n - 4 and n - 3
- If a label of a node has the value 01 in $b_0 b_1$, then that node is incident to links in dimensions n - 3 and n - 2
- If a label of a node has the value 11 in $b_0 b_1$, then that node is incident to links in dimensions n - 2 and n - 1
- If a label of a node has the value 10 in $b_0 b_1$, then that node is incident to links in dimensions n - 1 and n - 4
- If a label of a node has even parity in $b_{n-4} b_{n-3}\ b_{n-2} b_{n-1}$, then that node is incident to a link in dimensions 0, 2, ..., n - 6. Otherwise, that node is incident to a link in dimension 1, 3, ..., n - 5.

For example, the node with the label 010010, is incident to links of dimensions 3 and 4, because 01 is the value of $b_0 b_1$, and is connected to nodes with

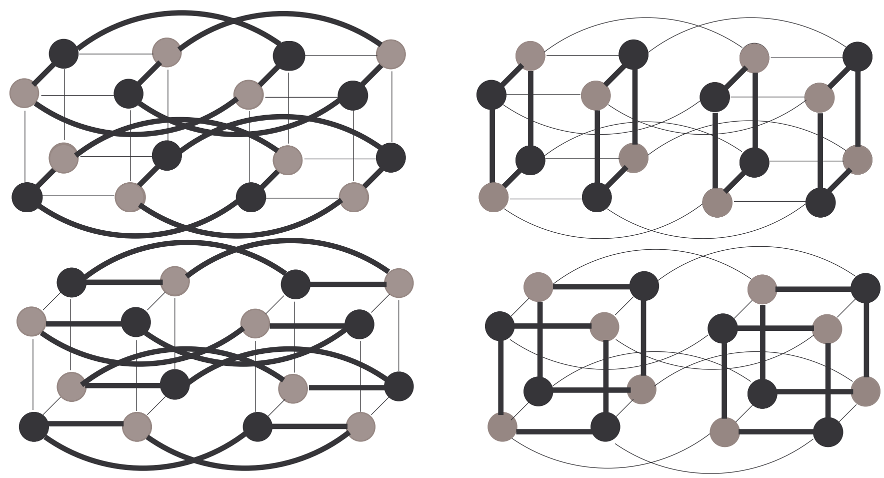

Figure 8: F_1, a spanning subnetwork of Q_6, for $n = 6$.

labels 010110 and 010000. This node is incident to a link of dimension 1, since 0010, the value of $b_2b_3b_4b_5$, has odd parity. Therefore, this node is connected to the node with label 000010. F_1 is shown in Figure 8, with the dimension 0 and 1 links not shown. The bold links in Figure 8 are the links of F_1. The black nodes are incident to dimension 0 links, and the gray nodes are incident to dimension 1 links.

Let F_2 be a degree $\frac{n}{2}$ spanning subnetwork, defined as follows:

- If a label of a node has the value 00 in b_0b_1, then that node is incident to links in dimensions n - 2 and n - 1
- If a label of a node has the value 01 in b_0b_1, then that node is incident to links in dimensions n - 1 and n - 4
- If a label of a node has the value 11 in b_0b_1, then that node is incident to links in dimensions n - 4 and n - 4
- If a label of a node has the value 10 in b_0b_1, then that node is incident to links in dimensions n - 3 and n - 2
- If a label of a node has odd parity in $b_{n-4}b_{n-3}\ b_{n-2}b_{n-1}$, then that node is incident to a link in dimensions 0, 2, . . . , n - 6. Otherwise, that node is incident to a link in dimension 1, 3, . . . , n - 5.

F_1 and F_2 form an $\frac{n}{2}$-factorization of Q_n. F_1 and F_2 contain exactly half of the links in each dimension. If a node u in Q_n is incident to links in dimensions

0, 2, ..., n - 6, then any node v adjacent to u by dimensions n - 4 through n - 1, is incident to links in dimensions 1, 3, ..., n - 5.

Theorem 5 *F_1 and F_2 are isomorphic to each other.*

Proof: F_1 and F_2 are isomorphic if there exists a mapping g from the nodes of F_1 to the nodes of F_2, such that for all pairs of adjacent nodes u and v in F_1, $g(u)$ is adjacent to $g(v)$ in F_2. Let $g(u)$ be obtained from u by toggling b_0 and b_1. If u and v are adjacent in F_1 by some dimension, then $g(u)$ will be adjacent to $g(v)$ in F_2 by the same dimension. □

Theorem 6 *Both F_1 and F_2 have diameter n + 2.*

Proof: Let u and v be two nodes in F_1. Without loss of generality, let the value of b_0b_1 in u be 00, and let $b_{n-4}b_{n-3}b_{n-2}b_{n-1}$ have even parity. Assume bits b_2 through b_{n-1} of u are to be toggled to form bits b_2 through b_{n-1} of v.

Case I: b_0b_1 in v is 00. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 10. Now b_{n-1} can be toggled. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 11. Now b_{n-2} can be toggled. Toggle b_0, causing b_0b_1 to be 01. Now b_{n-3} can be toggled. Toggle b_1, causing b_0b_1 to be 00. Now b_{n-4} can be toggled. b_0 and b_1 were toggled twice, while the remaining bits were toggled once, for a total of n + 2 links.

Case II: b_0b_1 in v is 01. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 10. Now b_{n-1} and b_{n-4} can be toggled. Toggle b_0, causing b_0b_1 to be 00. Now b_{n-2} can be toggled. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 11. Now b_{n-3} can be toggled. b_0 was toggled twice, while the remaining bits were toggled once, for a total of n + 1 links.

Case III: b_0b_1 in v is 11. Toggle b_{n-4} and b_{n-3}. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 10. Now b_{n-1} can be toggled. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 11. Now b_{n-2} can be toggled. No bit was toggled more than once for a total of n links.

Case IV: b_0b_1 in v is 10. Toggle b_{n-3}. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 01. Now b_{n-2} can be toggled. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 11. Now b_{n-1} can be toggled. Toggle b_1, causing b_0b_1 to be 10. Now b_{n-4} can be toggled. b_1 was toggled twice, while the remaining bits were toggled once, for a total of n + 1 links.

Let u and v be two nodes in F_2. Without loss of generality, let the value of b_0b_1 in u be 00, and let $b_{n-4}b_{n-3}b_{n-2}b_{n-1}$ have odd parity. Assume bits b_2 through b_{n-1} of u are to be toggled to form bits b_2 through b_{n-1} of v.

Case I: b_0b_1 in v is 00. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 01. Now b_{n-1} can be toggled. Toggle b_0, b_2, … b_{n-6}, causing b_0b_1 to be 11. Now b_{n-4} can be toggled. Toggle b_1, causing b_0b_1 to be 10. Now b_{n-3} can be toggled. Toggle b_0, causing b_0b_1 to be 00. Now b_{n-2} can be toggled. b_0 and b_1 were toggled twice, while the remaining bits were toggled once, for a total of n + 2 links.

Case II: b_0b_1 in v is 01. Toggle b_{n-2}. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 10. Now b_{n-3} can be toggled. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be

11. Now b_{n-4} can be toggled. Toggle b_0, causing b_0b_1 to be 01. Now b_{n-1} can be toggled. b_0 was toggled twice, while the remaining bits were toggled once, for a total of $n + 1$ links.

Case III: b_0b_1 in v is 11. Toggle b_{n-2} and b_{n-1}. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 01. Now b_{n-4} can be toggled. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 11. Now b_{n-3} can be toggled. No bit was toggled more than once for a total of n links.

Case IV: b_0b_1 in v is 10. Toggle b_1, b_3, …, b_{n-5}, causing b_0b_1 to be 01. Now b_{n-1} and b_{n-4} can be toggled. Toggle b_1, causing b_0b_1 to be 00. Now b_{n-2} can be toggled. Toggle b_0, b_2, …, b_{n-6}, causing b_0b_1 to be 10. Now b_{n-3} can be toggled. b_0 was toggled twice, while the remaining bits were toggled once, for a total of $n + 1$ links. □

Theorem 7 *The identity embedding embeds Q_n into both F_1 and F_2 with dilation 5.*

Proof: Let u and v be two nodes in Q_n, which differ in exactly 1 bit. Without loss of generality, let the value of b_0b_1 in u be 00, and let $b_{n-4}b_{n-3}b_{n-2}b_{n-1}$ have even parity. We show that the maximum distance in F_1 between u and v is 5.

Case I: u and v differ in either b_0, b_2, …, b_{n-6}, b_{n-4} or b_{n-3}. In this case, u and v are adjacent in F_1.

Case II: u and v differ in b_1, b_3, …, or b_{n-5}. In this case, toggle b_{n-3}, toggle the bit in which u and v differ, then toggle b_{n-3} again, for a total of three links.

Case III: u and v differ in b_{n-2}. In this case, toggle b_{n-3}, toggle b_1, toggle b_{n-2}, toggle b_{n-3}, and toggle b_1, for a total of five links.

Case IV: u and v differ in b_{n-1}. In this case, toggle b_0, toggle b_{n-1}, toggle b_{n-4}, toggle b_0, and toggle b_{n-4}, for a total of five links.

We now show that the maximum distance in F_2 between u and v is also 5.

Case I: u and v differ in either b_1, b_3, …, b_{n-5}, b_{n-2} or b_{n-1}. In this case, u and v are adjacent in F_1.

Case II: u and v differ in b_0, b_2, …, or b_{n-6}. In this case, toggle b_{n-2}, toggle the bit in which u and v differ, then toggle b_{n-2} again, for a total of three links.

Case III: u and v differ in b_{n-4}. In this case, toggle b_1, toggle b_{n-4}, toggle b_{n-1}, toggle b_1, and toggle b_{n-1}, for a total of five links.

Case IV: u and v differ in b_{n-3}. In this case, toggle b_{n-2}, toggle b_0, toggle b_{n-3}, toggle b_{n-2}, and toggle b_0, for a total of five links. □

4 Conclusions and Future Research

Table 4 summarizes our findings. The consequence of our findings is that the links of Q_n can be partitioned into two factors, each having a diameter close to that of Q_n. The factorizations can be produced either from Hamilton decompositions or directly. Furthermore, since there is an embedding of Q_n into these factors with constant dilation, the factors can efficiently simulate the operation of Q_n.

	Best Possible	Section 3.1	Section 3.2	Section 3.3
Original Hypercube	Q_n	Q_4, Q_6, Q_8, Q_{12}	Q_n, n = 2^k	Q_n, n is even
Degree of Factors	$\Theta(1)$	2, 3, 4, 6	$\frac{n}{2}$	$\frac{n}{2}$
Mutually Isomorphic?	Yes	Unknown	Yes	Yes
Diameter of factors	n	{8, 8}, {8, 10}, {8, 8}, {12, 12}	$2n$	$n + 2$
Best Dilation	$\Theta(1)$	Unknown	$\Theta(1)$	$\Theta(1)$
Hamilton Cycle?	Yes	Unknown	Yes	Unknown
Hamilton Decomposition?	Yes	Unknown	Yes	Unknown

Table 4: Properties of factorizations of hypercubes.

Possible directions for future research into Hamilton decompositions include identifying Hamilton decompositions for other well-known networks, determining if a given Hamilton cycle is part of a Hamilton decomposition and using Hamilton decompositions for solutions to various graph problems [21].

Possible directions for future research into k-factorizations include 1) determining the existence of a k-factorization of Q_n, constructed from perfect matchings, where the factors have diameter n, 2) determining if k-factorizations of Q_n exist where $k < \frac{n}{2}$, and the diameters of the factors is n, 3) finding embeddings of minimal dilation of Q_n into its factors.

Acknowledgments

The authors thank Brian Alspach of the University of Regina, for his assistance regarding Hamilton decompositions and k-factorizations, Richard Stong of Rice University for providing the Hamilton decomposition algorithm of Section 2.2, and the referees for several useful suggestions.

References

[1] B. Alspach. private communication.

[2] B. Alspach, J.-C. Bermond, and D. Sotteau. Decomposition into cycles 1: Hamilton decompositions. In G. Hahn, G. Sabidussi, and R. Woodrow, editors, *Cycles and Rays*, pages 9–18. Kluwer Academic Publishers, 1990.

[3] B. Alspach, K. Heinrich, and G. Liu. Orthogonal factorizations of graphs. In J. Dinitz and D. Stinson, editors, *Contemporary Design Theory: A Collection of Surveys*, pages 13–40. John Wiley and Sons, 1992.

[4] J. Aubert and B. Schneider. Dcompositions de la somme cartsienne d'un cycle et l'union de deux cycles hamiltoniens en cycles hamiltoniens. *Discrete Mathematics*, 38:7–16, 1982.

[5] D. Bass and I. H. Sudborough. Link-disjoint regular spanning subnetworks of hypercubes. In *Proceedings of the 2nd IASTED International Conference on Parallel and Distributed Computers and Networks*, pages 182–185, 1998.

[6] D. Bass and I. H. Sudborough. Pancake problems with restricted prefix reversals and some corresponding cayley networks. In *Proceedings of the 27th International Conference on Parallel Processing*, pages 11–17, 1998.

[7] D. Bass and I. H. Sudborough. Vertex-symmetric spanning subnetworks of hypercubes with small diameter. In *Proceedings of the 11th IASTED International Conference on Parallel and Distributed Computers and Systems*, pages 7–12, 1999.

[8] D. Bass and I. H. Sudborough. Removing edges from hypercubes to obtain vertex-symmetric networks with small diameter. *Telecommunications Systems*, 13(1):135–146, 2000.

[9] D. Bass and I. H. Sudborough. Symmetric k-factorizations of hypercubes with factors of small diameter. In *Proceedings of the 6th International Symposium on Parallel Architectures, Algorithms and Networks*, pages 219–224, 2002.

[10] A. Bouabdallah, C. Delorme, and S. Djelloul. Edge deletion preserving the diameter of the hypercube. *Discrete Applied Mathematics*, 63:91–95, 1995.

[11] B. D'Auriol. private communication.

[12] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[13] S. Dutt and J. P. Hayes. Subcube allocation in hypercube computers. *IEEE Transactions on Computers*, C-40:341–352, 1991.

[14] P. Erdos, P. Hamburger, R. Pippert, and W. Weakley. Hypercube subgraphs with minimal detours. *Journal of Graph Theory*, 23(2):119–128, 1996.

[15] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[16] N. Graham and F. Harary. Changing and unchanging the diameter of a hypercube. *Discrete Applied Mathematics*, 37/38:265–274, 1992.

[17] M.-C. Heydemann. Cayley graphs and interconnection networks. In G. Hahn and G. Sabidussi, editors, *Graph Symmetry*, pages 167–224. Kluwer Academic Publishers, 1997.

[18] O. Karam. Thin hypercubes for parallel computer architecture. In *Proceedings of the 11th IASTED International Conference on Parallel and Distributed Computers and Systems*, pages 66–71, 1999.

[19] S. Latifi. Distributed subcube identification algorithms for reliable hypercubes. *Information Processing Letters*, 38(6):315–321, 1991.

[20] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1992.

[21] U. Meyer and J. F. Sibeyn. Time-independent gossiping on full-port tori, research report mpi-i-98-1014. Technical report, Max-Planck Institut fr Informatik, 1998.

[22] B. Monien and I. H. Sudborough. Embedding one interconnection network in another. *Computing Supplement*, 7:257–282, 1990.

[23] F. P. Preparata and J. Vuillemin. The cube connected cycles: A versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, 1981.

[24] G. Ringel. Über drei kombinatorische probleme am n-dimensionalen würfel und würfelgitter. *Abh. Math. Sem. Univ. Hamburg*, 20:10–19, 1956.

[25] R. Stong. private communication.

[26] R. Stong. Hamilton decompositions of cartesian products of graphs. *Discrete Mathematics*, 90:169–190, 1991.

[27] S. G. Ziavras. Rh: a versatile family of reduced hypercube interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1210–1220, 1994.