

**Space- and Time-Efficient Packings and
Embeddings of Hypercubes into Star Graphs**

Marcelo Moraes de Azevedo and Nader Bagherzadeh

Department of Electrical and Computer Engineering
University of California, Irvine – Irvine, CA 92717

Shahram Latifi

Department of Electrical and Computer Engineering
University of Nevada, Las Vegas – Las Vegas, NV 89154-4026

November 1994 - Technical Report ECE 94-11-01

Space- and Time-Efficient Packings and Embeddings of Hypercubes into Star Graphs*

Marcelo Moraes de Azevedo[†], Shahram Latifi[‡] and Nader Bagherzadeh[†]

[†]Dept. of Electrical and Computer Engineering [‡]Dept. of Electrical and Computer Engineering
University of California, Irvine University of Nevada, Las Vegas
Irvine, CA 92717 Las Vegas, NV 89154-4026
email: mazevedo, nader@ece.uci.edu email: latifi@jb.ee.unlv.edu

Abstract — Packing is a graph simulation technique by which p_k node-disjoint copies of a k -dimensional guest graph $G(k)$ are embedded into an n -dimensional host graph $H(n)$. Many advantages result from this technique as opposed to a simple embedding of $G(k)$ into $H(n)$. The multiple copies of $G(k)$ can execute different instances of any algorithm designed to run in $G(k)$, providing high throughput via an efficient, low-expansion utilization of $H(n)$. Task migration mechanisms between the multiple copies of $G(k)$ also become possible, allowing a proper allocation of the nodes of $H(n)$, load balancing and support for fault tolerance. Other advantages that arise from a well-devised packing technique are variable-dilation embeddings and multiple-sized packings. A variable-dilation embedding consists of connecting $c(k \rightarrow k + \ell)$ copies of a graph $G(k)$, packed into a host graph $H(n)$ with dilation d , such that an embedding of a $(k + \ell)$ -dimensional graph $G(k + \ell)$, $\ell > 0$, into $H(n)$ is obtained. The resulting embedding has dilation d when the nodes of $G(k + \ell)$ communicate over the first k dimensions of $G(k + \ell)$, and dilation $d_i > d$ when a dimension i , $k < i \leq k + \ell$, is used. Since many parallel algorithms use a restricted number of dimensions of the guest graph at any given step, the resulting communication slowdown can be made significantly small on the average. We also extend the concept of connecting node-disjoint copies of a graph $G(k)$ to obtain multiple-sized packings, in which graphs $G(k), G(k + 1), \dots, G(k + \ell)$ of various sizes are packed into a host graph $H(n)$. Multiple-sized packings allow tasks with different node requirements to be allocated proper guest graphs in $H(n)$.

This paper focuses on the problem of packing hypercubes $Q(k)$ into a star graph $S(n)$ with dilation 3, for $\lfloor n/2 \rfloor \leq k < n$. Our techniques achieve dense packings of $Q(k)$ into $S(n)$, and often reach 100% or close to 100% utilization of the star graph. We also show how to connect packed $Q(n - 1)$'s to obtain a variable-dilation embedding of $Q(n - 1 + \ell)$, $0 < \ell \leq \lfloor \log_2(\lfloor n/2 \rfloor! \cdot \lfloor (n - 1)/2 \rfloor!) \rfloor$, into $S(n)$. Such an embedding has dilation 3 for the first $(n - 1)$ dimensions of $Q(n - 1 + \ell)$ and guarantees a minimal slowdown by using a slightly higher dilation (4 in most cases) for the remaining dimensions of $Q(n - 1 + \ell)$. Finally, we address the issue of multiple-sized packings of hypercubes into $S(n)$.

Index terms — Graph simulation, interconnection networks, hypercube embedding, hypercube packing, parallel processing, star graph, variable-dilation embeddings.

1 Introduction

The star graph was proposed as an attractive interconnection network for massively parallel processing [1], [2], featuring rich symmetry properties, and a degree and diameter that are sublogarithmic on the number

*This research is supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - Brazil), under the grant No. 200392/92-1. A preliminary version of this paper has appeared in the Proceedings of the VI Brazilian Symposium on Computer Architecture and High Performance Computing (August 1994).

of nodes in the graph. These properties compare favorably with hypercube networks [3], as described in [1] and [4]. However, the earlier introduction of hypercube networks, along with their interesting characteristics, has given to such networks a considerable popularity. A number of hypercube-configured massively parallel processors (MPPs) was built in recent years [5], [6], [7], and various hypercube-compatible algorithms have been developed for these MPPs. Although efficient parallel algorithms have already been proposed for the star graph as well (e.g., sorting [8], FFT [9], linear systems solving [10]), simulation of hypercubes in star graph-configured MPPs is clearly an issue of great interest, due to the possibility of reutilization of a large number of algorithms currently supported by the hypercube.

Different techniques to simulate a single copy of a hypercube in the star graph have been proposed in [11] and [12]. There is, however, an evident trade-off between performance and efficiency in these techniques, in the sense that if a hypercube is to be simulated with small communication slowdown, a large portion of the star graph remains unused, and vice versa.

In this paper, we introduce hypercube simulation techniques that reduce this trade-off significantly. One of these techniques is referred to as *packing*, and consists of simulating multiple node-disjoint copies of the hypercube in the star graph. Such a technique achieves an efficient utilization of the star graph, while guaranteeing a small communication slowdown within each of the simulated hypercubes.

The multiple, node-disjoint hypercubes being simulated via our packing techniques can make an efficient utilization of the hardware resources of the star graph, via simultaneous execution of multiple hypercube-compatible tasks. Task migration mechanisms to handle aspects such as load balancing and fault tolerance also become possible with our packing techniques.

We also consider in this paper extensions of our packing techniques, which we refer to as *variable-dilation embeddings* and *multiple-sized packings*. Variable-dilation embeddings address the necessity of accommodating tasks requiring hypercubes that are larger than each of the copies made available through our basic packing techniques. Larger hypercubes are formed by connecting packed copies of lower-dimensional hypercubes, which assures that an efficient utilization of the star graph is still achieved. Although some performance degradation is introduced along higher dimension hypercube links connecting packed copies, our variable-dilation embeddings guarantee that the *average* communication slowdown increases by only a small factor in comparison to the communication slowdown obtained through our basic packing techniques.

Multiple-sized packings that achieve 100% utilization of the star graph are presented, providing the means by which tasks with different requirements in terms of number of nodes can be supported. Larger hypercubes can also be composed via variable-dilation techniques in this case, which suggests the flexibility of multiple-sized packings.

This paper is organized as follows. Section 2 presents some definitions and the terminology that is used in the remainder of the paper. Section 3 provides some background on hypercubes and star graphs, as well as on an embedding technique from which we derive the main results of our paper. Section 4 presents our techniques for packing hypercubes into the star graph. Section 5 considers variable-dilation embeddings and Section 6 addresses multiple-sized packings. A comparison with related work is given in Section 7 and Section 8 concludes the paper.

2 Definitions and Terminology

Let $G(k)$ be a k -dimensional graph with hierarchical structure [2], such that a $(k + 1)$ -dimensional graph $G(k + 1)$ can be obtained recursively from $c(k)$ copies of $G(k)$. Several graphs belonging to the class of

Cayley graphs have this recursive decomposition property, such as the hypercube and the star graph (which we denote by $Q(k)$ and $S(n)$, respectively). $Q(k+1)$, for example, can be obtained by connecting two $Q(k)$'s, while $S(n+1)$ can be obtained by connecting $(n+1)$ $S(n)$'s. The links connecting the $c(k)$ copies of $G(k)$ that exist within $G(k+1)$ are referred to as *dimension $(k+1)$ links*.

A graph $G(k)$ modeling a particular interconnection network can be described by a set of nodes $V(G(k))$ and a set of links $E(G(k))$, such that each node in $V(G(k))$ corresponds to a processing element (PE) and each link in $E(G(k))$ corresponds to a communication channel connecting two of these PEs. One possible mechanism for simulating $G(k)$ in a second graph $H(n)$ is referred to as *embedding $G(k)$ into $H(n)$* . The embedding of $G(k) = \{V(G(k)), E(G(k))\}$ into $H(n) = \{V(H(n)), E(H(n))\}$ consists of a mapping of $V(G(k))$ into $V(H(n))$ and of $E(G(k))$ into paths of $H(n)$. The set of nodes resulting from the mapping $M : V(G(k)) \mapsto V(H(n))$ is $V_G(H(k)) \subseteq V(H(k))$. The graph that is being embedded ($G(k)$) is referred to as the *guest* graph, while the graph that receives the embedding ($H(n)$) is referred to as the *host* graph [13]. The *load* of an embedding is the maximum number of nodes of the guest graph that are embedded in any single node of the host graph. We assume in this paper that each node of $G(k)$ is mapped to a distinct node of $H(n)$, i.e. the load of the embedding is 1. Clearly, in order to achieve such an embedding we must have $|V(H(n))| \geq |V(G(k))|$, where $|V(H(n))|$ and $|V(G(k))|$ are, respectively, the number of nodes in $H(n)$ and $G(k)$. The ratio $|V(H(n))|/|V(G(k))|$ is called the *expansion* of the embedding. The *dilation* of the embedding is the longest path in $H(n)$ used to map any link $E_{u,v} = (u, v) \in E(G(k))$, $u, v \in V(G(k))$. We refer to the expansion of an embedding with load λ and dilation d as $X(\lambda, d) = |V(H(n))|/|V(G(k))|$.

Different techniques have been proposed for embedding a hypercube $Q(k)$ into a star graph $S(n)$ [11], [12]. Embeddings with load 1 and dilations 2, 3 and 4 were considered in [11]. The corresponding expansions of these embeddings are $X(1, 2) = (2^k + 1)/2^k$, $X(1, 3) = k!/2^k$, and $X(1, 4) \leq (2^h + j)!/2^{2^h(h-2)+hj+2}$, where $h \geq 2$ and $j \leq 2^k$ are some integers such that $2^h(h-2) + hj + 2 \leq k$. Embeddings of $Q(k)$ into $S(n)$ with dilations 1, 2 and 3 were proposed in [12], with expansions $X(1, 1) = (3^k + 1)/2^k$, $X(1, 2) = (13j + 2)!/2^{11j+2}$ (where $k = 11j + 2$ and $n = 13j + 2$), and $X(1, 3) = 2^h!/2^{h2^{h-1}}$ (where $k = h2^{h-1}$ and $n = 2^h$), respectively. The dilation 1 and 2 embeddings proposed in [12] use the concept of one-to-many mappings, in which each node of the guest graph is mapped into multiple nodes of the host graph. This results in lower dilation than that obtainable with one-to-one mappings.

A major drawback of the lower-dilation embeddings of $Q(k)$ into $S(n)$ proposed in [11] and [12] is their large expansion ratios, which results in only a minor fraction of the available nodes in $S(n)$ being used. Embeddings with smaller expansions can be obtained, although at the expense of an increased dilation [12].

As it is defined, the embedding of a guest graph $G(k)$ into a host graph $H(n)$ is concerned with simulating a single copy of $G(k)$ (namely, $G_1(k)$) in $H(n)$. In embeddings with large expansion ratios (as is the case of low-dilation embeddings of a hypercube into a star graph), a more efficient utilization of $H(n)$ can be obtained if nodes in $V(H(n)) - V_{G_1}(H(n))$ are used to build additional node-disjoint copies of $G(k)$ (namely, $G_2(k), G_3(k), \dots, G_p(k)$). A number of advantages results from these multiple embeddings, as we shall explain shortly. Such a graph simulation technique is referred to as *packing $G(k)$ into $H(n)$* .

In its simplest form, the *packing* of a graph $G(k)$ into a graph $H(n)$ consists of a mapping of $V(G(k))$ into node-disjoint sets $V_{G_1}(H(n)), V_{G_2}(H(n)), \dots, V_{G_p}(H(n)) \subset V(H(n))$, and of $E(G(k))$ into paths of $H(n)$, such that p_k copies of $G(k)$ (i.e., $G_1(k), G_2(k), \dots, G_p(k)$) are packed into $H(n)$. These copies do not share any common processing node in $H(n)$ (i.e., $V_{G_a}(H(n)) \cap V_{G_b}(H(n)) = \emptyset, \forall a, b, 1 \leq a, b \leq p, a \neq b$), and the dilation of each $G_i(k)$ being simulated in $H(n)$ is d . In addition, we assume in this paper that the packing of $G(k)$ into $H(n)$ uses a one-to-one, load 1 mapping $P : V(G(k)) \mapsto V_{G_1}(H(n)), V(G(k)) \mapsto$

$V_{G_2}(H(n)), \dots, V(G(k)) \mapsto V_{G_p}(H(n))$. Note that packing is an extension of the embedding problem, in which the mapping P must be chosen such as to maximize the number of node-disjoint copies of $G(k)$ into $H(n)$. We define the *expansion of a packing* of $G(k)$ into $H(n)$ as the ratio:

$$X(\lambda, d, p_k) = \frac{|V(H(n))|}{p_k |V(G(k))|}, \quad (1)$$

where λ , d and p_k are respectively the load, the dilation, and the number of copies of $G(k)$ resulting from the packing.

Packing is a useful technique for exploiting the multitasking capability of a host graph $H(n)$, and allows concurrent execution of algorithms originally designed to run in $G(k)$. The processes running in each of the p_k copies of $G(k)$ are either different instances of the same algorithm, or instances of distinct algorithms executable in $G(k)$. Task migration strategies between the multiple copies of $G(k)$ also become possible, providing advantages such as load balancing and support of fault tolerance. Note that we will not be discussing task migration or node allocation strategies in this paper (see [14], [15] and [16] for more on this topic). Rather, we present efficient packing techniques that can support these strategies in star graphs when multiple hypercubes are being simulated.

A well-devised packing technique can be readily extended to support *variable-dilation embeddings*. A variable-dilation embedding consists of simulating a $(k + \ell)$ -dimensional graph $G(k + \ell)$, $\ell > 0$, in $H(n)$ by connecting $c(k \rightarrow k + \ell) = \prod_{i=1}^{\ell} c(k + i - 1)$ copies of $G(k)$ (i.e., $c(k \rightarrow k + \ell)$ is the number of packed copies of $G(k)$ required to form a $(k + \ell)$ -dimensional graph $G(k + \ell)$). We refer to such an embedding as having variable dilation due to the fact that the maximum number of links in $H(n)$ that are required to simulate any given link (u, v) of $G(k + \ell)$ depends on the dimension of (u, v) . Let $\overline{d_{k+\ell}} = [d_1, d_2, \dots, d_i, \dots, d_{k+\ell}]$ be a *dilation vector* defining the maximum number of links in $H(n)$ that is required to simulate any of the dimension i links of $G(k + \ell)$, $1 \leq i \leq k + \ell$. We denote the largest dilation along any dimension of $G(k + \ell)$, in its corresponding variable-dilation embedding into $H(n)$, by d_{max} (i.e., $d_{max} = \max(d_i)$, for $1 \leq i \leq k + \ell$).

Assume that the communication slowdown $\eta(G(k) \mapsto H(n))$ resulting from simulating $G(k)$ in $H(n)$ is equal to the dilation of the embedding, namely $\eta(G(k) \mapsto H(n)) = d$. A conservative estimate for the communication slowdown of a variable-dilation embedding $V : G(k + \ell) \mapsto H(n)$ is d_{max} . Many algorithms, however, use a limited number of dimensions at any given step of their execution, resulting in a smaller slowdown. Notably, algorithms following the SIMD computational model require that all nodes in the interconnection network operate in a lockstep fashion, causing a unique dimension to be used at every step of the algorithm. Note that the slowdown for an algorithm devised according to the MIMD computational model may also be smaller than d_{max} , depending on how the dimensions of $G(k + \ell)$ are used and on the nature of code dependencies arising during the execution of the algorithm.

Let A be an algorithm devised to run under the SIMD computational model in an MPP with interconnection network $G(k + \ell)$. Let the number of steps using dimension i of $G(k + \ell)$, $1 \leq i \leq k + \ell$, be τ_i . The communication slowdown in A 's execution, resulting from a variable-dilation embedding of $G(k + \ell)$ into $H(n)$, is

$$\eta(G(k + \ell) \mapsto H(n)) = \frac{\sum_{i=1}^{k+\ell} \tau_i d_i}{\sum_{i=1}^{k+\ell} \tau_i} \quad (2)$$

If we make the simplifying assumption that A is such that $\tau_i = \tau$, $\forall i$, Equation 2 reduces to

$$\eta(G(k+\ell) \mapsto H(n)) = d_{avr} = \frac{1}{k+\ell} \sum_{i=1}^{k+\ell} d_i, \quad (3)$$

where d_{avr} is referred to as the *average dilation* of the variable-dilation embedding of $G(k+\ell)$ into $H(n)$.

Note that a major advantage of variable-dilation embeddings, as opposed to conventional embedding methods, is that the communication slowdown can be made significantly smaller *on the average*. A conventional embedding of a guest graph $G(k+\ell)$ into a host graph $H(n)$ often requires a large dilation along all dimensions of $G(k+\ell)$ if the expansion ratio of the embedding is at premium. Consequently, a larger communication slowdown results. On the other hand, variable-dilation embeddings can still achieve low expansion ratios with smaller slowdown by forcing the dilation along the first k dimensions of $G(k+\ell)$ to be $d < d_{max}$. As explained earlier, this is achieved by generating the embedding of $G(k+\ell)$ from packed copies of lower dimensional graphs $G(k)$. The expansion of a variable-dilation embedding is:

$$X(\lambda, \overline{d_{k+\ell}}) = \frac{|V(H(n))|}{|V(G(k+\ell))|} = \frac{|V(H(n))|}{c(k \rightarrow k+\ell)|V(G(k))|} \quad (4)$$

The concept of achieving an efficient utilization of a host graph $H(n)$ by packing multiple copies of a guest graph $G(k)$ can be extended to an interesting technique referred to as a *multiple-sized packing*. In this case, graphs of various sizes $|V(G(k))|, |V(G(k+1))|, \dots, |V(G(k+j))|, \dots, |V(G(k+\ell))|$ are packed into $H(n)$. Let a given multiple-sized packing P_m have $p_{k+j|m}$ copies of a $(k+j)$ -dimensional graph $G(k+j)$, $0 \leq j \leq \ell$. As in the equal-sized case, we require that all copies of a given graph $G(k+j)$ are node-disjoint in P_m . In addition, copies of $G(k+j)$ and $G(k+j')$, $j \neq j'$, $0 \leq j, j' \leq \ell$, are also node-disjoint in P_m .

Let $\overline{P_m} = \{p_{k|m}, \dots, p_{k+j|m}, \dots, p_{k+\ell|m}\}$ be a vector specifying the number of copies of $G(k+j)$ in P_m , for $0 \leq j \leq \ell$. In Section 6, we show that a multiple-sized packing of hypercubes into the star graph, P_m , can be transformed into another multiple-sized packing P'_m , such that the total number of nodes used in both P_m and P'_m is the same, but the distribution of hypercube sizes in P_m and P'_m are different (i.e., $\overline{P_m} \neq \overline{P'_m}$). Larger hypercubes in P'_m can be obtained by connecting smaller hypercubes in P_m via variable-dilation embedding techniques. Accordingly, we can create smaller hypercubes in P'_m by splitting larger hypercubes in P_m .

To fully specify a given multiple-sized packing P_m , we consider the general case where each copy of a graph $G(k+j)$ is embedded into $H(n)$ with variable-dilation. Let $\overline{d_{k+j}}$ be the dilation vector referring to each $G(k+j)$ that is packed into $H(n)$, and let $\overline{D_m} = \{\overline{d_k}, \dots, \overline{d_{k+j}}, \dots, \overline{d_{k+\ell}}\}$ be a set specifying all dilation vectors $\overline{d_{k+j}}$, for $0 \leq j \leq \ell$. The expansion of a given multiple-sized packing P_m is

$$X(\lambda, \overline{D_m}, \overline{P_m}) = \frac{|H(n)|}{\sum_{j=0}^{\ell} (p_{k+j|m} |G(k+j)|)} = \frac{|H(n)|}{\sum_{j=0}^{\ell} (p_{k+j|m} c(k \rightarrow k+j) |G(k)|)} \quad (5)$$

Multiple-sized packings allow tasks with different node requirements to be allocated proper guest graphs $G(k+j)$ in $H(n)$. A smaller communication slowdown is still guaranteed in the case of larger guest graphs, which can be built in $H(n)$ via variable-dilation embeddings.

3 Background

3.1 The hypercube

A k -dimensional hypercube¹ graph $Q(k) = \{V(Q(k)), E(Q(k))\}$ contains 2^k nodes which are labeled with binary strings of length k . In this paper, we use the digits $\{0, 1\}$ to form the node labels of $Q(k)$. A node $\phi = q_1q_2 \dots q_i \dots q_k$ is connected to k distinct nodes, respectively labeled with strings $\phi_i = q_1q_2 \dots \bar{q}_i \dots q_k$, $1 \leq i \leq k$. In other words, node ϕ is connected to other k nodes whose labels are the binary strings resulting from complementing the digit in position i in ϕ , where $1 \leq i \leq k$ [3]. A 3-dimensional hypercube is shown in Figure 1. Note that a link connecting a node $\phi = q_1q_2 \dots q_i \dots q_k$ to a node $\phi_i = q_1q_2 \dots \bar{q}_i \dots q_k$ is labeled i to indicate a connection along the i^{th} dimension of $Q(k)$.

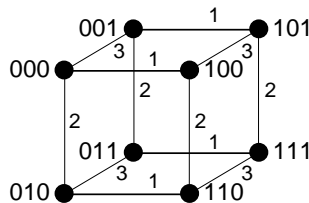


Figure 1: A 3-dimensional hypercube $Q(3)$

$Q(k)$ is a regular graph with degree $\delta(Q(k)) = k$ and diameter $d(Q(k)) = k$ [3]. $Q(k)$ is node- and edge-symmetric, has hierarchical structure and supports node communication via simple routing algorithms.

3.2 The star graph

An n -dimensional star graph $S(n) = \{V(S(n)), E(S(n))\}$ contains $n!$ nodes which are labeled with the $n!$ possible permutations of n distinct symbols. In this paper, we use the digits $\{1, 2, \dots, n\}$ to label the nodes of $S(n)$. A node $\pi = p_1p_2 \dots p_i \dots p_n$ is connected to $(n-1)$ distinct nodes, respectively labeled with permutations $\pi_i = p_i p_2 \dots p_{i-1} p_1 p_{i+1} \dots p_n$, $2 \leq i \leq n$. In other words, node π is connected to other $(n-1)$ nodes whose labels are the permutations resulting from exchanging the digit in position i in π with the first digit of π , where $2 \leq i \leq n$ [1], [2]. A 4-dimensional star graph is shown in Figure 2. A link connecting a node $\pi = p_1p_2 \dots p_i \dots p_n$ to a node $\pi_i = p_i p_2 \dots p_{i-1} p_1 p_{i+1} \dots p_n$ is labeled i to indicate a connection along the i^{th} dimension of $S(n)$. $S(n)$ is a regular graph with degree $\delta(S(n)) = n-1$ and diameter $d(S(n)) = \lfloor 3(n-1)/2 \rfloor$ [1]. Similarly to $Q(k)$, $S(n)$ is a hierarchical graph featuring simple routing and both node and edge symmetry.

3.3 Embedding an $(n-1)$ -dimensional mesh into $S(n)$

The packing and embedding techniques presented in this paper use a two-step mapping, in which hypercubes are initially packed into an $(n-1)$ -dimensional mesh of size $2 \times 3 \times \dots \times (n-1) \times n$, which we denote by $M(n-1) = \{V(M(n-1)), E(M(n-1))\}$. $M(n-1)$ is then embedded into $S(n)$ with load 1, dilation 3, and expansion 1 via the mapping proposed by Ranka et al. in [18]. A brief review of this mapping is presented below.

We assume that the nodes of $M(n-1)$ are labeled with an $(n-1)$ -digit vector $m_1 m_2 \dots m_{n-1}$, $0 \leq m_i \leq s_i - 1$, where $s_i = i + 1$ is the size of the mesh along the i^{th} dimension.

¹For simplicity, we use the term *hypercube* to refer to the well-known *binary hypercube*, which is in fact a graph belonging to the larger class of generalized hypercube graphs [17].

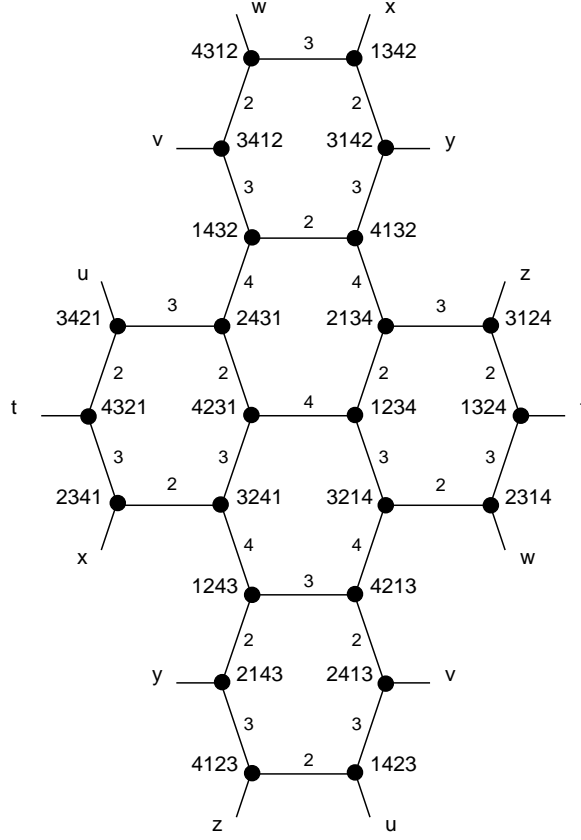


Figure 2: A 4-dimensional star graph $S(4)$

The mapping of a node $\omega = m_1 m_2 \dots m_{n-1}$, $\omega \in V(M(n-1))$, onto a node $\pi = p_1 p_2 \dots p_n$, $\pi \in V(S(n))$, is accomplished by the algorithm shown below. We assume that the identity node of $M(n-1)$ (i.e., $00 \dots 0$) maps onto the identity node of $S(n)$ (i.e., $12 \dots n$). Note that in Algorithm 1 the labels of the mesh and the star nodes are represented with vectors, such that $\omega = m[]$ and $\pi = p[]$.

Algorithm 1 (Mapping $M(n-1)$ onto $S(n)$):

```

mesh_to_star (int m[ ], int n, int p[ ])
{
  int i, j, temp;
  for (i = 1; i ≤ n; i++) p[i] = i;
  for (i = 1; i < n; i++)
    for (j = 0; j < m[i]; j++) {
      temp = p[i - j + 1];
      p[i - j + 1] = p[i - j];
      p[i - j] = temp;
    }
}

```

Algorithm 1 initially sets permutation π to $p[] = 12 \dots n$. The next step of the algorithm consists of an iterative inspection of the $(n-1)$ coordinates of the mesh node ($\omega = m[]$). Assuming that the coordinate of

ω along the i^{th} dimension is $m[i]$, the i^{th} iteration of the externally nested *for* loop results in the following sequence of transpositions:

$$(p[i + 1] \leftrightarrow p[i]), (p[i] \leftrightarrow p[i - 1]), \dots, (p[i - m[i] + 2] \leftrightarrow p[i - m[i] + 1])$$

Let the transposition of two digits $p[r]$, $p[s]$ in π be denoted by $(r\ s)$ (i.e., $(r\ s)$ corresponds to the exchange of the digits occupying the r^{th} and the s^{th} positions in permutation π). Table 1 lists the sequences of transpositions used by Algorithm 1 along the $(n - 1)$ dimensions of the mesh. Note that if the coordinate of the mesh node along dimension i is $m[i]$, then only the first $m[i]$ transpositions of the sequence corresponding to dimension i are used.

<i>Dimension (i)</i>	<i>Sequence of transpositions</i>
1	(2 1)
2	(3 2) (2 1)
\vdots	\vdots
$n - 1$	$(n\ n - 1) (n - 1\ n - 2) \dots (2\ 1)$

Table 1: Sequences of transpositions used by Algorithm 1

As an example, assume the mapping of node $\omega = 103 \in M(3)$ onto a node $\pi \in S(4)$. Initially, Algorithm 1 sets $\pi = 1234$. Since $m[1] = 1$, a (21) transposition is performed on 1234, giving 2134. Next, the algorithm examines the coordinate of ω along the 2nd dimension ($m[2]$). Since $m[2] = 0$, no transpositions are performed at this step. Next, $m[3]$ is examined, resulting in a sequence of transpositions (43) (32) (21). Such a sequence affects π as shown below:

$$2134 \xrightarrow{(43)} 2143 \xrightarrow{(32)} 2413 \xrightarrow{(21)} 4213$$

Hence, $\omega = 103$ is mapped onto node $\pi = 4213$. Figure 3 shows the complete mapping of a 3-dimensional mesh onto $S(4)$.

The mapping algorithm described in this paper differs slightly from that proposed in [18] with respect to the definition of a transposition $(r\ s)$. In Algorithm 1, $(r\ s)$ corresponds to an exchange of the digits occupying the r^{th} and the s^{th} positions in permutation π . In the corresponding algorithm described in [18], $(r\ s)$ corresponds to an exchange of *digits* r and s in π . Both approaches, however, result in a load 1, dilation 3, expansion 1 embedding of $M(n - 1)$ onto $S(n)$.

The following lemma combines results of [18] that are relevant to our packing techniques. Although such results are proved in [18], we present a simpler proof which is useful for other lemmas that will be stated later in this paper.

Lemma 1 *Any two nodes $\omega, \omega_i \in V(M(n - 1))$ which are adjacent over the i^{th} dimension of $M(n - 1)$, $1 \leq i \leq n - 1$, are connected by a path containing either 1 or 3 links in the corresponding embedding into $S(n)$.*

Proof: Without loss of generality, assume that the i^{th} coordinates of ω and ω_i (respectively, $m[i]$ and $m'[i]$) are such that $m[i] = m'[i] - 1$. By inspection of Algorithm 1 and Table 1, we note that the mapping of $\omega \in V(M(n - 1))$ onto $\pi \in V(S(n))$ uses a sequence of transpositions of the form:

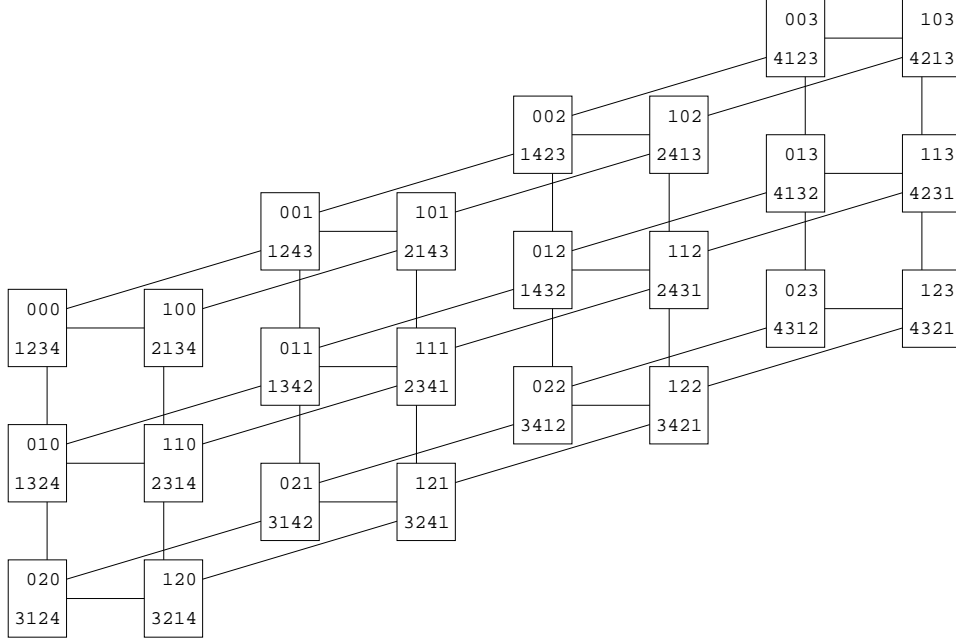


Figure 3: Mapping of $M(3)$ onto $S(4)$

$$\sigma = (a\ b)(c\ d)\dots(i\ j)(m\ n)\dots(y\ z)$$

Accordingly, ω_i is mapped onto π_i via a sequence of transpositions of the form:

$$\sigma_i = (a\ b)(c\ d)\dots(i\ j)(k\ l)(m\ n)\dots(y\ z)$$

Note that identical transpositions are used in σ and σ_i , except for the fact that σ_i has one transposition more than does σ (namely, $(k\ l) = ((i - m'[i] + 2)\ (i - m'[i] + 1))$). Hence, the mapping of ω and ω_i is initially carried out identically, with transpositions $(a\ b)(c\ d)\dots(i\ j)$ being used in both σ and σ_i . At the i^{th} iteration of the externally nested *for* loop of Algorithm 1, however, the extra transposition $(k\ l)$ is used only in the mapping sequence of ω_i .

Let the digits occupying the k^{th} and the l^{th} positions of the permutation resulting from applying transpositions $(a\ b)(c\ d)\dots(i\ j)$ to the identity $123\dots n$ be respectively d_1 and d_2 . Therefore, $(k\ l)$ moves d_1 into d_2 's position, and d_2 into d_1 's position. Since the remaining transpositions in σ , σ_i (i.e., $(m\ n)\dots(y\ z)$) are also identical, π and π_i will differ only in the final positions occupied by digits d_1 and d_2 . Let these positions be α and β , such that if in π we have $p[\alpha] = d_1$ and $p[\beta] = d_2$, in π_i we have $p[\alpha] = d_2$ and $p[\beta] = d_1$. Hence, routing from π to π_i in $S(n)$ can be easily accomplished with a single transposition $(\alpha\ \beta)$.

Naturally, such a transposition must be properly accomplished by means of star operations that are available in $S(n)$. Let g be the star operation performed along the g^{th} dimension of $S(n)$ (i.e., g exchanges the first and the g^{th} digit of a permutation of n digits). Assume that transposition $(\alpha\ \beta)$ is ordered such that $\alpha < \beta$. Therefore, $(\alpha\ \beta)$ can be minimally executed by the following sequences of star operations [1], [19]:

$$(\alpha\ \beta) \equiv \begin{cases} \beta & , \text{ if } \alpha = 1 \\ \beta \rightarrow \alpha \rightarrow \beta \equiv \alpha \rightarrow \beta \rightarrow \alpha & , \text{ if } \alpha, \beta \neq 1 \end{cases} \quad (6)$$

Note that transposition $(\alpha\ \beta)$ requires 1 star operation if $\alpha = 1$ and 3 star operations otherwise. \square

4 Packing Hypercubes into Star Graphs

4.1 Fundamentals

The packing techniques which are presented in this paper take advantage of the regular structure of $M(n-1)$ to achieve an efficient utilization of the nodes of $S(n)$. An important result on which our techniques are based is given below:

Lemma 2 $Q(k)$ can be embedded into $M(n-1)$ with load 1, dilation 1, for $k \leq n-1$.

Proof: A basic requirement for a load 1, dilation 1 embedding of $Q(k)$ into $M(n-1)$ is that the degree of any node in $M(n-1)$ must be greater than or equal to the degree of $Q(k)$. Since the degree of the nodes in $M(n-1)$ is at least $(n-1)$ and $\delta(Q(k)) = k$, this condition is satisfied for all $k \leq n-1$.

To complete the proof, we give a straightforward mapping of the nodes of $Q(k)$ onto the nodes of $M(n-1)$ (Algorithm 2). Algorithm 2 does a one-to-one mapping in which mesh nodes whose first k coordinates are either 0 or 1 are selected to embed $Q(k)$. Note that such a mapping is possible because the size of $M(n-1)$ along any dimension is at least 2. A null value is selected for the remaining $(n-1-k)$ coordinates of these nodes, although different values could have been used as well. Note that $Q(k)$ can be seen as a k -dimensional mesh of size $s_1 \times s_2 \times \dots \times s_k = 2 \times 2 \times \dots \times 2$, which corresponds to using a limited range of the coordinates available in $M(n-1)$. \square

Algorithm 2 (Mapping $Q(k)$ onto $M(n-1)$):

```

cube_to_mesh (int q[ ], int k, int n, int m[ ])
{
    int i;
    for (i = 1; i ≤ k; i++) m[i] = q[i];
    for (i = k + 1; i < n; i++) m[i] = 0;
}

```

A natural extension of Algorithm 2 consists of selecting proper values for the coordinates of the mesh nodes, such that node-disjoint mappings of $Q(k)$ onto $M(n-1)$ result. This is exactly the basis for the packing techniques we present in this section. Naturally, our ultimate goal is to pack hypercubes into the star graph. This can be accomplished after hypercubes have been packed into the mesh, via the one-to-one mapping of the nodes of $M(n-1)$ into $S(n)$ described in Algorithm 1.

We introduce our techniques by considering initially the packing of $Q(n-1)$ into $S(n)$, and the packing of $Q(n-2)$ into $S(n)$. A general algorithm that packs $Q(n-t)$ into $S(n)$, for $1 \leq t \leq \lfloor (n+1)/2 \rfloor$, is presented later in this section, followed by an expansion 1 packing of $Q(\lfloor n/2 \rfloor)$ into $S(n)$.

From the viewpoint of how the hypercubes are packed into the mesh, we classify our packings as *symmetric* or *asymmetric*. Symmetric packings are those in which a dimension a link of every packed hypercube $Q(n-t)$, $1 \leq a \leq n-t$, is always assigned to a dimension b link of $M(n-1)$, $1 \leq b \leq n-1$. Accordingly, asymmetric packings are those in which dimension a links of two different packed hypercubes $Q(n-t)$ and $Q'(n-t)$, $1 \leq a \leq n-t$, are assigned respectively to dimension b and c links of $M(n-1)$, $1 \leq b, c \leq n-1$, where b and c are not necessarily equal. Note, however, that all dimension a links of a given copy of $Q(n-t)$ are mapped onto a unique dimension of $M(n-1)$ in an asymmetric packing.

The dimension assignment rules that characterize a given packing technique are not preserved when $M(n-1)$ is ultimately embedded into $S(n)$ via Algorithm 1. In other words, in both symmetric and

asymmetric packings, a dimension a link of any packed hypercube $Q(n-t)$, $1 \leq a \leq n-t$, is assigned either to a dimension b link of $S(n)$, or to a path of length 3 in $S(n)$ of the form $b \rightarrow c \rightarrow b$, where b, c can be any of the dimensions of $S(n)$ ($2 \leq b, c \leq n$). Although the symmetry (or asymmetry) of a particular technique used to pack $Q(n-t)$ into $S(n)$ can not be distinguished unless for the intermediary step where the hypercubes are packed into $M(n-1)$, we use throughout the paper the terms *symmetric* (or *asymmetric*) *packings of $Q(n-t)$ into $S(n)$* .

Symmetric packings provide a very regular arrangement of the copies of $Q(n-t)$ in $M(n-1)$. This is particularly useful for another hypercube simulation technique we will describe later in this paper, namely variable dilation embeddings. Asymmetric packings, on the other hand, allow a greater number of copies of $Q(n-t)$ to be packed into $M(n-1)$ (and, consequently, into $S(n)$), and should therefore be considered whenever a very efficient utilization of the nodes of $S(n)$ is desired. As we shall see, asymmetric packings are advantageous in cases where symmetric packings would result in expansion greater than 1, and often can provide 100% or close to 100% utilization of $S(n)$. Hence, an asymmetric packing will often be the method of choice to pack hypercubes into the star graph.

It is also important to shed some light on the issue of how the role of communication between hypercube nodes is accomplished in our packing techniques. To do so, we consider initially the intermediary step used in all packing techniques presented in this section (i.e., a dilation 1, load 1 packing of $Q(n-t)$ into $M(n-1)$). In an $M(n-1)$ -configured MPP, the communication within a packed copy of $Q(n-t)$ mimics exactly the communication pattern found in an $Q(n-t)$ -configured MPP. In other words, any algorithm designed to run in $Q(n-t)$ would find all the communication resources needed for its execution self-contained within the copy of $Q(n-t)$. Furthermore, since both the load and the dilation of the packing are 1, any routing required by the algorithm can be accomplished in the packed copy under identical conditions, as far as routing strategies and end-to-end distances are concerned.

When the nodes of $M(n-1)$ are mapped onto $S(n)$ via Algorithm 1, however, these conditions do not hold any more. We recall that Algorithm 1 provides a one-to-one mapping, and therefore from the viewpoint of processing load each node of $Q(n-t)$ will be ultimately mapped onto a unique node of $S(n)$. However, according to Lemma 1 two nodes that are adjacent in $Q(n-t)$ will be connected by a path of length 1 or 3 in $S(n)$. In the cases where the distance between adjacent nodes remains 1, a trivial routing along a single link of $S(n)$ suffices. Now, assume that $q[]$ and $q'[]$ are two adjacent nodes in $Q(n-t)$ which are connected by a path of length 3 in $S(n)$. In this case, two additional nodes will be needed to forward messages between the corresponding pair of nodes in $S(n)$. Furthermore, these routing nodes may lie within the same packed copy of $Q(n-t)$ which contains $q[]$ and $q'[]$, or may be located in one or two other copies nearby.

Hence, an important assumption made in this paper is that any node of $S(n)$ being used as one of the processing elements (PEs) of a given copy of $Q(n-t)$ may also be used to perform routing. As explained earlier, each node of $S(n)$ should not only provide communication services to requests that arise during the execution of a particular algorithm running within a given copy of $Q(n-t)$, but also serve as a routing device for messages occurring between nodes of a neighboring $Q(n-t)$, whenever this should be needed. This dual role can be supported efficiently if each node in $S(n)$ is implemented with a separate routing chip that offloads communication-related tasks from the actual processor. In fact, this implementation model is often found in many commercial MPPs, such as the Intel Paragon XP/S.

4.2 Packing $Q(n-1)$ into $S(n)$

$Q(n-1)$ is the largest hypercube considered in this paper as far as load 1, dilation 3 packings into $S(n)$ are concerned. In addition, the technique used to pack $Q(n-1)$ into $S(n)$ is *symmetric*.

Theorem 1 *It is possible to symmetrically pack p_{n-1} node-disjoint copies of $Q(n-1)$ into $S(n)$, with load 1, dilation 3, and expansion $X(1, 3, p_{n-1})$, where*

$$p_{n-1} = \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \quad \text{and} \quad X(1, 3, p_{n-1}) = \frac{n}{2^{n-1}} \cdot \binom{n-1}{\lfloor n/2 \rfloor}$$

Proof: The size of $M(n-1)$ along the i^{th} dimension is $s_i = i + 1$ (i.e., the i^{th} coordinate of the nodes of $M(n-1)$ ranges from 0 to i). It is possible to partition $M(n-1)$ along dimension i into $\lfloor s_i/2 \rfloor$ $(n-1)$ -dimensional submeshes whose size along that dimension is at least 2. If this process is repeated for all dimensions of $M(n-1)$, the resulting number of $(n-1)$ -dimensional submeshes of size at least 2 along any dimension is

$$p_{n-1} = \left\lfloor \frac{s_1}{2} \right\rfloor \times \left\lfloor \frac{s_2}{2} \right\rfloor \times \dots \times \left\lfloor \frac{s_{n-1}}{2} \right\rfloor = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor!$$

Due to the partitioning process, these submeshes are node-disjoint. In addition, by applying Lemma 2 we note that it is possible to embed $Q(n-1)$ with load 1, dilation 1 into each of these p_{n-1} submeshes. By simply using the same dimension assignment for each copy of $Q(n-1)$, we symmetrically pack p_{n-1} node-disjoint copies of $Q(n-1)$ into $M(n-1)$, with load 1 and dilation 1. If we now embed $M(n-1)$ into $S(n)$ using the mapping given by Algorithm 1, a load 1, dilation 3 packing results. The fact that the load remains 1 follows from the observation that Algorithm 1 provides a one-to-one mapping [18]. The dilation, however, increases from 1 to 3 as a direct consequence of Lemma 1.

To complete the proof, we note that the expansion of the packing can be obtained by direct application of Equation 1, noting that $|S(n)| = n!$ and $|Q(n-1)| = 2^{n-1}$:

$$X(1, 3, p_{n-1}) = \frac{n!}{p_{n-1} \cdot 2^{n-1}} = \frac{n!}{\left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \cdot 2^{n-1}} = \frac{n}{2^{n-1}} \cdot \binom{n-1}{\lfloor n/2 \rfloor}$$

□

We now present an algorithm that symmetrically packs $Q(n-1)$ into $S(n)$:

Algorithm 3 (Symmetric packing of $Q(n-1)$ into $S(n)$):

```

spack_cube_n_1 (int q[ ], int C, int n, int p[ ])
{
  int i, offset[ ], m[ ];
  for (i = 1; i < n; i++) m[i] = q[i];
  for (i = 3; i < n; i++) {
    offset[i] = 2 * ( ( ( C / ( ( ( i / 2 ) ! * ( ( i - 1 ) / 2 ) ! ) ) ) mod ( ( ( i + 1 ) / 2 ) ) );
    m[i] = m[i] + offset[i];
  }
  mesh_to_star (m[ ], n, p[ ])
}

```

Initially, the algorithm copies the coordinates of the hypercube node ($q[\]$) onto the coordinates of a mesh node $m[\]$. Let $q[\]$ be a node belonging to the C^{th} packed hypercube, $0 \leq C \leq p_{n-1} - 1$. Algorithm 3 computes an offset vector ($offset[\]$) to correctly map any node belonging to the C^{th} hypercube onto a node-disjoint submesh. This offset is added to $m[\]$, completing the mapping onto $M(n-1)$. The resulting mesh coordinate is finally mapped onto $S(n)$ via a call to Algorithm 1. Table 2 shows the offset vectors that are required to pack $Q(5)$ into $S(6)$. Note that, according to Theorem 1, it is possible to symmetrically pack 12 copies of $Q(5)$ into $S(6)$. To further illustrate the packing of $Q(n-1)$ into $S(n)$, Figure 4a shows how 4 copies of $Q(4)$ can be symmetrically packed into $M(4)$. The corresponding packing into $S(5)$ follows by simple application of Algorithm 1.

Hypercube number (C)	0	1	2	3	4	5	6	7	8	9	10	11
$offset[3] = 2(C \bmod 2)$	0	2	0	2	0	2	0	2	0	2	0	2
$offset[4] = 2 \left(\left\lfloor \frac{C}{2} \right\rfloor \bmod 2 \right)$	0	0	2	2	0	0	2	2	0	0	2	2
$offset[5] = 2 \left(\left\lfloor \frac{C}{4} \right\rfloor \bmod 3 \right)$	0	0	0	0	2	2	2	2	4	4	4	4

Table 2: Mesh coordinate offsets used in the packing of $Q(5)$ into $S(6)$

Now, consider that Algorithm 3 is used to map any 2 hypercube nodes $q[\]$ and $q'[\]$, which are neighbors along the i^{th} dimension of $Q(n-1)$. Without loss of generality, we assume that $q[i] = q'[i] + 1$. The corresponding mesh nodes, $m[\]$ and $m'[\]$, are neighbors along the i^{th} dimension of $M(n-1)$, since Algorithm 3 results in $m[i] = m'[i] + 1$. Hence, the packing accomplished via Algorithm 3 is symmetric.

4.3 Packing $Q(n-2)$ into $S(n)$

Intuitively, denser packings of $Q(k)$ into $S(n)$ seem to be easier to be achieved with smaller hypercubes. By considering the packing of $Q(n-2)$ into $S(n)$, we shall show that such denser packings can be obtained not only due to the smaller size of the hypercubes being packed, but also due to the possibility of using asymmetric packing techniques. Initially, we consider *symmetric* packings of $Q(n-2)$ into $S(n)$ as follows:

Theorem 2 *It is possible to symmetrically pack p_{n-2} node-disjoint copies of $Q(n-2)$ into $S(n)$, with load 1, dilation 3, and expansion $X(1, 3, p_{n-2})$, where*

$$p_{n-2} = 3 \cdot \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \quad \text{and} \quad X(1, 3, p_{n-2}) = \frac{n}{3 \cdot 2^{n-2}} \cdot \binom{n-1}{\lfloor n/2 \rfloor}$$

Proof: Partitioning $M(n-1)$ into $\lfloor s_i/2 \rfloor$ submeshes along dimension i , as described in the proof of Theorem 1, results in unused nodes whenever s_i is odd. Notably, the largest underutilization of $M(n-1)$ occurs for $i = 2$, when $1/3$ of the nodes are discarded by the partitioning process. These nodes can actually be used while packing $Q(n-2)$ into $S(n)$, if we partition $M(n-1)$ into $(n-2)$ -dimensional submeshes instead.

We modify the partitioning of $M(n-1)$ as follows. Initially, $M(n-1)$ is partitioned into 3 $(n-2)$ -dimensional meshes along dimension 2. For the remaining dimensions, the partitioning process occurs as

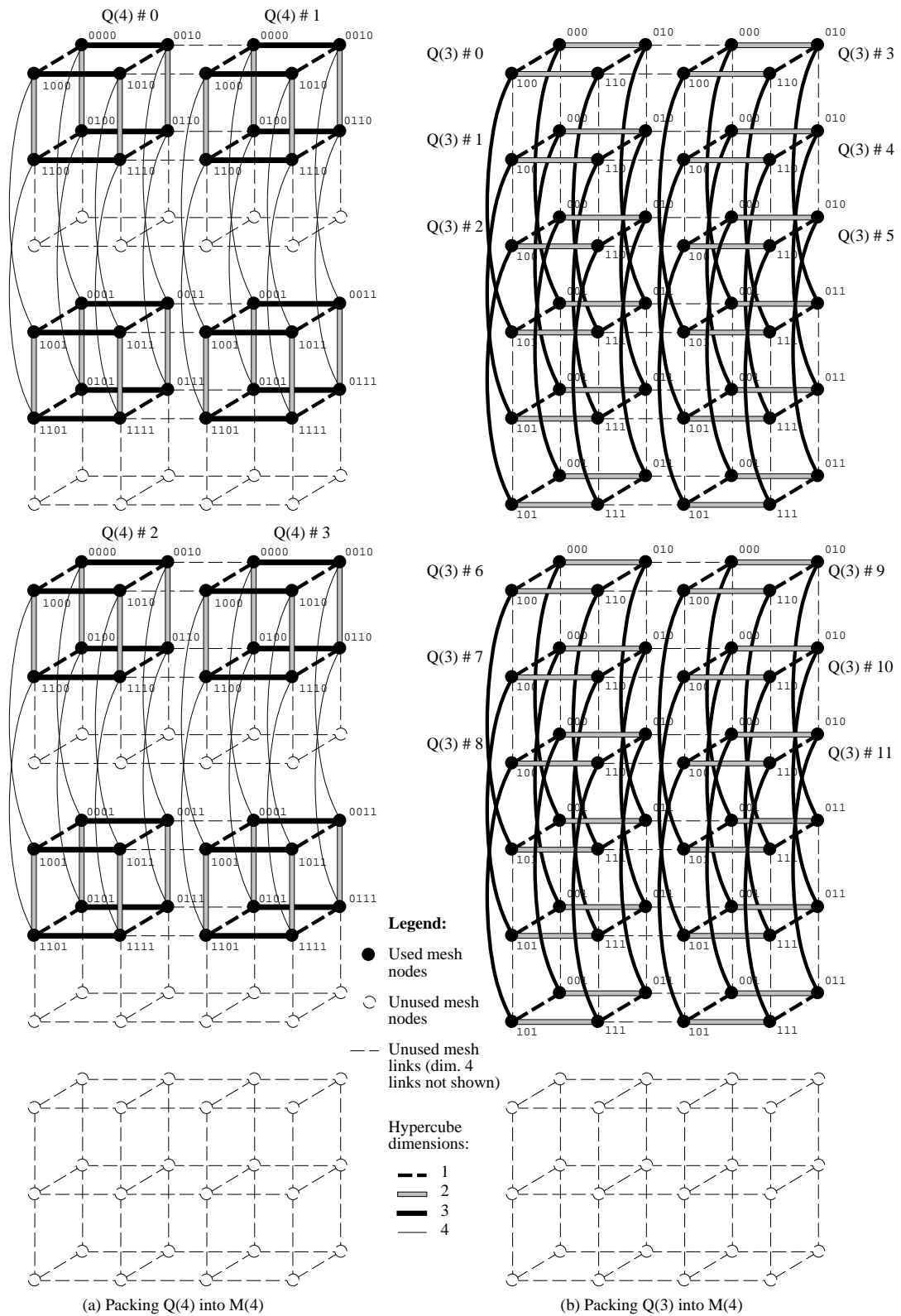


Figure 4: Symmetric packings of $Q(4)$ and $Q(3)$ into $M(4)$

described in Theorem 1. Hence, the resulting number of $(n - 2)$ -dimensional submeshes of size at least 2 along any dimension is

$$p_{n-2} = \left\lfloor \frac{s_1}{2} \right\rfloor \times 3 \times \left\lfloor \frac{s_3}{2} \right\rfloor \times \dots \times \left\lfloor \frac{s_{n-1}}{2} \right\rfloor = \left\lfloor \frac{2}{2} \right\rfloor \times 3 \times \left\lfloor \frac{4}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = 3 \cdot \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor!$$

Due to the partitioning process, these submeshes are node-disjoint. In addition, by applying Lemma 2 we note that it is possible to embed $Q(n - 2)$ with load 1, dilation 1 into each of these p_{n-2} submeshes. Hence, it is possible to pack p_{n-2} node-disjoint copies of $Q(n - 2)$ into $M(n - 1)$, with load 1 and dilation 1. Furthermore, the packing can be done symmetrically by using the same dimension assignment for every copy of $Q(n - 2)$. If we now embed $M(n - 1)$ into $S(n)$ using the mapping given by Algorithm 1, a load 1, dilation 3 packing results. The derivation of the expansion ratio is done as in the proof of Theorem 1. \square

A straightforward extension of Algorithm 3 that symmetrically packs $Q(n - 2)$ into $S(n)$ follows:

Algorithm 4 (Symmetric packing of $Q(n - 2)$ into $S(n)$):

```

spack_cube_n_2 (int q[ ], int C, int n, int p[ ])
{
  int i, offset[ ], m[ ];
  m[1] = q[1];
  m[2] = C mod 3;
  for (i = 3; i < n; i++) {
    offset[i] = 2 * ( ( ( C / ( 3 * floor(i/2)! * floor((i-1)/2)! ) ) ) mod floor((i+1)/2) );
    m[i] = q[i-1] + offset[i];
  }
  mesh_to_star (m[ ], n, p[ ])
}

```

The variables used in the above algorithm are identical to those of Algorithm 3. An offset vector ($offset[]$) is computed to map any node $q[]$ belonging to the C^{th} hypercube onto a node-disjoint submesh. Note that $q[] \in Q(n - 2)$ is a node label containing $(n - 2)$ coordinates, and $m[] \in M(n - 1)$ is a node label containing $(n - 1)$ coordinates. Following the partitioning mechanism described in the proof of Theorem 2, we make $m[1] = q[1]$, and $m[i] = q[i - 1] + offset[i]$, for $3 \leq i < n$. To complete the mapping of $q[]$ onto $m[]$, $m[2]$ is simply assigned a value corresponding to the offset ($C \bmod 3$). Finally, $m[]$ is mapped onto $S(n)$ via a call to Algorithm 1. The packing accomplished by Algorithm 4 can be shown to be symmetric by the same reasoning used in the description of Algorithm 3. To illustrate the packing of $Q(n - 2)$ into $S(n)$, Figure 4b shows how 12 copies of $Q(3)$ can be symmetrically packed into $M(4)$. The corresponding packing into $S(5)$ follows by simple application of Algorithm 1.

A careful inspection of Figure 4b reveals that 2 additional copies of $Q(3)$ could be packed with load 1, dilation 1 into $M(4)$ (and consequently, with load 1, dilation 3 into $S(5)$). For $n = 5$, the partitioning process described in the proof of Theorem 2 results in an unused 3-dimensional submesh of size $2 \times 3 \times 4$, which can hold 2 $Q(3)$'s as shown in Figure 5.

Note, however, that these 2 additional hypercubes are packed with a different dimension assignment when compared to the 12 copies shown in Figure 4b. As defined earlier in this section, we refer to this resulting packing of $12 + 2 = 14$ copies of $Q(3)$ into $M(4)$ as an *asymmetric packing*.

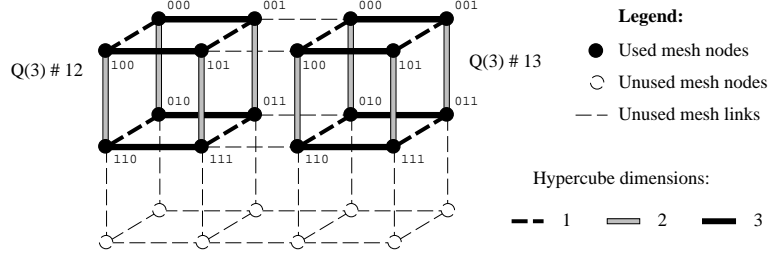


Figure 5: Packing two additional copies of $Q(3)$ into $M(4)$ asymmetrically

Theorem 3 *It is possible to asymmetrically pack p_{n-2}^+ node-disjoint copies of $Q(n-2)$ into $S(n)$, with load 1, dilation 3, and expansion $X(1, 3, p_{n-2}^+)$, where*

$$p_{n-2}^+ = (3 + A) \cdot \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor!, \quad X(1, 3, p_{n-2}^+) = \frac{n}{(3 + A) \cdot 2^{n-2}} \cdot \binom{n-1}{\lfloor n/2 \rfloor}, \quad \text{and} \quad A = \sum_{j=2}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j}$$

Proof: We initially pack p_{n-2} $Q(n-2)$'s into $S(n)$ via Algorithm 4. Algorithm 4 discards $1/5$ of $M(n-1)$ along dimension 4, $1/7$ of $M(n-1)$ along dimension 6, and so on. The slice of width 1 along dimension 4, for instance, is an $(n-2)$ -dimensional submesh of size $2 \times 3 \times 4 \times 6 \times 7 \times \dots \times n$. Not a single node of this submesh is used by the symmetric packing technique described in the proof of Theorem 2 (i.e., all mesh nodes with coordinate $m[4] = 4$ along the 4th dimension are available after the partitioning required by the symmetric packing of $Q(n-2)$ into $M(n-1)$ has been accomplished).

Let $M_4(n-1)$ be the induced submesh formed by all nodes $m[\] \in V(M(n-1))$, such that $m[4] = 4$. Since the size of $M_4(n-1)$ along any of its $(n-2)$ dimensions is at least 2, we can partition $M_4(n-1)$ into slices of width 2 along all its dimensions. Let the number of submeshes of width 2 along any dimension resulting from such partitioning be $p_{n-2}(4)$, where:

$$p_{n-2}(4) = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \left\lfloor \frac{6}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = \frac{p_{n-2}}{3 \cdot 2}$$

Clearly, partitioning $M_4(n-1)$ as shown above allows the packing of $p_{n-2}(4)$ additional copies of $Q(n-2)$ into $M(n-1)$, with load 1 and dilation 1. Note, however, that the $Q(n-2)$'s packed symmetrically according to Algorithm 4 do not have any of its links mapped onto the dimension 2 links of $M(n-1)$. The $p_{n-2}(4)$ extra copies, however, use dimension 2 links of $M(n-1)$. Hence, the resulting packing is asymmetric.

Note that the partitionings accomplished on both $M(n-1)$ and $M_4(n-1)$ do not use any mesh node with coordinate $m[6] = 6$ along the 6th dimension of $M(n-1)$. Therefore, we can apply the same technique just described for $M_4(n-1)$ to $M_6(n-1)$ (i.e., the induced submesh formed by all nodes $m[\] \in V(M(n-1))$, such that $m[6] = 6$). This partitioning results in $p_{n-2}(6)$ submeshes of width 2 along any dimension, where:

$$p_{n-2}(6) = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \left\lfloor \frac{5}{2} \right\rfloor \times \left\lfloor \frac{6}{2} \right\rfloor \times \left\lfloor \frac{8}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = \frac{p_{n-2}}{3 \cdot 3}$$

This technique can be extended to all dimensions of odd-size $s_i = i + 1$ in $M(n-1)$, for $4 \leq i \leq n$. If we call $i = 2j$, the number of additional packed $Q(n-2)$'s resulting from the partitioning of $M_{2j}(n-1)$ is $1/3j$. Since the largest possible value of j is $\lfloor (n-1)/2 \rfloor$, the total number of $Q(n-2)$'s that can be packed asymmetrically into $M(n-1)$ is:

$$p_{n-2}^+ = p_{n-2} + p_{n-2}(4) + p_{n-2}(6) + \dots + p_{n-2}(2 \lfloor (n-1)/2 \rfloor) = p_{n-2} \left(1 + \frac{1}{3 \cdot 2} + \frac{1}{3 \cdot 3} + \dots + \frac{1}{3 \cdot \lfloor \frac{n-1}{2} \rfloor} \right) =$$

$$p_{n-2} \left(1 + \frac{1}{3} \sum_{j=2}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j} \right) = (3 + A) \cdot \lfloor \frac{n}{2} \rfloor! \cdot \lfloor \frac{n-1}{2} \rfloor, \quad \text{where } A = \sum_{j=2}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j}$$

The corresponding packing of these p_{n-2}^+ copies of $Q(n-2)$ into $S(n)$ with load 1 and dilation 3 can be done by direct application of Algorithm 1. The derivation of the expansion ratio is also straightforward and follows from Equation 1. \square

For conciseness, we will not present here an algorithm that asymmetrically packs $Q(n-2)$ into $S(n)$. Such an algorithm, however, can be built as an extension of Algorithm 4 as follows. Initially, the number of the hypercube being packed (i.e., C) is tested to check which dimension assignment is to be used by the algorithm. If $0 \leq C < p_{n-2}$, the hypercube is packed according to Algorithm 4. If $p_{n-2} \leq C < p_{n-2}(1 + \frac{1}{6})$, then the hypercube node is mapped onto a node of $M_4(n-1)$. This can be done by making $m[1] = q[1]$, $m[2] = q[2]$, $m[3] = \text{offset}[3] + q[3]$, $m[4] = 4$, and $m[i] = \text{offset}[i] + q[i-1]$, for $5 \leq i < n$. In the general case, if $p_{n-2}(1 + \frac{1}{3} \sum_{k=2}^{j-1} \frac{1}{k}) \leq C < p_{n-2}(1 + \frac{1}{3} \sum_{k=2}^j \frac{1}{k})$, for some j such that $2 \leq j \leq \lfloor (n-1)/2 \rfloor$, then the hypercube node is mapped onto $M_{2j}(n-1)$. This can be done by making $m[1] = q[1]$, $m[2] = q[2]$, $m[i] = \text{offset}[i] + q[i]$ (if $3 \leq i \leq 2j$), $m[2j] = 2j$ and $m[i] = \text{offset}[i] + q[i-1]$ (if $2j < i < n$). As in Algorithm 4, $\text{offset}[i]$ has to be properly computed such that node-disjoint copies of $Q(n-2)$ are produced.

From the proof of Theorem 3, it becomes apparent why a similar approach was not used for obtaining an asymmetric packing of $Q(n-1)$ into $S(n)$. All unused induced submeshes $M_{2j}(n-1)$, $1 \leq j \leq \lfloor (n-1)/2 \rfloor$, resulting from the partitioning mechanism described in Theorem 1, are $(n-2)$ -dimensional. These induced submeshes can not be used to pack $Q(n-1)$ with load 1 and dilation 3 as a direct consequence of Lemma 2.

4.4 Packing $Q(n-t)$ into $S(n)$ ($1 \leq t \leq \lfloor (n+1)/2 \rfloor$)

It becomes apparent from the proofs of Theorems 1 and 2 that a fraction of the nodes is left unused whenever $M(n-1)$ is partitioned into slices of width 2 along any of its odd-sized dimensions. If $M(n-1)$ is partitioned along such dimensions into slices of width 1, smaller expansion ratios can be achieved, at the expense of reducing the size of the hypercubes that can still be packed in the resulting submeshes. Packing $Q(n-2)$ into $S(n)$ instead of $Q(n-1)$, for instance, reduces the expansion ratio by a factor of $1/3$ (Theorems 1 and 2).

We now consider the general case where $Q(n-t)$ is packed into $S(n)$, both symmetrically and asymmetrically. In particular, we are interested in the range $1 \leq t \leq \lfloor (n+1)/2 \rfloor$. The case $t = \lfloor (n+1)/2 \rfloor$ corresponds to the packing of $Q(n - \lfloor (n+1)/2 \rfloor) = Q(\lfloor n/2 \rfloor)$ into $S(n)$. Such a packing has unitary expansion ratio (i.e., it provides 100% utilization of the nodes in $S(n)$). Notably, $Q(\lfloor n/2 \rfloor)$ is the largest hypercube that can be packed with expansion 1 into $S(n)$ with our techniques, as explained later in the proof of Theorem 6.

Theorem 4 *It is possible to symmetrically pack p_{n-t} node-disjoint copies of $Q(n-t)$ into $S(n)$ ($1 \leq t \leq \lfloor (n+1)/2 \rfloor$), with load 1, dilation 3, and expansion $X(1, 3, p_{n-t})$, where*

$$p_{n-t} = \frac{\lfloor \frac{n}{2} \rfloor! \cdot \lfloor \frac{n-1}{2} \rfloor! \cdot t}{2^{t-1}} \cdot \binom{2t-1}{t-1} \quad \text{and} \quad X(1, 3, p_{n-t}) = \frac{n}{t \cdot 2^{n-2t+1}} \cdot \frac{\binom{n-1}{\lfloor n/2 \rfloor}}{\binom{2t-1}{t-1}}$$

Proof: The cases $t = 1$ and $t = 2$ have already been considered in the proofs of Theorems 1 and 2. As explained earlier, we partition $M(n-1)$ into slices of width 1 along odd-sized dimensions i in order to minimize the number of unused nodes in the mesh. For $t = 3$, we partition $M(n-1)$ in this fashion along dimensions $i = 2$ and $i = 4$. Since the sizes of $M(n-1)$ along these dimensions are, respectively, $s_2 = 3$ and $s_4 = 5$, partitioning $M(n-1)$ into slices of width 2 as described in the proof of Theorem 1 would result in a utilization of only $\frac{2}{3} \times \frac{4}{5}$ of the nodes. Clearly, partitioning $M(n-1)$ into unitary slices along dimensions $i = 2$ and $i = 4$ optimizes the utilization of the mesh when packing $Q(n-3)$'s. Accordingly, for $t = 4$, we partition $M(n-1)$ into slices of width 1 along dimensions $i = 2$, $i = 4$ and $i = 6$.

In the general case, to pack $Q(n-t)$'s we partition $M(n-1)$ into $(t-1)$ slices of width 1 along its first $(t-1)$ odd-sized dimensions. These dimensions are of size $s_2 = 3, s_4 = 5, \dots, s_{2t-2} = 2t-1$, respectively. Slices of width 2 are used along all even-sized dimensions of $M(n-1)$, as well as along any remaining odd-sized dimensions. Upon completion of the partitioning process, p_{n-t} $(n-t)$ -dimensional submeshes of size at least 2 along any dimension result, where:

$$\begin{aligned}
p_{n-t} &= \left\lfloor \frac{s_1}{2} \right\rfloor \times 3 \times \left\lfloor \frac{s_3}{2} \right\rfloor \times 5 \times \left\lfloor \frac{s_5}{2} \right\rfloor \times \dots \times \left\lfloor \frac{s_{2t-3}}{2} \right\rfloor \times (2t-1) \times \left\lfloor \frac{s_{2t-1}}{2} \right\rfloor \times \left\lfloor \frac{s_{2t}}{2} \right\rfloor \times \left\lfloor \frac{s_{2t+1}}{2} \right\rfloor \times \dots \times \left\lfloor \frac{s_{n-1}}{2} \right\rfloor = \\
&\left\lfloor \frac{2}{2} \right\rfloor \times 3 \times \left\lfloor \frac{4}{2} \right\rfloor \times 5 \times \left\lfloor \frac{6}{2} \right\rfloor \times \dots \times \left\lfloor \frac{2t-2}{2} \right\rfloor \times (2t-1) \times \left\lfloor \frac{2t}{2} \right\rfloor \times \left\lfloor \frac{2t+1}{2} \right\rfloor \times \left\lfloor \frac{2t+2}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n-1}{2} \right\rfloor \times \left\lfloor \frac{n}{2} \right\rfloor = \\
&\left(\left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \left\lfloor \frac{6}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor \right) \times (3 \times 5 \times \dots \times (2t-1)) \times \left(\left\lfloor \frac{2t}{2} \right\rfloor \times \left\lfloor \frac{2t+2}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n-1}{2} \right\rfloor \right) = \\
&\left(1 \times 2 \times 3 \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor \right) \times (3 \times 5 \times \dots \times (2t-1)) \times \left(t \times (t+1) \times \dots \times \left\lfloor \frac{n-1}{2} \right\rfloor \right) = \\
&\left\lfloor \frac{n}{2} \right\rfloor! \times \frac{(2t-1)!}{2^{t-1} \cdot (t-1)!} \times \frac{\left\lfloor \frac{n-1}{2} \right\rfloor!}{(t-1)!} = \frac{\left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \cdot t}{2^{t-1}} \cdot \binom{2t-1}{t-1}
\end{aligned}$$

Due to the partitioning process, the resulting $(n-t)$ -dimensional submeshes are node-disjoint. In addition, by applying Lemma 2 we note that it is possible to embed $Q(n-t)$ with load 1, dilation 1 into each of these p_{n-t} submeshes. Hence, it is possible to pack p_{n-t} node-disjoint copies of $Q(n-t)$ into $M(n-1)$, with load 1 and dilation 1. Furthermore, the packing can be done symmetrically by using the same dimension assignment for every copy of $Q(n-t)$. If we now embed $M(n-1)$ into $S(n)$ using the mapping given by Algorithm 1, a load 1, dilation 3 packing results.

To complete the proof, we note that the expansion of the packing can be obtained by direct application of Equation 1:

$$X(1, 3, p_{n-t}) = \frac{n!}{p_{n-t} \cdot 2^{n-t}} = \frac{n!}{\frac{\left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \cdot t}{2^{t-1}} \cdot \binom{2t-1}{t-1} \cdot 2^{n-t}} = \frac{n}{t \cdot 2^{n-2t+1}} \cdot \frac{\binom{n-1}{\left\lfloor \frac{n}{2} \right\rfloor}}{\binom{2t-1}{t-1}}$$

□

An algorithm that packs symmetrically $Q(n-t)$ into $S(n)$ follows:

Algorithm 5 (Symmetric packing of $Q(n-t)$ into $S(n)$):

```

spack_cube_n_t (int q[ ], int C, int n, int t, int p[ ])
{
  int i, int j, int R offset[ ], m[ ];
  j = 0;
  R = 1;
  m[1] = q[1];
  for (i = 2; i < n; i++) {
    if ((i mod 2 == 0) and (i ≤ (2t - 2))) {
      j++;
      m[i] = ⌊ $\frac{C}{R}$ ⌋ mod (i + 1);
      R = R · (i + 1);
    }
    else {
      offset[i] = 2 ⋅ (⌊ $\frac{C}{R}$ ⌋ mod ⌊ $\frac{i+1}{2}$ ⌋);
      R = R · ⌊ $\frac{i+1}{2}$ ⌋;
      m[i] = q[i - j] + offset[i];
    }
  }
  mesh_to_star (m[ ], n, p[ ])
}

```

In addition to the variables already used in Algorithms 3 and 4, Algorithm 5 uses two extra variables to map $q[] \in Q(n-t)$ onto $m[] \in M(n-1)$. One of these variables is referred to as R and simply keeps track of the amount of partitioning accomplished up to the i^{th} dimension of $M(n-1)$. In fact, in Algorithm 5 each iteration of the *for* loop corresponds to one step of the partitioning technique described in the proof of Theorem 4. For example, consider the packing of $Q(3)$ into $S(6)$ (i.e., $n = 6$, $t = 3$). As i is incremented from 2 to 5, R takes the values $R = 3$, $R = 3 \times 2$, $R = 3 \times 2 \times 5$, and $R = 3 \times 2 \times 5 \times 3$, respectively.

Another variable, j , counts the number of odd-sized dimensions along which $M(n-1)$ has been partitioned into slices of width 1. Note that $q[] \in Q(n-t)$ is a node label containing $(n-t)$ coordinates, and $m[] \in M(n-1)$ is a node label containing $(n-1)$ coordinates. To properly map $q[]$ onto $m[]$, we initially make $m[1] = q[1]$, and then check if the current partitioning step results in slices of width 1 or 2. Slices of width 1 are used along dimensions of odd size $s_i = i + 1$, if $i \leq 2t - 2$. In such dimensions, we simply assign to $m[i]$ a value corresponding to the offset $\lfloor C/R \rfloor \bmod (i + 1)$. In the remaining dimensions, we make $m[i] = q[i - j] + \text{offset}[i]$. The packing accomplished by Algorithm 5 can be shown to be symmetric by the same reasoning used in the description of Algorithm 3. Finally, $m[]$ is mapped onto $S(n)$ via a call to Algorithm 1.

Note that Algorithm 5 symmetrically packs $Q(n-t)$ into $S(n)$, for $1 \leq t \leq \lfloor (n+1)/2 \rfloor$, and therefore can be used instead of Algorithms 3 and 4 in the cases $t = 1$ and $t = 2$, respectively. In addition, any of the Algorithms 3, 4 and 5 have a time complexity of $O(n^2)$ and a space complexity of $O(n)$. $O(n^2)$ time complexity arises during the mapping of $m[]$ onto $S(n)$.

We now consider the case where $Q(n-t)$ is packed asymmetrically into $S(n)$, for $2 \leq t < \lfloor (n+1)/2 \rfloor$:

Theorem 5 *It is possible to asymmetrically pack p_{n-t}^+ node-disjoint copies of $Q(n-t)$ into $S(n)$ ($2 \leq t < \lfloor (n+1)/2 \rfloor$), with load 1, dilation 3, and expansion $X(1, 3, p_{n-t}^+)$, where*

$$p_{n-t}^+ = (1+B) \cdot \frac{\lfloor \frac{n}{2} \rfloor! \cdot \lfloor \frac{n-1}{2} \rfloor! \cdot t}{2^{t-1}} \cdot \binom{2t-1}{t-1}, \quad X(1, 3, p_{n-t}^+) = \frac{n}{(1+B) \cdot t \cdot 2^{n-2t+1}} \cdot \frac{\binom{n-1}{\lfloor n/2 \rfloor}}{\binom{2t-1}{t-1}},$$

$$\text{and } B = \frac{t-1}{2t-1} \sum_{j=t}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j}$$

Proof: The partitioning technique described in the proof of Theorem 4 discards $1/s_i$ of $M(n-1)$ along each of its dimensions of odd size $s_i = i+1$, where $i = 2j$, for $t \leq j \leq \lfloor (n-1)/2 \rfloor$. An unused slice of width 1 along dimension $2j$ is an $(n-2)$ -dimensional submesh of size $2 \times 3 \times 4 \times \dots \times (2j-1) \times 2j \times (2j+2) \times \dots \times (n-1) \times n$. Let this submesh be referred to as $M_{2j}(n-1)$ (i.e., $M_{2j}(n-1)$ is the induced submesh formed by all nodes $m[\] \in V(M(n-1))$, such that $m[2j] = 2j$). Clearly, not a single node of $M_{2j}(n-1)$ is used by the symmetric packing technique described in the proof of Theorem 4.

We initially pack p_{n-t} $Q(n-t)$'s into $S(n)$ via Algorithm 5. To pack additional $Q(n-t)$'s, we partition $M_{2j}(n-1)$ into $(t-2)$ slices of width 1 along its first $(t-2)$ odd-sized dimensions. These dimensions are of size $s_2 = 3, s_4 = 5, \dots, s_{2t-4} = 2t-3$, respectively. Slices of width 2 are used along all even-sized dimensions of $M(n-1)$, as well as along any remaining odd-sized dimensions. Upon completion of the partitioning process, $p_{n-t}(2j)$ $(n-t)$ -dimensional submeshes of size at least 2 along any dimension result, where:

$$p_{n-t}(2j) = \left\lfloor \frac{2}{2} \right\rfloor \times 3 \times \left\lfloor \frac{4}{2} \right\rfloor \times 5 \times \left\lfloor \frac{6}{2} \right\rfloor \times \dots \times \left\lfloor \frac{2t-4}{2} \right\rfloor \times (2t-3) \times \left\lfloor \frac{2t-2}{2} \right\rfloor \times \left\lfloor \frac{2t-1}{2} \right\rfloor \times \left\lfloor \frac{2t}{2} \right\rfloor \times \dots$$

$$\dots \times \left\lfloor \frac{2j-1}{2} \right\rfloor \times \left\lfloor \frac{2j}{2} \right\rfloor \times \left\lfloor \frac{2j+2}{2} \right\rfloor \times \left\lfloor \frac{2j+3}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = \frac{\lfloor \frac{2t-1}{2} \rfloor!}{\lfloor \frac{2j+1}{2} \rfloor! \cdot (2t-1)} \cdot p_{n-t} = \frac{t-1}{j \cdot (2t-1)} \cdot p_{n-t}$$

Clearly, partitioning $M_{2j}(n-1)$ as shown above allows the packing of $p_{n-t}(2j)$ additional copies of $Q(n-t)$ into $M(n-1)$, with load 1 and dilation 1. Note, however, that the $Q(n-t)$'s packed symmetrically according to Algorithm 4 do not have any of its links mapped onto the dimension $(2t-2)$ links of $M(n-1)$. The $p_{n-t}(2j)$ extra copies, however, use dimension $(2t-2)$ links of $M(n-1)$. Hence, the resulting packing is asymmetric.

Note that the partitionings accomplished on both $M(n-1)$ and $M_{2j}(n-1)$ do not use any mesh node with coordinate $m[2k] = 2k$ along dimension $2k$ of $M(n-1)$, for any j, k such that $t \leq j, k \leq \lfloor (n-1)/2 \rfloor$ and $j \neq k$. Therefore, we can apply the technique just described for all induced submeshes $M_{2j}(n-1)$, which provides a total of $\sum_{j=t}^{\lfloor (n-1)/2 \rfloor} p_{n-t}(2j)$ additional packed copies of $Q(n-t)$. Hence, the total number of $Q(n-t)$'s that can be packed asymmetrically into $M(n-1)$ is:

$$p_{n-t}^+ = p_{n-t} + \sum_{j=t}^{\lfloor \frac{n-1}{2} \rfloor} p_{n-t}(2j) = p_{n-t} \left(1 + \frac{t-1}{2t-1} \sum_{j=t}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j} \right) = (1+B) \cdot \frac{\lfloor \frac{n}{2} \rfloor! \cdot \lfloor \frac{n-1}{2} \rfloor! \cdot t}{2^{t-1}} \cdot \binom{2t-1}{t-1},$$

$$\text{where } B = \frac{t-1}{2t-1} \sum_{j=t}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j}$$

The corresponding packing of these p_{n-t}^+ copies of $Q(n-t)$ into $S(n)$ with load 1 and dilation 3 can be done by direct application of Algorithm 1. The derivation of the expansion ratio is also straightforward and follows from Equation 1. \square

For conciseness, we will not present here an algorithm that asymmetrically packs $Q(n-t)$ into $S(n)$. Such an algorithm, however, can be built as an extension of Algorithm 5, by using the same principles that were discussed for the case $t = 2$.

4.5 Packing $Q(\lfloor n/2 \rfloor)$ into $S(n)$ with expansion 1

We now consider a load 1, dilation 3, expansion 1 symmetric packing of $Q(\lfloor n/2 \rfloor)$ into $S(n)$. As shown below, $Q(\lfloor n/2 \rfloor)$ is the largest hypercube that can be packed into $S(n)$ with expansion 1 via our technique.

Theorem 6 *It is possible to symmetrically pack $p_{\lfloor n/2 \rfloor}$ node-disjoint copies of $Q(\lfloor n/2 \rfloor)$ into $S(n)$, with load 1, dilation 3, and expansion $X(1, 3, p_{\lfloor n/2 \rfloor})$, where*

$$p_{\lfloor n/2 \rfloor} = \frac{n!}{2^{\lfloor n/2 \rfloor}} \quad \text{and} \quad X(1, 3, p_{\lfloor n/2 \rfloor}) = 1$$

Proof: We restrict the partitioning of $M(n-1)$ into slices of width 2 to occur only along dimensions in which s_i (i.e., the size of $M(n-1)$ along dimension i) is even. Clearly, there are $\lfloor n/2 \rfloor$ such dimensions. Accordingly, $M(n-1)$ is partitioned into slices of width 1 along all its odd-sized dimensions. The submeshes resulting from this partitioning strategy are $\lfloor n/2 \rfloor$ -dimensional, of size 2 along any dimension, and node-disjoint. In fact, each of these submeshes is an $\lfloor n/2 \rfloor$ -dimensional hypercube, whose dimension assignment can be fixed such that a symmetric packing results. The expansion ratio of such a packing is 1, and the number of copies of $Q(\lfloor n/2 \rfloor)$ that can be packed into $S(n)$ follows immediately from Equation 1, noting that $|S(n)| = n!$ and $|Q(\lfloor n/2 \rfloor)| = 2^{\lfloor n/2 \rfloor}$. \square

An algorithm that symmetrically packs $Q(\lfloor n/2 \rfloor)$ into $S(n)$ with expansion 1 can be obtained from Algorithm 5 by making $t = \lfloor (n+1)/2 \rfloor$.

4.6 Results on packing $Q(n-t)$ into $S(n)$

Table 3 depicts the number of packed hypercubes and the expansion ratios resulting from the techniques presented in this section. The values that are listed between brackets refer to asymmetric packings. Accordingly, the remaining values refer to symmetric packings. Note that low expansion ratios are obtained, meaning that a large portion of the nodes of $S(n)$ are used in the packings. Figure 6 plots the expansion ratios of the packings shown in Table 3 and provides further insight on how the different approaches that were considered in this section can be used to achieve dense packings of $Q(n-t)$ into $S(n)$. For $n = 9$ and $n = 10$, for example, the expansion ratio drops from 2.46 to 1.13 as we reduce the size of the hypercubes being packed, respectively by varying t from 1 to 4. Asymmetry also proves to be an efficient technique to achieve denser packings, resulting in an expansion ratio of at most 1.20 among all cases shown in Table 3, which corresponds to a very efficient utilization of the star graph. For $n = 9$ and $n = 10$, for example, the expansion ratio drops from 1.64 to 1.20 if an asymmetric packing of $Q(n-2)$ is used instead of a symmetric packing technique.

It is also interesting to note that for $n > 6$, an asymmetric packing of $Q(n-2)$ achieves lower expansion ratio than does a symmetric packing of $Q(n-3)$. Similarly, for $n > 8$, an asymmetric packing of $Q(n-3)$ achieves lower expansion ratio than does a symmetric packing of $Q(n-4)$. Note that in all cases a slight

increase in the expansion ratios occurs whenever n is incremented to an odd number, which can be explained by the partitioning strategies presented throughout this section.

$S(n)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$	$S(8)$	$S(9)$	$S(10)$
Number of packed $Q(n-1)$'s	1	2	4	12	36	144	576	2,880
Expansion ratio	1.50	1.50	1.88	1.88	2.19	2.19	2.46	2.46
Number of packed $Q(n-2)$'s	3	6	12	36	108	432	1,728	8,640
Expansion ratio	1.00	1.00	[1.25]	[1.25]	[1.46]	[1.46]	[1.64]	[1.64]
			[1.07]	[1.07]	[1.14]	[1.14]	[1.20]	[1.20]
Number of packed $Q(n-3)$'s	6	12	30	90	270	1,080	4,320	21,600
Expansion ratio	1.00	1.00	1.00	1.00	1.17	1.17	1.31	1.31
					[1.03]	[1.03]	[1.06]	[1.06]
Number of packed $Q(n-4)$'s	–	24	60	180	630	2,520	10,080	50,400
Expansion ratio	–	1.00	1.00	1.00	1.00	1.00	1.13	1.13
							[1.02]	[1.02]

Table 3: Results on packing $Q(n-t)$ into $S(n)$

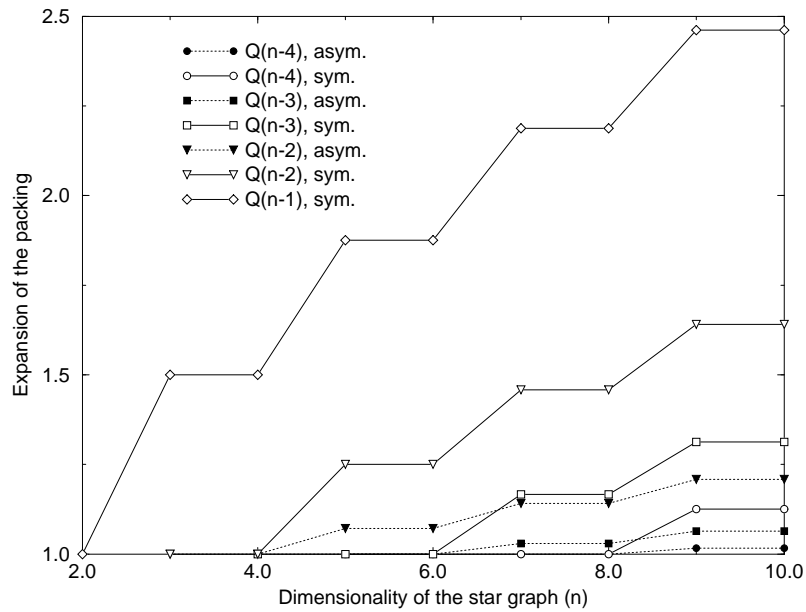


Figure 6: Expansion ratios of packings of $Q(n-t)$ into $S(n)$

5 Variable-Dilation Embeddings of $Q(n - 1 + \ell)$ into $S(n)$

5.1 Basic techniques

The largest hypercube considered by the packing techniques described in the previous section is $Q(n - 1)$. In this section, we describe how an $(n - 1 + \ell)$ -dimensional hypercube can be embedded with variable dilation into $S(n)$. Such an embedding requires 2^ℓ copies of $Q(n - 1)$, packed symmetrically into $S(n)$, where ℓ is limited by

$$\ell \leq \lfloor \log_2 p_{n-1} \rfloor = \left\lfloor \log_2 \left(\left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \right) \right\rfloor \quad (7)$$

Figure 7 depicts an example of the technique we will be describing in this section. A 4-dimensional hypercube is embedded into $M(3)$ as the result of the connection of two packed $Q(3)$'s. Note that the dilation of such an embedding along the i^{th} dimension of $Q(4)$ is

$$d_i = \begin{cases} 1 & , \text{ if } i \leq 3 \\ 2 & , \text{ if } i = 4 \end{cases}$$

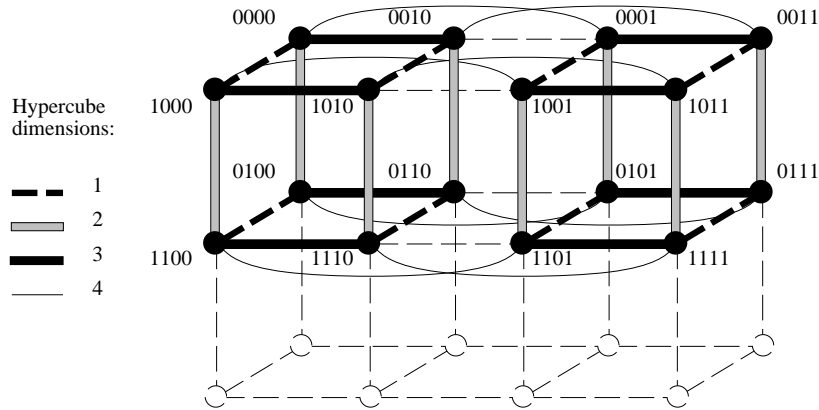


Figure 7: A variable-dilation embedding of $Q(4)$ into $M(3)$

As defined in Section 2, Figure 7 corresponds to a variable dilation embedding whose dilation vector is $\overline{d}_4 = [1, 1, 1, 2]$. If we now map $M(3)$ into $S(4)$ using Algorithm 1, the dilation vector of the corresponding embedding of $Q(4)$ into $S(4)$ becomes $\overline{d}_4 = [3, 3, 3, 4]$. The dilation along the first 3 dimensions of $Q(4)$ increases from 1 to 3 as a direct consequence of Lemma 1. The fact that the dilation along the 4th dimension of $Q(4)$ increases from 2 to 4 is justified by the following Lemma:

Lemma 3 *Let $\omega, \omega_{i,2} \in V(M(n - 1))$ be a pair of nodes separated by 2 links along the i^{th} dimension of $M(n - 1)$, $2 \leq i \leq n - 1$. In the corresponding embedding of $M(n - 1)$ into $S(n)$, ω and $\omega_{i,2}$ are connected by a path containing either 2 or 4 links.*

Proof: Without loss of generality, we assume that the i^{th} coordinates of ω and $\omega_{i,2}$ (respectively, $m[i]$ and $m''[i]$) are such that $m[i] = m''[i] - 2$. By inspection of Algorithm 1 and Table 1, we note that the mapping of $\omega \in V(M(n - 1))$ onto $\pi \in V(S(n))$ uses a sequence of transpositions of the form:

$$\sigma = (a b)(c d) \dots (i j)(o p) \dots (y z)$$

Accordingly, $\omega_{i,2}$ is mapped onto $\pi_{i,2}$ via a sequence of transpositions of the form:

$$\sigma_{i,2} = (a b)(c d) \dots (i j)(k l)(m n)(o p) \dots (y z)$$

Note that identical transpositions are used in σ and $\sigma_{i,2}$, except for the fact that $\omega_{i,2}$ has two transpositions more than does ω_i (namely, $(k l)$ and $(m n)$). An inspection of Table 1 reveals that these extra transpositions are $((i - m''[i] + 3) (i - m''[i] + 2))$ and $((i - m''[i] + 2) (i - m''[i] + 1))$, respectively. Alternatively, we can use the cyclic representation $(k l)(m n) = (k l)(l n)$, since $l = m = n + 1 = k - 1$. According to Equation 6, these two transpositions can be accomplished via a sequence of star operations as follows:

$$(k l)(l n) \equiv \begin{cases} l \rightarrow k \rightarrow l \rightarrow l \equiv l \rightarrow k & , \text{ if } n = 1 \\ l \rightarrow k \rightarrow l \rightarrow l \rightarrow n \rightarrow l \equiv l \rightarrow k \rightarrow n \rightarrow l & , \text{ if } n \neq 1 \end{cases} \quad (8)$$

Note that the execution of the two transpositions requires either 2 or 4 star operations. We can actually concatenate $(k l)(l n)$ into a single permutation cycle [20] $(n l k)$, such that:

$$(n l k) \equiv \begin{cases} l \rightarrow k & , \text{ if } n = 1 \\ n \rightarrow l \rightarrow k \rightarrow n \equiv l \rightarrow k \rightarrow n \rightarrow l \equiv k \rightarrow n \rightarrow l \rightarrow k & , \text{ if } n \neq 1 \end{cases} \quad (9)$$

Let the digits occupying the n^{th} , l^{th} and k^{th} positions of the permutation resulting from applying transpositions $(a b)(c d) \dots (i j) \in \sigma, \sigma_{i,2}$ to the identity $123 \dots n$ be respectively d_1, d_2 and d_3 . Therefore, $(n l k)$ moves d_1 into d_2 's position, d_2 into d_3 's position, and d_3 into d_1 's position. Since the remaining transpositions in $\sigma, \sigma_{i,2}$ (i.e., $(o p) \dots (y z)$) are identical, π and $\pi_{i,2}$ will differ only in the final positions occupied by digits d_1, d_2 and d_3 . Let these positions be α, β and γ , such that if in π we have $p[\alpha] = d_1, p[\beta] = d_2$, and $p[\gamma] = d_3$, in $\pi_{i,2}$ we have $p[\alpha] = d_3, p[\beta] = d_1$ and $p[\gamma] = d_2$. Therefore, routing from π to $\pi_{i,2}$ requires a single cycle $(\alpha \beta \gamma)$, which corresponds to a path containing either 2 or 4 links in $S(n)$ as indicated by Equation 9. \square

Theorem 7 For $n \geq 4$, there is a load 1, variable-dilation embedding of $Q(n-1+\ell)$ into $S(n)$, $0 < \ell \leq n-3$, whose dilation vector and expansion ratio are respectively

$$d_i = \begin{cases} 3 & , \text{ if } i \leq n-1 \\ 4 & , \text{ if } n \leq i \leq n-1+\ell \end{cases} \quad \text{and} \quad X(1, \overline{d_{n-1+\ell}}) = \frac{n!}{2^{n-1+\ell}}$$

Proof: Due to the partitioning process described in the proof of Theorem 1, at least 2 slices containing symmetrically packed $Q(n-1)$'s can be found if $M(n-1)$ is traversed along dimension $i \geq 3$. If we connect the packed hypercubes along ℓ of these dimensions as shown in Figure 7, a variable-dilation embedding of $Q(n-1+\ell)$ into $M(n-1)$ results. The dilation vector of such an embedding is:

$$d_i = \begin{cases} 1 & , \text{ if } i \leq n-1 \\ 2 & , \text{ if } n \leq i \leq n-1+\ell \end{cases} \quad (10)$$

Note that an extra hypercube dimension can be created with dilation 2 along each dimension $i \geq 3$ in $M(n-1)$ via the technique depicted in Figure 7. Since in $M(n-1)$ there are $(n-3)$ such dimensions, the embedding must observe the constraint $0 \leq \ell \leq n-3$.

If we now map $M(n-1)$ onto $S(n)$ via Algorithm 1, the dilation vector claimed in the theorem results as a direct consequence of Lemmas 1 and 3. Finally, the expansion of the embedding can be obtained by a simple application of Equation 4. \square

We now present an algorithm that embeds $Q(n-1+\ell)$ with variable dilation into $S(n)$, according to the technique described in the proof of Theorem 7. The algorithm requires that $n \geq 4$.

Algorithm 6 (Variable-dilation embedding of $Q(n-1+\ell)$ into $S(n)$, $n \geq 4$):

```

embed_cube_4 (int q[ ], int n, int ℓ, int p[ ])
{
  int i, m[ ];
  for (i = 1; i < n; i++) m[i] = q[i];
  for (i = n; i < (n + ℓ); i++) m[i - n + 3] = m[i - n + 3] + 2q[i];
  mesh_to_star (m[ ], n, p[ ])
}

```

Initially, the algorithm copies the first $(n-1)$ coordinates of the hypercube node $(q[])$ onto the coordinates of a mesh node $m[]$. The remaining ℓ coordinates of $q[]$ are used by the algorithm in the computation of an appropriate offset vector (i.e., $q[]$ is multiplied by 2 and added sequentially to coordinates $m[3]$ to $m[\ell+2]$). The resulting mesh coordinate is then mapped onto $S(n)$ via a call to Algorithm 1.

We now state without proof the following lemma, which will be helpful to the next variable-dilation technique considered in this section (proof is actually analogous to those of Lemmas 1 and 3):

Lemma 4 *Let $\omega, \omega_{i,4} \in V(M(n-1))$ be a pair of nodes separated by 4 links along the i^{th} dimension of $M(n-1)$, $4 \leq i \leq n-1$. In the corresponding embedding of $M(n-1)$ into $S(n)$, ω and $\omega_{i,4}$ are connected by a path containing 6 links.*

Theorem 8 *For $n \geq 8$, there is a load 1, variable-dilation embedding of $Q(n-1+\ell)$ into $S(n)$, $0 < \ell \leq 2n-10$, whose dilation vector and expansion ratio are respectively*

$$d_i = \begin{cases} 3 & , \text{ if } i \leq n-1 \\ 4 & , \text{ if } n \leq i \leq 2n-4 \\ 6 & , \text{ if } 2n-3 \leq i \leq n-1+\ell \end{cases} \quad \text{and} \quad X(1, \overline{d_{n-1+\ell}}) = \frac{n!}{2^{n-1+\ell}}$$

Proof: As described in the proof of Theorem 7, for $n \geq 4$ up to 2^{n-3} symmetrically packed $Q(n-1)$'s can be connected in $M(n-1)$ along dimensions i , such that $i \geq 4$. This results in an $(2n-4)$ -dimensional hypercube whose dilation vector is given by Equation 10.

For $n \geq 8$, there are $(n-7)$ dimensions along which $M(n-1)$'s size is $s_i \geq 8$. In addition, along these dimensions at most one half of the packed $Q(n-1)$'s is used by the technique depicted in Figure 7 when forming $Q(2n-4)$. It is therefore possible to form additional $Q(2n-4)$'s with the unused $Q(n-1)$'s. Once these $Q(2n-4)$ hypercubes are properly connected, larger hypercubes result. In fact, a new hypercube dimension can be created with dilation 4 by connecting $Q(2n-4)$'s along each dimension $i \geq 7$ in $M(n-1)$, as shown in Figure 8. Hence, at most $(n-7)$ extra dimensions can be obtained with this technique, which results in the following dilation vector for the embedding of $Q(n-1+\ell)$ into $M(n-1)$:

$$d_i = \begin{cases} 1 & , \text{ if } i \leq n-1 \\ 2 & , \text{ if } n \leq i \leq 2n-4 \\ 4 & , \text{ if } 2n-3 \leq i \leq n-1+\ell \end{cases}$$

Note that this embedding follows the constraint $0 \leq \ell \leq (n-3) + (n-7) = 2n-10$.

If we now map $M(n-1)$ onto $S(n)$ via Algorithm 1, the dilation vector claimed in the theorem results as a direct consequence of Lemmas 1, 3 and 4. Finally, the expansion of the embedding can be obtained by a simple application of Equation 4. \square

We now present an algorithm that embeds $Q(n-1+\ell)$ with variable dilation into $S(n)$, according to the technique described in the proof of Theorem 8. The algorithm requires that $n \geq 8$.

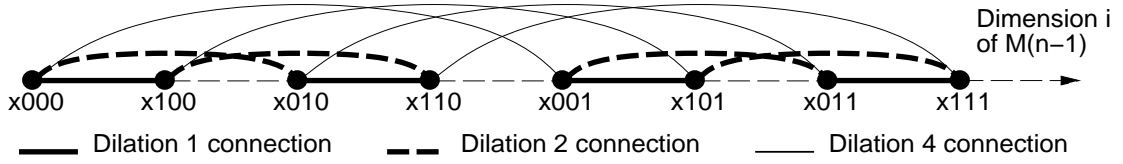


Figure 8: Embedding 3 dimensions of $Q(n-1+\ell)$ along the i^{th} dimension of $M(n-1)$, $i \geq 7$

Algorithm 7 (Variable-dilation embedding of $Q(n-1+\ell)$ into $S(n)$, $n \geq 8$):

```

embed_cube_8 (int q[ ], int n, int ell, int p[ ])
{
  int i, m[ ];
  for (i = 1; i < n; i++) m[i] = q[i];
  for (i = n; i <= (2n - 4); i++) m[i - n + 3] = m[i - n + 3] + 2q[i];
  for (i = 2n - 3; i < (n + ell); i++) m[i - 2n + 10] = m[i - 2n + 10] + 4q[i];
  mesh_to_star (m[ ], n, p[ ])
}

```

Initially, the algorithm copies the first $(n-1)$ coordinates of the hypercube node ($q[\]$) onto the coordinates of a mesh node $m[\]$. The remaining ℓ coordinates of $q[\]$ are used in the computation of proper offsets by the algorithm. Coordinates $q[n]$ to $q[2n-4]$ are multiplied by 2 and added sequentially to coordinates $m[3]$ to $m[n-1]$, while coordinates $q[2n-3]$ to $q[n-1+\ell]$ are multiplied by 4 and added sequentially to coordinates $m[7]$ to $m[\ell-n-9]$. The resulting mesh coordinate is finally mapped onto $S(n)$ via a call to Algorithm 1.

Extensions of Theorems 7 and 8 for the cases $n \geq 16$, $n \geq 32$, ... are left for the reader and will not be presented here. Accordingly, algorithms for these cases follow as extensions of Algorithms 6 and 7.

5.2 Advanced techniques

Let $Q(k_a)$ and $Q(k_b)$ be the largest hypercubes that can be embedded with variable dilation into $S(n)$, via Algorithms 6 and 7 respectively. Algorithm 6 requires that $n \geq 4$ and results in $k_a = n-1+n-3 = 2n-4$ (see Theorem 7). Accordingly, Algorithm 7 requires that $n \geq 8$ and results in $k_b = n-1+2n-10 = 3n-11$ (see Theorem 8). On the other hand, Equation 7 sets an upper limit on the number of extra dimensions that can be obtained when symmetrically packed $Q(n-1)$'s are used to build a variable-dilation embedding of $Q(k)$ into $S(n)$.

Let $Q(k_c)$ be the largest hypercube that can be embedded with variable dilation into $S(n)$, based solely on the limitation imposed by the number of copies of $Q(n-1)$ produced by the corresponding packing technique presented in Section 4 (i.e., p_{n-1}). Hence, $k_c = n-1 + \lceil \log_2 p_{n-1} \rceil$. Table 4 shows how well Algorithms 6 and 7 do in comparison to the upper limit k_c . We stress that k_c is computed from the number of copies of $Q(n-1)$ obtained with our packing techniques, and may not correspond to the largest hypercube that can actually be embedded with load 1 into $S(n)$ (we denote such a hypercube $Q(k_{max})$). A random one-to-one mapping of the nodes of $Q(k_{max})$ into the nodes of $S(n)$, for example, can be used to embed a hypercube containing at most $k_{max} = \lceil \log_2(n!) \rceil$ dimensions, where k_{max} is computed in accordance to the number of nodes existing in $S(n)$ (i.e., $|V(S(n))| = n!$). Such an approach, however, may result in a dilation of $d(S(n)) = \lceil 3(n-1)/2 \rceil$, where $d(S(n))$ is the diameter of $S(n)$. For completeness, we also list values of k_{max} in Table 4.

$S(n)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$	$S(8)$	$S(9)$	$S(10)$
k_a (Algorithm 6)	4	6	8	10	12	14	16
k_b (Algorithm 7)	–	–	–	–	13	16	19
k_c (Upper limit from p_{n-1})	4	6	8	11	14	17	20
k_{max} (Upper limit from $ V(S(n)) $)	4	6	9	12	15	18	21

Table 4: Number of dimensions achieved by Algorithms 6 and 7 vs. upper limits k_c and k_{max}

As Table 4 indicates, Algorithm 6 matches the upper limit k_c for $4 \leq n \leq 6$, and provides a maximum number of dimensions (k_a) that is one less than the upper limit k_c for $n = 7$. Assuming that Algorithm 7 is used for $8 \leq n \leq 10$, we achieve a number of dimensions k_b that is one less than the upper limit k_c .

Constructing a variable-dilation embedding of $Q(k_c)$ into $S(n)$ requires a special connection arrangement between the packed $Q(n-1)$'s. Although most dimensions of $Q(k_c)$ can be constructed using the same techniques that were described earlier in this section, we need to identify where do Algorithms 6 and 7 fail to achieve an efficient utilization of the packed hypercubes. We note that for each dimension i of $M(n-1)$, these algorithms use only $2^{\lceil \log_2(i+1) \rceil - 1}$ slices of width 2 resulting from the partitioning strategy required to pack $Q(n-1)$'s into $S(n)$ (see Theorem 1). For $n \geq 7$, for example, there are 3 such slices along dimensions 5 and 6 of $M(n-1)$, but both Algorithm 6 and Algorithm 7 use only 2 slices along these dimensions. Furthermore, only 2 extra hypercube dimensions are obtained via the basic variable-dilation embedding techniques accomplished by Algorithms 6 and 7 along dimensions 5 and 6 of $M(n-1)$. By using a special connection arrangement of the packed $Q(n-1)$'s along these dimensions, we can actually obtain one extra hypercube dimension and reach the upper limit k_c .

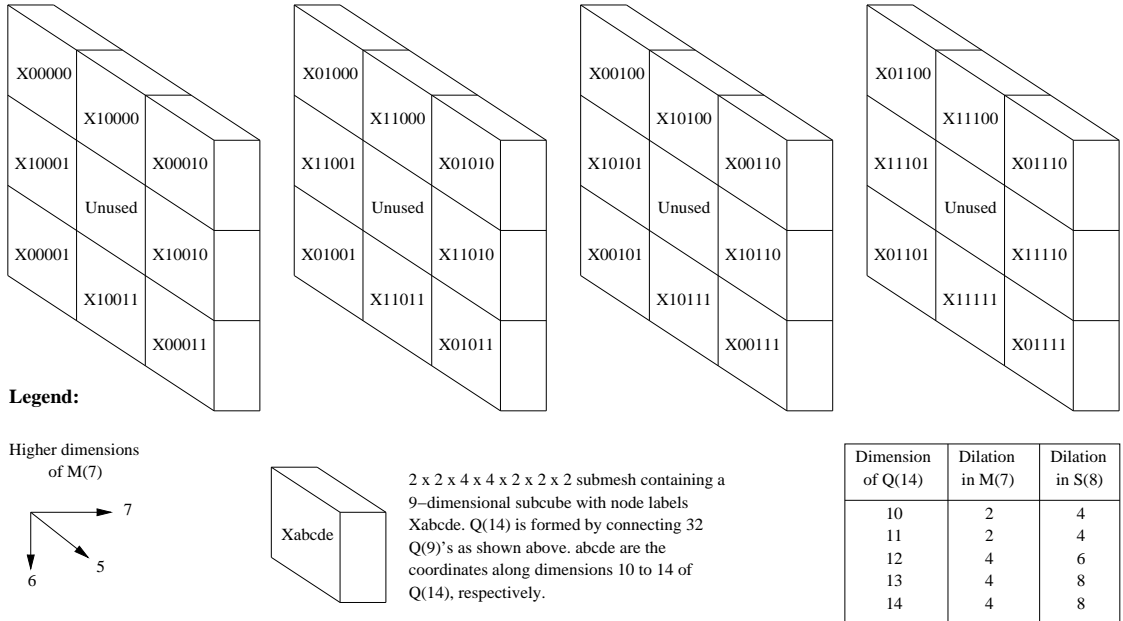


Figure 9: Subcube arrangement required to embed $Q(14)$ into $M(7)$

One example of the technique employed to embed $Q(k_c)$ into $S(n)$, $n \geq 7$, is given in Figure 9. We show

how to embed $Q(14)$ into $S(8)$, by properly using up to 3 slices of width 2 along dimensions 5 and 6 of $M(7)$. For clarity, only dimension 7 of $M(7)$ is shown in Figure 9 in addition to dimensions 5 and 6. The remaining dimensions of $M(7)$ are shown in a compacted format, following the assumption that each submesh resulting from the partitioning made along dimensions 5 through 7 holds a 9-dimensional hypercube. A copy of $Q(9)$ can be easily embedded with variable dilation in each of these submeshes by using the same techniques employed by Algorithm 6.

The most important observation that has to be made in regard to Figure 9 is the fact that some connections along dimensions 13 and 14 of $Q(14)$ require the traversal of a path containing 2 different dimensions of $M(7)$ (i.e., dimensions 5 and 6). The dilation along dimensions 13 and 14 of $Q(14)$, from the viewpoint of the variable-dilation embedding in $M(7)$, is $2 + 2 = 4$. Note, however, that Lemma 4 does not hold in this case. In other words, the dilation along dimensions 13 and 14 of $Q(14)$, once $M(7)$ is mapped onto $S(8)$, is 8 instead of 6 since two consecutive applications of Lemma 3 have to be considered.

The same technique shown in Figure 9 can be used to reach the upper limit k_c for the cases $n = 7$, $n = 9$ and $n = 10$. For conciseness, we will not present these embeddings in detail here, as well as an algorithm to embed $Q(k_c)$ into $S(n)$ with variable dilation. However, we list in Table 5 the dilation vectors of such embeddings for $7 \leq n \leq 10$.

<i>Embedding</i>	<i>Dilation vector $(\overline{d_{k_c}})$</i>
$Q(11)$ into $S(7)$	[3, 3, 3, 3, 3, 3, 4, 4, 4, 8, 8]
$Q(14)$ into $S(8)$	[3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 6, 8, 8]
$Q(17)$ into $S(9)$	[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 6, 6, 8, 8]
$Q(20)$ into $S(10)$	[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 8, 8]

Table 5: Variable-dilation embeddings of $Q(k_c)$ into $S(n)$, $k_c = n - 1 + \lceil \log_2 p_{n-1} \rceil$, $7 \leq n \leq 10$

As a last observation in regard to the results of this subsection, we note that the number of dimensions achieved with our advanced variable-dilation embeddings is still one less than the limit k_{max} for $6 \leq n \leq 10$. To obtain this extra dimension via a variable-dilation embedding, we need more packed $Q(n-1)$'s than those produced by the packing techniques of Section 4. In Section 6, we present expansion 1 multiple-sized packings that can be transformed into packings containing a sufficiently large number of copies of $Q(n-1)$. The additional $Q(n-1)$'s are obtained via a combination of asymmetric packing and variable-dilation embedding techniques. The details of how these $Q(n-1)$'s can be connected to form $Q(k_{max})$, however, are beyond the scope of this paper.

5.3 Communication slowdown resulting from variable-dilation embeddings

As explained in Section 2, variable-dilation embeddings can achieve a considerably smaller communication slowdown for certain classes of algorithms than d_{max} , the maximum dilation existing along any dimension of the embedded guest graph. Figure 10 gives estimates for the communication slowdown $\eta(Q(k) \mapsto S(n))$, assuming an algorithm where each dimension of $Q(k)$ is used during an equal number of steps. In other words, the communication slowdown is estimated by the average dilation of the embedding, d_{avr} (Equation 3).

In the case of $S(10)$, for example, Figure 10 shows that $Q(k)$ can be embedded with fixed dilation 3 for $k \leq 9$, which is a consequence of Lemma 2. For $10 \leq k \leq 16$, a variable-dilation embedding resulting from the

connection of packed $Q(9)$'s causes the average dilation to increase slowly up to 3.44. The dilation vectors used in the case $10 \leq k \leq 16$ are as defined in Theorem 7. For $17 \leq k \leq 19$, a variable-dilation embedding resulting from the connection of $Q(16)$'s causes the average dilation to increase at a slightly faster rate up to 3.84. The dilation vectors used in the case $17 \leq k \leq 19$ are as defined in Theorem 8. For $k = 20$, the dilation vector of the embedding is as given in Table 5. Although this last case presents a dilation of 8 along the two highest dimensions of $Q(20)$, the average dilation is nearly one half of that value (4.25).

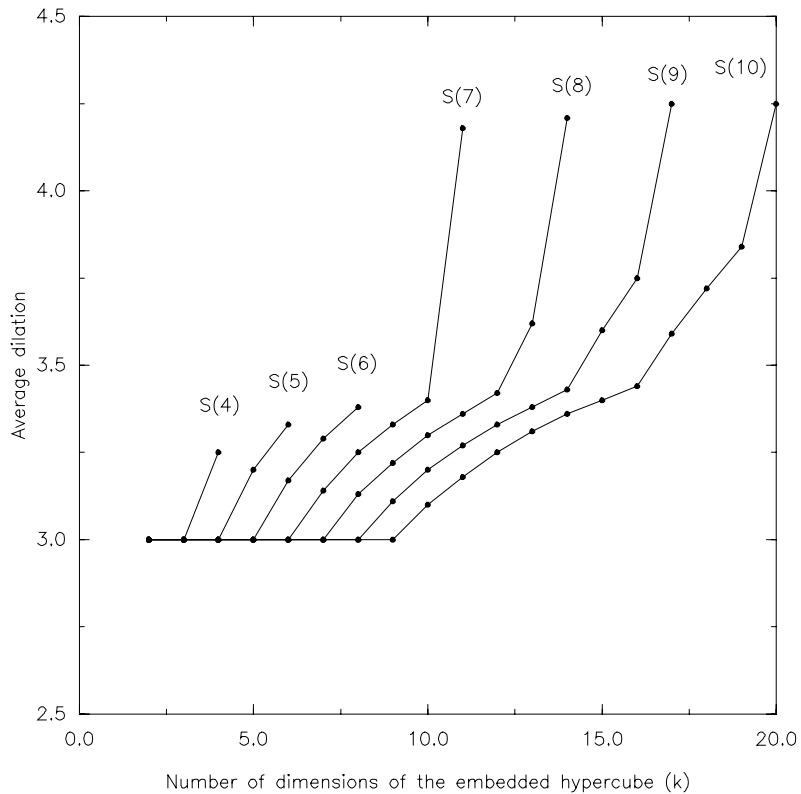


Figure 10: Average dilation of embeddings of $Q(k)$ into $S(n)$

Note that the communication slowdown for any particular algorithm depends on different factors, such as the frequency of utilization of each dimension of the hypercube and code dependencies. For an algorithm with uneven utilization of hypercube dimensions, improved performance can be obtained by simply reassigning these dimensions prior to the embedding into $S(n)$. In other words, due to the symmetry existing in $Q(k)$, it is possible to relabel the hypercube nodes such that in the final embedding into $S(n)$ the most frequently used hypercube dimensions have the smallest dilation. Under these assumptions, values even smaller than those shown in Figure 10 can be expected if Equation 2 is used to estimate the communication slowdown resulting from the embedding. The effect of code dependencies is difficult to evaluate analytically, but simulations can be used to point out the best hypercube dimension assignment if the highest possible performance is required for a given application.

6 Multiple-Sized Packings

We now consider a technique by which hypercube tasks with different requirements in terms of numbers of nodes can be efficiently handled in the star graph. Such a technique is referred to as a *multiple-sized packing* and was defined in Section 2. As we shall see, multiple-sized packings provide a nice unification of the concepts presented so far in this paper.

Various approaches can lead to a multiple-sized packing of hypercubes into the star graph. Interestingly enough, a given multiple-sized packing P_m can be transformed into another packing P'_m , such that the total number of nodes used in both P_m and P'_m is the same, but the distribution of hypercube sizes in P_m and P'_m are different. It is therefore useful to define the concept of a *template packing*, i.e. an initial multiple-sized packing upon which transformations can be applied. Assuming that P_m is a template packing of hypercubes into $S(n)$, we wish that P_m is flexible enough to support efficient transformations into many other packings. More specifically, we require that the expansion ratios of P_m and P'_m are equal, and that larger hypercubes $Q(k + \ell)$ in P'_m are created from smaller hypercubes $Q(k)$ in P_m via efficient variable-dilation techniques. Naturally, we can also create smaller hypercubes in P'_m by splitting larger hypercubes in P_m .

The template packing P_m described in this paper exhibits load 1 and expansion 1. Hypercubes in P_m range from $Q(\lfloor n/2 \rfloor)$ to $Q(n-1)$ and have a dilation of 3. The technique used to construct P_m can be summarized as follows. Initially, we pack $Q(n-1)$'s symmetrically into $S(n)$ via Algorithm 3. Using the submeshes that are left after this step, we pack $Q(n-2)$'s asymmetrically. This process continues with asymmetric packings of $Q(n-3)$, $Q(n-4)$, \dots , $Q(\lfloor n/2 \rfloor)$, always using in each step nodes of $M(n-1)$ that were not used in the previous steps. The resulting packing uses 100% of the nodes in $S(n)$, and contains symmetrically packed copies of $Q(n-1)$ that are required to support the variable-dilation embedding techniques presented in the previous section. In addition, asymmetrically packed hypercubes containing fewer than $(n-1)$ dimensions can also be used to some extent in the construction of larger hypercubes via simple extensions of our variable-dilation embedding techniques.

Theorem 9 Let $\overline{P}_m = [p_{n-1|m}, p_{n-2|m}, \dots, p_{\lfloor n/2 \rfloor|m}]$ be a vector referring to a template multiple-sized packing P_m of hypercubes into $S(n)$, such that $p_{k|m}$ indicates the number of copies of $Q(k)$ existing in P_m , $\lfloor n/2 \rfloor \leq k \leq n-1$. P_m can be built with load 1, dilation 3, and expansion 1, and is characterized by:

$$\begin{aligned}
 p_{n-1|m} &= \left\lfloor \frac{n}{2} \right\rfloor! \cdot \left\lfloor \frac{n-1}{2} \right\rfloor! \\
 p_{n-2|m} &= p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_1} \right) \\
 p_{n-3|m} &= p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-3}{2} \rfloor} \left(\frac{1}{j_1} \sum_{j_2=j_1+1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_2} \right) \right) \\
 p_{n-4|m} &= p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-5}{2} \rfloor} \left(\frac{1}{j_1} \sum_{j_2=j_1+1}^{\lfloor \frac{n-3}{2} \rfloor} \left(\frac{1}{j_2} \sum_{j_3=j_2+1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_3} \right) \right) \right)
 \end{aligned}$$

$$\begin{aligned}
& \vdots \\
p_{n-t|m} &= p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-2t+3}{2} \rfloor} \left(\frac{1}{j_1} \sum_{j_2=j_1+1}^{\lfloor \frac{n-2t+5}{2} \rfloor} \left(\frac{1}{j_2} \cdots \sum_{j_{t-2}=j_{t-3}+1}^{\lfloor \frac{n-3}{2} \rfloor} \left(\frac{1}{j_{t-2}} \sum_{j_{t-1}=j_{t-2}+1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_{t-1}} \right) \cdots \right) \right) \right) \\
& \vdots \\
p_{\lfloor n/2 \rfloor |m} &= p_{n-1|m} \cdot \left(\prod_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j} \right)
\end{aligned}$$

Proof: We build P_m by initially packing $Q(n-1)$'s symmetrically into $S(n)$ via Algorithm 3. Hence, derivation of $p_{n-1|m}$ follows by direct application of Theorem 1.

We now pack $Q(n-2)$'s in the submeshes remaining from the partitioning accomplished by Algorithm 3. We adopt the same principle that was described in the proof of Theorem 3, i.e. we pack $Q(n-2)$'s in each $(n-2)$ -dimensional submesh $M_{i_1}(n-1)$, where $M_{i_1}(n-1)$ is the induced submesh formed by all nodes $m[\] \in V(M(n-1))$, such that $m[i_1] = i_1$. Algorithm 3 produces $\lfloor (n-1)/2 \rfloor$ such submeshes, since all nodes having $m[i_1] = i_1$ along odd-sized dimensions of $M(n-1)$ are not used to pack $Q(n-1)$'s. If we call $i_1 = 2j_1$, then the available submeshes are $M_{2j_1}(n-1)$, $1 \leq j_1 \leq \lfloor (n-1)/2 \rfloor$. To pack $Q(n-2)$'s, we partition each $M_{2j_1}(n-1)$ into slices of width 2 along all its dimensions. The partitioning of $M_{2j_1}(n-1)$ produces $p_{n-2}(2j_1)$ $(n-2)$ -dimensional submeshes of width 2 along any dimension, where:

$$p_{n-2}(2j_1) = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \cdots \times \left\lfloor \frac{2j_1}{2} \right\rfloor \times \left\lfloor \frac{2j_1+2}{2} \right\rfloor \times \cdots \times \left\lfloor \frac{n}{2} \right\rfloor = \frac{p_{n-1|m}}{j_1}$$

Since two induced submeshes $M_{2j_1}(n-1)$ and $M_{2j'_1}(n-1)$, $j_1 \neq j'_1$, do not share any common nodes, the total number of $Q(n-2)$'s that can be packed after every $M_{2j_1}(n-1)$ has been partitioned is:

$$p_{n-2|m} = \sum_{j_1=1}^{\lfloor \frac{n-1}{2} \rfloor} p_{n-2}(2j_1) = p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_1} \right)$$

Partitioning $M_{2j_1}(n-1)$ as described leaves unused nodes if $n \geq 5$. Let $M_{i_1, i_2}(n-1)$ be the $(n-3)$ -dimensional induced submesh formed by all nodes $m[\] \in V(M(n-1))$, such that $m[i_1] = i_1$ and $m[i_2] = i_2$. The partitioning of $M_{2j_1}(n-1)$ produces one induced submesh $M_{i_1, i_2}(n-1)$, for each even i_2 , $i_2 \neq i_1$. That is, both i_1 and i_2 are odd-sized dimensions in $M(n-1)$, and $M_{i_1, i_2}(n-1)$ contains nodes that are not used either to pack $Q(n-1)$'s or $Q(n-2)$'s. We use such nodes to pack $Q(n-3)$'s as follows. Let $i_1 = 2j_1$ and $i_2 = 2j_2$. We partition each $M_{2j_1, 2j_2}(n-1)$ into slices of width 2 along all its dimensions. Such a partitioning produces $p_{n-3}(2j_1, 2j_2)$ $(n-3)$ -dimensional submeshes of width 2 along any dimension, where:

$$p_{n-3}(2j_1, 2j_2) = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \cdots \times \left\lfloor \frac{2j_1}{2} \right\rfloor \times \left\lfloor \frac{2j_1+2}{2} \right\rfloor \times \cdots \times \left\lfloor \frac{2j_2}{2} \right\rfloor \times \left\lfloor \frac{2j_2+2}{2} \right\rfloor \times \cdots \times \left\lfloor \frac{n}{2} \right\rfloor = \frac{p_{n-1|m}}{j_1 \cdot j_2}$$

To compute the number of $Q(n-3)$'s in P_m (i.e., $p_{n-3|m}$), we must consider all combinations of j_1 and j_2 , such that $j_1 \neq j_2$ and $1 \leq j_1, j_2 \leq \lfloor (n-1)/2 \rfloor$. We derive $p_{n-3|m}$ by taking at each time a fixed value of j_1 , $1 \leq j_1 \leq \lfloor (n-1)/2 \rfloor - 1 = \lfloor (n-3)/2 \rfloor$, and by varying j_2 accordingly from $j_1 + 1$ to $\lfloor (n-1)/2 \rfloor$. Hence:

$$p_{n-3|m} = \sum_{j_1=1}^{\lfloor \frac{n-3}{2} \rfloor} \sum_{j_2=j_1+1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{p_{n-1|m}}{j_1 \cdot j_2} = p_{n-1|m} \cdot \left(\sum_{j_1=1}^{\lfloor \frac{n-3}{2} \rfloor} \left(\frac{1}{j_1} \sum_{j_2=j_1+1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j_2} \right) \right)$$

This reasoning can be extended to pack $Q(n-4)$'s, $Q(n-5)$'s, and so on, as long as there are unused nodes left in $M(n-1)$. Expressions for $p_{n-t|m}$, $t \geq 4$, can in fact be derived via the same approach adopted for the case $t = 3$. Note that whenever the process used to build P_m moves to another class of lower dimensional hypercubes, we use induced submeshes containing one less odd-sized dimension than those used in the previous step. The process eventually finishes with a single induced submesh $M_{2,4,\dots,2\lfloor(n-1)/2\rfloor}(n-1)$, formed by all nodes $m[\cdot] \in V(M(n-1))$, such that $m[2] = 2, m[4] = 4, \dots, m[2\lfloor(n-1)/2\rfloor] = 2\lfloor(n-1)/2\rfloor$. In other words, $M_{2,4,\dots,2\lfloor(n-1)/2\rfloor}(n-1)$ is a $\lfloor n/2 \rfloor$ -dimensional submesh, formed by the nodes with maximum coordinates along all odd-sized dimensions in $M(n-1)$. $M_{2,4,\dots,2\lfloor(n-1)/2\rfloor}(n-1)$ contains only even-sized dimensions, and can be fully utilized to pack $Q(\lfloor n/2 \rfloor)$'s once it is partitioned in slices of width 2 along all its dimensions. The resulting number of $Q(\lfloor n/2 \rfloor)$'s in P_m is:

$$p_{\lfloor n/2 \rfloor|m} = \left\lfloor \frac{2}{2} \right\rfloor \times \left\lfloor \frac{4}{2} \right\rfloor \times \left\lfloor \frac{6}{2} \right\rfloor \times \dots \times \left\lfloor \frac{n}{2} \right\rfloor = p_{n-1|m} \cdot \left(\prod_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{j} \right)$$

P_m uses all of the nodes in $M(n-1)$, meaning that we have a load 1, dilation 1 and expansion 1 multiple-sized packing. If we now embed $M(n-1)$ into $S(n)$ via Algorithm 1, we obtain the load, dilation and expansion claimed in the theorem. \square

Note that P_m combines the concepts of symmetric and asymmetric packings to achieve an efficient, yet flexible utilization of $S(n)$. Symmetric packings are used for $Q(n-1)$'s and $Q(\lfloor n/2 \rfloor)$'s, while asymmetric packings are used in all other cases.

Table 6 lists template multiple-sized packings P_m for the cases $3 \leq n \leq 10$. It is interesting to note that the smaller hypercubes (e.g., $Q(\lfloor n/2 \rfloor)$) use only a minor portion of the nodes in $S(n)$. For example, packed $Q(\lfloor n/2 \rfloor)$'s use only 0.11% of the nodes in $S(9)$ and $S(10)$.

$S(n)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$	$S(8)$	$S(9)$	$S(10)$
No. of $Q(n-1)$'s	1	2	4	12	36	144	576	2,880
No. of $Q(n-2)$'s	1	2	6	18	66	264	1,200	6,000
No. of $Q(n-3)$'s	-	-	2	6	36	144	840	4,200
No. of $Q(n-4)$'s	-	-	-	-	6	24	240	1,200
No. of $Q(n-5)$'s	-	-	-	-	-	-	24	120

Table 6: Template multiple-sized packings of hypercubes into $S(n)$

We now illustrate the flexibility existing in our multiple-sized packing techniques by listing in Table 7 a few among the many possible transformations that can be applied to a template packing P_m . Table 7 refers only to the case $n = 8$, but similar tables can be constructed for other values of n . We point that quantities marked with a * in Table 7 are obtained via variable-dilation embedding techniques.

Packing	P_m^0	P_m^1	P_m^2	P_m^3	P_m^4	P_m^5	P_m^6	P_m^7	P_m^8
No. of $Q(n-4)$'s	24	-	-	-	-	-	-	-	-
No. of $Q(n-3)$'s	144	12*	-	-	-	-	-	-	-
No. of $Q(n-2)$'s	264	72*	6*	-	-	-	-	-	-
No. of $Q(n-1)$'s	144	132*	36*	3*	1*	1*	1*	1*	144 + 171*
No. of $Q(n)$'s	-	72*	66*	18*	3*	3*	3*	3*	-
No. of $Q(n+1)$'s	-	-	36*	33*	9*	3*	3*	3*	-
No. of $Q(n+2)$'s	-	-	-	18*	16*	5*	3*	3*	-
No. of $Q(n+3)$'s	-	-	-	-	9*	8*	3*	3*	-
No. of $Q(n+4)$'s	-	-	-	-	-	4*	3*	1*	-
No. of $Q(n+5)$'s	-	-	-	-	-	-	2*	1*	-
No. of $Q(n+6)$'s	-	-	-	-	-	-	-	1*	-

Table 7: Some possible multiple-sized packings of hypercubes into $S(8)$

The first packing listed in Table 7 is the template multiple-sized packing for $S(8)$, and is referred to as P_m^0 in the table just for convenience of notation. Starting from P_m^0 , we apply consecutive transformations to construct packings P_m^1 through P_m^7 as follows. $Q(k+1)$'s are constructed in P_m^{i+1} by connecting in P_m^i $Q(k)$'s via an extension of the variable-dilation techniques presented in the previous section. We recall that such techniques require symmetrically packed hypercubes, and in P_m we find both categories of packings. However, a closer look at the proof of Theorem 9 reveals that within the context of each induced submesh $M_{i_1, i_2, \dots, i_{t-1}}(n-1)$, $Q(n-t)$'s can be packed symmetrically. Hence, the extended variable-dilation embedding techniques that we assume here restrict the connection of asymmetrically packed $Q(n-t)$'s to occur only within the corresponding induced submesh containing them. Details or algorithmic descriptions of such techniques will not be presented here due to space limitations.

The concept of limiting the scope of application of variable-dilation techniques in a multiple-sized packing can be better understood with an example. Consider, for instance, the $Q(n-2)$'s existing in P_m . Using the notation adopted in the proof of Theorem 9, the number of $Q(n-2)$'s that are packed into $M_2(n-1)$, $M_4(n-1)$, and $M_6(n-1)$ are respectively $p_{n-2}(2) = p_{n-1|m}/1$, $p_{n-2}(4) = p_{n-1|m}/2$, and $p_{n-2}(6) = p_{n-1|m}/3$. Since in $S(8)$ we have $p_{n-1|m} = 144$, we obtain $p_{n-2}(2) = 144$, $p_{n-2}(4) = 72$, and $p_{n-2}(6) = 48$. We now describe how these $Q(n-2)$'s can be combined to contribute to packings P_m^1 to P_m^7 . By connecting $Q(n-2)$'s according to the restriction we just described, we obtain in P_m^1 a contribution of $144/2 + 72/2 + 48/2 = 72 + 36 + 24 = 132$ $Q(n-1)$'s. This procedure is repeated to obtain P_m^2 , giving $36 + 18 + 12 = 66$ $Q(n)$'s. Continuing, we have a contribution of $18 + 9 + 6 = 33$ $Q(n+1)$'s in P_m^3 . At this point, note that not all $Q(n+1)$'s can be combined in $M_4(n-1)$, giving a contribution of $18/2 + \lceil 9/2 \rceil + 6/2 = 16$ $Q(n+2)$'s and 1 $Q(n+1)$ in P_m^4 . This method can be easily followed to compute the contribution of the $Q(n-2)$'s initially available in P_m to packings P_m^1 to P_m^7 . Accordingly, the contribution of the remaining hypercubes in P_m follows the same reasoning. By adding up all such contributions, we obtain the quantities shown in Table 7. In any case, the expansion of the packing is 1. The average dilation of hypercubes constructed via variable-dilation embedding techniques is expected to present values similar to those plotted in Figure 10, since we employ simple extensions of our

previously discussed techniques.

As a last example of the flexibility found in our multiple-sized packings, we list in Table 7 a packing in which $Q(n-4)$'s, $Q(n-3)$'s and $Q(n-2)$'s in P_m are connected to form $Q(n-1)$'s. The resulting packing (namely, P_m^8) contains 144 $Q(n-1)$'s packed with fixed dilation and 171 $Q(n-1)$'s packed with variable dilation. This result is a nice extension of the packing techniques we presented in Section 4. By combining the concepts of asymmetry, multiple-sized packings, and variable-dilation embeddings, we use every single node remaining in $S(n)$ after $Q(n-1)$'s have been packed symmetrically, thereby achieving 100% utilization of the star graph. Using variable-dilation techniques that go beyond the scope of this paper, the $144 + 171 = 315$ $Q(n-1)$'s can be connected to form one $Q(n+7)$, which happens to be the largest hypercube that can be embedded with load 1 into $S(n)$ for $n = 8$ (see Table 4).

As a last observation, we note that due to space constraints we do not present in this paper algorithms to implement the multiple-sized packings we just described. Such algorithms, as well as any extensions of variable-dilation techniques that may be needed, can be built from the previous algorithms we presented in this paper, along with a careful observance of the corresponding partitioning processes and the submeshes they create.

7 Comparison with related work

Previous work on embedding hypercubes into star graphs has been accomplished by Nigam et al. [11] and by Miller et al. [12]. Embeddings of other classes of graphs into $S(n)$ has also been investigated, including $(n-1)$ -dimensional meshes [18], arbitrary binary trees [21], grids [22], and cycles [11], [22]. We consider here only embeddings of hypercubes into $S(n)$ and, since the techniques presented in [12] outperform those of [11], we use only the results of [12] in the following comparison with our embedding and packing techniques.

Table 8 lists the largest hypercubes that can be embedded in $S(n)$ via the variable-dilation techniques described in this paper, for $4 \leq n \leq 10$. For each embedding, Table 8 depicts the smallest dilation that is achieved in [12] and the *average dilation* obtained with the techniques of Section 5. Just for accuracy, we note that the dilation 2 embedding of $Q(4)$ into $S(4)$, marked with a * in Table 8, is obtained via a one-to-many mapping (i.e., one node of $Q(4)$ can be mapped onto multiple nodes in $S(4)$). From the viewpoint of the dilation, our embeddings outperform those of [12] for $n \geq 6$.

<i>Embedding</i>	$Q(4)$ <i>into</i> $S(4)$	$Q(6)$ <i>into</i> $S(5)$	$Q(8)$ <i>into</i> $S(6)$	$Q(11)$ <i>into</i> $S(7)$	$Q(14)$ <i>into</i> $S(8)$	$Q(17)$ <i>into</i> $S(9)$	$Q(20)$ <i>into</i> $S(10)$
<i>Dilation (Miller et al.)</i>	2*	3	4	6	6	6	6
<i>Average dilation (this paper)</i>	3.25	3.33	3.38	4.18	4.21	4.25	4.25
<i>Expansion of the embedding (Miller et. al)</i>	1.50	1.88	2.81	2.46	2.46	2.77	3.46
<i>Expansion of the packing (this paper)</i>	1.00	1.00	1.00	1.00	1.00	1.00	1.00
<i>Dilation \times expansion product (Miller et al.)</i>	3.00	5.64	11.2	14.8	14.8	16.6	20.8
<i>Dilation \times expansion product (this paper)</i>	3.25	3.33	3.38	4.18	4.21	4.25	4.25

Table 8: Comparison with related work

Table 8 also depicts the corresponding expansion ratios for each embedding. If we consider solely the embedding listed at the top of each column in Table 8, then clearly the expansion ratios resulting from the techniques of [12] and the techniques presented in this paper should be equal. However, as discussed in Section 6, our multiple-sized packings can achieve 100% utilization of $S(n)$, meaning that any node of the star graph not used in the embeddings listed in Table 8 can still be used in some other packed hypercube. This certainly allows an efficient utilization of the star graph, and hence we compute our expansion ratios within the context of a *packing*. From this viewpoint, our fixed expansion ratio (i.e., 1) is always smaller than that achieved in [12].

The fact that some applications may favor dilation instead of expansion (or vice versa) makes it difficult to define a single metric combining these two factors. In Table 8, we use the *product* of the dilation and the expansion ratio, denoted by *DEP*, to do an overall comparison between our results and those of [12]. Smaller DEP values are indicative of superiority, since a smaller dilation and expansion ratio are usually preferable. As Table 8 indicates, our techniques achieve smaller DEP values than those of [12], for $n \geq 5$. Note also that in [12], the DEP figures grow rapidly with n . With our techniques, however, the same metric exhibits a slow-growing behavior. This can be attributed not only to the fact that our multiple-sized packings guarantee unitary expansion ratio, but also to the nature of our embedding techniques. Since we achieve our embeddings by connecting smaller packed hypercubes, we preserve the smaller dilation existing in these hypercubes and use higher dilation only on the newly formed hypercube dimensions. The corresponding average dilation of the embedding is usually small and has a slow-growing dependence on n , simply because regardless of n many dimensions of the embedded hypercube can always be built with small dilation (e.g., the dilation along the first $n - 1$ hypercube dimensions is always 3 under our variable-dilation techniques).

For $n = 10$, the DEP value obtained with the techniques described in this paper exhibits a 5-fold improvement when compared to the corresponding DEP value of [12], shown in Table 8. We note that other possible embeddings are presented in [12], which can lead to different DEP values. For example, consider the embeddings of $Q(16)$ into $S(10)$, $Q(19)$ into $S(10)$, and $Q(21)$ into $S(10)$, which can be obtained with dilation 3, 4, and 8, respectively [12]. The corresponding DEP values for these embeddings are 166.1, 27.7, and 13.8. Our DEP value for $S(10)$ is still significantly smaller in any of those cases.

We claim therefore that our techniques are an efficient alternative to the problem of simulating hypercubes into the star graph. Space-efficiency is obtained via multiple-sized packing techniques that guarantee utilization of all the nodes in the star graph. Time-efficiency is obtained with a small communication slowdown, provided by our variable-dilation embedding techniques.

8 Conclusion

This paper addressed the issue of packing hypercubes into the star graph. Efficient packing techniques achieving low dilation and low expansion ratios were presented. Variable-dilation embeddings resulting from connecting packed $Q(n - 1)$'s into $S(n)$ demonstrated the possibility of embedding large hypercubes into the star graph, with corresponding small expansion while still maintaining a low dilation on the average. Such an embedding technique is advantageous for different classes of algorithms that have been devised for the hypercube, providing a small communication slowdown when the star graph is used for hypercube simulation purposes. Finally, we also presented multiple-sized packings that achieve 100% utilization of the nodes in $S(n)$. Our techniques can provide the required support for node allocation and task migration strategies in applications where $S(n)$ must handle a workload of parallel algorithms originally devised for the hypercube.

References

- [1] S. B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to the n -Cube," *Proc. Int'l Conf. on Parallel Processing*, 1987, pp. 393-400.
- [2] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Transactions on Computers*, Vol. 38, No. 4, April 1989, pp. 555-566.
- [3] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Transactions on Computers*, Vol. 37, No. 7, July 1988, pp. 867-872.
- [4] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, January 1994, pp. 31-38.
- [5] C. L. Seitz, "The Cosmic Cube," *Communications of the ACM*, Vol. 28, No. 1, January 1985, pp. 22-33.
- [6] W. D. Hillis, *The Connection Machine*, Cambridge, MA: MIT Press, 1985.
- [7] J. P. Hayes and T. Mudge, "Hypercube Supercomputers," *Proceedings of the IEEE*, Vol. 77, No. 12, December 1989, pp. 1829-1841.
- [8] A. Menn and A. K. Somani, "An Efficient Sorting Algorithm for the Star Graph Interconnection Network," *Proc. Int'l Conf. on Parallel Processing*, 1990, Vol. 3, pp. 1-8.
- [9] P. Fragopoulou and S. G. Akl, "A Parallel Algorithm for Computing Fourier Transforms on the Star Graph," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 5, May 1994, pp. 525-531.
- [10] K. Kim and V. K. P. Kumar, "An Iterative Sparse Linear System Solver on Star Graphs," *Proc. Int'l Conf. on Parallel Processing*, 1991, Vol. 3, pp. 9-16.
- [11] M. Nigam, S. Sahni, and B. Krishnamurthy, "Embedding Hamiltonians and Hypercubes in Star Interconnection Networks," *Proc. Int'l Conf. Parallel Processing*, 1990, pp. 340-343.
- [12] Z. Miller, D. Pritikin, and I. H. Sudborough, "Near Embeddings of Hypercubes into Cayley Graphs on the Symmetric Group," *IEEE Transactions on Computers*, Vol. 43, No. 1, January 1994, pp. 13-22.
- [13] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kaufmann Publishers, San Mateo, California, 1992, pp. 466-469.
- [14] M.-S. Chen and K. G. Shin, "Subcube Allocation and Task Migration in Hypercube Multiprocessors," *IEEE Transactions on Computers*, Vol. 39, No. 9, September 1990, pp. 1146-1155.
- [15] O. Kang, B. M. Kim, H. Yoon, S. R. Maeng, and others, "A Graph-Based Subcube Allocation and Task Migration in Hypercube Systems," *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation*, October 1992, IEEE Computer Society Press, pp. 535-538.
- [16] S. Latifi, "Task Allocation in the Star Graph," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 11, November 1994, pp. 1220-1224.
- [17] L. N. Bhuyan and D. P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Transactions on Computers*, Vol. C-33, No. 4, April 1984, pp. 323-333.

- [18] S. Ranka, J.-C. Wang, and N. Yeh, "Embedding Meshes on the Star Graph," *Journal of Parallel and Distributed Computing* 19, 1993, pp. 131-135.
- [19] S. Latifi, "Parallel Dimension Permutations on Star Graph," *IFIP Transactions A: Computer Science and Technology*, 1993, A23, pp. 191-201.
- [20] D. E. Knuth, *The Art of Computer Programming, Vol. 1*, Addison-Wesley, 1968, pp. 73, pp. 176-177.
- [21] N. Bagherzadeh, M. Dowd, and N. Nassif, *Embedding an Arbitrary Binary Tree into the Star Graph*, Department of Electrical and Computer Engineering, University of California, Irvine, Technical Report ECE 94-02-03, February 1994.
- [22] J. S. Jwo, S. Lakshmivarahan, and S. K. Dhall, "Embedding of Cycles and Grids in Star Graphs," *Journal of Circuits, Systems, and Computers*, Vol. 1, No. 1, 1991, pp. 43-74.