On Packing and Embedding Hypercubes into Star Graphs

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

[†]Dept. of Electrical and Computer Engineering
University of California, Irvine
Irvine, CA 92717
email: mazevedo, nader@ece.uci.edu

[‡]Dept. of Electrical and Computer Engineering
University of Nevada, Las Vegas
Las Vegas, NV 89154-4026
email: latifi@jb.ee.unlv.edu

Abstract — Packing *is a graph simulation technique by which p_k node-disjoint copies of a guest graph $G(k)$ are embedded into a 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 processors of $H(n)$, load balancing and support of 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 copies of a graph $G(k)$, packed into a host graph $H(n)$ with dilation d, such as to obtain an embedding of a graph $G(k+\ell)$, $\ell > 0$, into $H(n)$. 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 (e.g., SIMD-based algorithms), 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 $G(k), G(k+1), \ldots, G(k+\ell)$ of various sizes are packed into a host graph $H(n)$. Multiple-sized packings allow tasks with different processor requirements to be allocated proper guest graphs $G(k+j)$ in $H(n)$ (variable-dilation embeddings result when $j > 0$).*

This paper focuses on the problem of packing hypercubes $Q(n-2)$ and $Q(n-1)$ into a star graph $S(n)$ with dilation 3. We show that $3 \cdot \lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!$ copies of $Q(n-2)$ or $\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!$ copies of $Q(n-1)$ can be packed into $S(n)$, with expansion $\frac{n!}{3 \cdot \lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor! \cdot 2^{n-2}}$ and $\frac{n!}{\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor! \cdot 2^{n-1}}$, respectively. We also show how to connect packed $Q(n-1)$'s to obtain a variable-dilation embedding of $Q(n-1+\ell)$, $\ell \leq \lfloor \log_2(\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!) \rfloor$, 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 also address the issue of multiple-sized packings of hypercubes into $S(n)$.

Key words — *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 of processors in the graph. These properties compare favorably with hypercube networks [3], as described in [1]

[*]This research is supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - Brazil), under the grant No. 200392/92-1.

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) were built in recent years [5], [6], [7], and various hypercube-compatible algorithms have been developed and tested in these MPPs. It is therefore of particular interest to reuse such algorithms in star graph-configured MPPs via proper hypercube simulation mechanisms.

A graph $G(k)$ modeling a particular interconnection network can be described by a set of nodes $V(G)$ and a set of links $E(G)$. 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 $G(k) = \{V(G), E(G)\}$ into $H(n) = \{V(H), E(H)\}$ consists of a mapping of $V(G)$ into $V(H)$ and of $E(G)$ into paths of $H(n)$. The set of nodes resulting from the mapping $M : V(G) \mapsto V(H)$ is $V_G(H) \subset V(H)$. 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 [8]. 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 embedding we must have $|H(n)| \geq |G(k)|$. The ratio $|H(n)|/|G(k)|$ is called the *expansion* of the mapping. The *dilation* of the embedding is the longest path in $H(n)$ used to map any link $E_{u,v} = (u, v) \in E(G)$, $u, v \in V(G)$. We refer to the expansion of an embedding with load λ and dilation d as $X(\lambda, d)$.

Different techniques have been proposed for embedding a hypercube $Q(k)$ into a star graph $S(n)$ [9], [10]. Embeddings with load 1 and dilations of 2, 3 and 4 were considered in [9]. 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 of 1, 2 and 3 were proposed in [10], 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 [10] 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 embeddings of $Q(k)$ into $S(n)$ proposed in [9] and [10] are 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 increased dilation [10].

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 embeddings of a hypercube into a star graph), a more efficient utilization of $H(n)$ can be obtained if nodes in $V(H) - V_{G_1}(H)$ are used to build additional node-disjoint copies of $G(k)$ (namely, $G_2(k), G_3(k), \ldots, G_p(k)$). A number of advantages results from such multiple embeddings, as we shall see 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 $G(k) = \{V(G), E(G)\}$ into a graph $H(n) = \{V(H), E(H)\}$ consists of a mapping of $V(G)$ into node-disjoint sets $V_{G_1}(H), V_{G_2}(H), \ldots, V_{G_p}(H) \subset V(H)$, and of $E(G)$ into paths of $H(n)$, such that p_k copies of $G(k)$ (i.e., $G_1(k), G_2(k), \ldots, G_p(k)$) are simulated in $H(n)$. These copies do not share any common processing node in $H(n)$ (i.e., $V_{G_1}(H) \cap V_{G_2}(H) \cap \ldots \cap V_{G_p}(H) = \emptyset$, 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 $P : V(G) \mapsto V_{G_1}(H), V(G) \mapsto V_{G_2}(H), \ldots, V(G) \mapsto V_{G_p}(H)$. 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:

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

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 any other algorithm 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 processor allocation strategies in this paper. Rather, we present efficient packing techniques that are required to support these strategies in star graphs when multiple hypercubes are being simulated. We refer the reader to [11] and [12] if the particular issues of task migration and processor allocation in hypercubes is of interest.

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 ($Q(k)$) and the star graph ($S(n)$). $Q(k+1)$, 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.

A well-devised packing technique can be readily extended to support *variable-dilation embeddings* and *multiple-sized packings*. A variable-dilation embedding consists of simulating a single higher dimensional graph $G(k+\ell)$, $\ell > 0$, in $H(n)$ by connecting $c(k \rightarrow k+\ell) = \prod_{i=1}^{\ell} c(k+i-1)$ packed copies of $G(k)$. 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 $\overline{d_{k+\ell}} = [d_1, d_2, \ldots, d_i, \ldots, d_{k+\ell}]$ 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 $G(k+\ell)$, $1 \leq i \leq k+\ell$. For embeddings resulting from the packing techniques described in this paper, for example, the dilation along dimension i of $G(k+\ell)$ is

$$d_i = \begin{cases} d & \text{, if } i \leq k \\ \geq d & \text{, if } k < i \leq k+\ell \end{cases} \tag{2}$$

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 $V : G(k+\ell) \mapsto H(n)$ would be $\eta(G(k+\ell) \mapsto H(n))_{max} = \max(d, d_i) = d'$, $k < i \leq k+\ell$. 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 processors in the interconnection network must operate in a lockstep fashion, causing a unique dimension to be used at every step of the algorithm. Note that the slowdown resulting from running algorithms devised according to the MIMD computational model may also be smaller than $\eta(G(k+\ell) \mapsto H(n))_{max}$, depending on how the dimensions of $G(k+\ell)$ are used and on how the processors synchronize themselves during the execution of the algorithm.

Let A be an algorithm devised to run in a SIMD version of an 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

$$\eta(G(k+\ell) \mapsto H(n)) = \frac{\sum_{i=1}^{k+\ell} \tau_i d_i}{\sum_{i=1}^{k+\ell} \tau_i} \tag{3}$$

If we make the simplifying assumptions that A is such that $\tau_i = \tau$, $\forall i$, and that $d_i = d'$ for $k < i \leq k+\ell$, Equation 3 reduces to

$$\eta(G(k+\ell) \mapsto H(n)) = d' + \frac{k}{k+\ell}(d-d') \tag{4}$$

Notice 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 $H(n)$, however, often results in a large dilation d' being required along all dimensions of $G(k+\ell)$ if the expansion ratio of the embedding is at premium. Consequently, a larger communication slowdown d' 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 $d < d'$. 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 such a variable-dilation embedding is:

$$X(\lambda, \overline{d_{k+\ell}}) = \frac{|H(n)|}{|G(k+\ell)|} = \frac{|H(n)|}{c(k \rightarrow k+\ell)|G(k)|}, \tag{5}$$

where $c(k \rightarrow k+\ell)$ is the number of packed copies of $G(k)$ required to form a $(k+\ell)$-dimensional graph $G(k+\ell)$.

The concept of connecting packed copies of a guest graph $G(k)$ can also be used for *multiple-sized packings*, in which graphs of various sizes $|G(k)|, |G(k+1)|, \ldots, |G(k+j)|, \ldots, |G(k+\ell)|$ are packed into a host graph $H(n)$. Let $c(k \rightarrow k+j)$ be the number of packed copies of $G(k)$ required to form a $(k+j)$-dimensional graph $G(k+j)$, and let p_{k+j} be the number of copies of $G(k+j)$ packed into $H(n)$. Since all packed $G(k)$'s are node-disjoint, the multiple-sized packing resulting from connecting these $G(k)$'s also yields node-disjoint graphs $G(k+j)$, $0 \leq j \leq \ell$. For $j > 0$, a graph $G(k+j)$ is embedded into $H(n)$ via a variable-dilation mapping, as described in Equation 2. Let $\overline{d_{k+j}}$ be the dilation vector resulting from simulating a graph $G(k+j)$ via packed $G(k)$'s. The expansion of a given multiple-sized packing P_m is

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

Multiple-sized packings allow tasks with different processor requirements to be allocated proper guest graphs $G(k+j)$ in $H(n)$, while still providing smaller communication slowdown since a variable-dilation embedding is used for each $G(k+j)$.

This paper considers the packing of hypercubes $Q(n-2)$ and $Q(n-1)$ into a star graph $S(n)$ with dilation 3. Our packing technique uses a two-step mapping, in which hypercubes are initially packed with dilation 1 into an $(n-1)$-dimensional mesh $M(n-1)$ of size $2 \times 3 \times \ldots \times (n-1) \times n$. This $(n-1)$-dimensional mesh is then embedded into $S(n)$ with dilation 3, load 1 and expansion 1 via the mapping proposed by Ranka et al. in [13].

We show that $3 \cdot \lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!$ copies of $Q(n-2)$ or $\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!$ copies of $Q(n-1)$ can be packed into $S(n)$, with expansion $\frac{n!}{3 \cdot \lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor! \cdot 2^{n-2}}$ and $\frac{n!}{\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor! \cdot 2^{n-1}}$, respectively. We also show how to obtain a variable-dilation embedding of $Q(n-1+\ell)$, $\ell \leq \lfloor \log_2(\lfloor n/2 \rfloor! \cdot \lfloor (n-1)/2 \rfloor!) \rfloor$, into $S(n)$. The resulting 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 $Q(n-1+\ell)$. In addition, we also address the issue of multiple-sized packings of hypercubes into star graphs.

2 Background

2.1 The hypercube

A k-dimensional hypercube graph $Q(k) = \{V(Q), E(Q)\}$ 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 $\phi = q_1 q_2 \ldots q_i \ldots q_n$ is connected to n distinct nodes, respectively labeled with strings $\phi_i = q_1 q_2 \ldots \overline{q_i} \ldots q_n$, $1 \leq i \leq n$. In other words, node ϕ is connected to other n nodes whose labels are the binary strings resulting from complementing the digit in position i in ϕ, where $1 \leq i \leq n$ [3]. A *3*-dimensional hypercube is shown in Figure 1. Note that a link connecting a node $\phi = q_1 q_2 \ldots q_i \ldots q_n$ to a node $\phi_i = q_i q_2 \ldots \overline{q_i} \ldots q_n$ is labeled i to indicate a connection along the ith dimension of $Q(k)$.

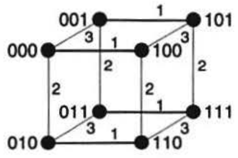

Figure 1: A *3*-dimensional hypercube $Q(3)$

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

2.2 The star graph

An n-dimensional star graph $S(n) = \{V(S), E(S)\}$ contains $n!$ nodes which are labeled with the $n!$ possible permutations of n distinct symbols. In this paper, we use the digits $\{1, 2, \ldots, n\}$ to label the nodes of $S(n)$. A node $\pi = p_1 p_2 \ldots p_i \ldots p_n$ is connected to $(n-1)$ distinct nodes, respectively labeled with permutations $\pi_i = p_i p_2 \ldots p_{i-1} p_1 p_{i+1} \ldots p_n$, $2 \leq i \leq n$. 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 $\pi = p_1 p_2 \ldots p_i \ldots p_n$ to a node $\pi_i = p_i p_2 \ldots p_{i-1} p_1 p_{i+1} \ldots p_n$ is labeled i to indicate a connection along the ith 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 vertex and edge symmetry.

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

As mentioned earlier, 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 $2 \times 3 \times \ldots \times (n-1) \times n$, namely $M(n-1) = \{V(M), E(M)\}$. $M(n-1)$ is then embedded into $S(n)$ with load 1, dilation 3, and

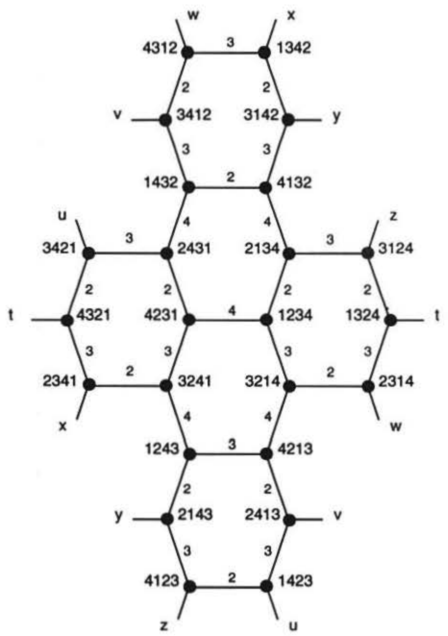

Figure 2: A *4*-dimensional star graph $S(4)$

expansion 1 via the mapping proposed by Ranka et al. in [13]. 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 $m_1m_2\dots m_{n-1}$, $0 \leq m_i \leq s_i - 1$, where $s_i = i+1$ is the size of the mesh along the ith dimension.

The mapping of a node $\omega = m_1m_2\dots m_{n-1}$, $\omega \in V(M)$, onto a node $\pi = p_1p_2\dots p_n$, $\pi \in V(S)$, is accomplished by the algorithm shown below. We assume that the identity node of $M(n-1)$ $(00\dots0)$ maps onto the identity node of $S(n)$ $(12\dots n)$.

<u>Algorithm 1 (Mapping $M(n-1)$ onto $S(n)$):</u>

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

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 external *for* loop results in the following sequence of transpositions:

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

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 rth and the sth 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.

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 (12) 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 (3 4) (2 3) (1 2). Such sequence affects π as shown below:

$$2134 \rightarrow 2143 \rightarrow 2413 \rightarrow 4213$$

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

Dimension (i)	*Sequence of transpositions*
1	(1 2)
2	(2 3) (1 2)
$\vdots$	$\vdots$
$n-1$	$(n-1\ \ n)\ (n-2\ \ n-1) \cdots (1\ 2)$

Table 1: Sequences of transpositions used by Algorithm 1

The mapping algorithm described in this paper differs slightly from that proposed in [13] in respect to the definition of a transposition $(r\ s)$. In Algorithm 1, $(r\ s)$ corresponds to an exchange of the digits occupying the rth and the sth positions in permutation π. In the corresponding algorithm described in [13], $(r\ s)$ corresponds to an exchange of *digits* r and s in π. Both approaches however result in a correct one-to-one mapping of $M(n-1)$ onto $S(n)$.

As shown in Table 1, communication between two nodes $\omega, \omega_i \in V(M)$ which are adjacent over the $i$th dimension of $M(n-1)$ requires a transposition $(r\ s)$ which must be properly accomplished by means of star operations that are available in $S(n)$. Let g be the star operation performed along the gth dimension of $S(n)$ (i.e., g exchanges the first and the gth digit of a permutation of n digits). Assume that transposition $(r\ s)$ is ordered such that $r < s$. Therefore, $(r\ s)$ can be minimally executed by the following sequences of star operations:

$$(r\ s) \equiv \begin{cases} s & \text{, if } r = 1 \\ s \rightarrow r \rightarrow s \equiv r \rightarrow s \rightarrow r & \text{, if } r, s \neq 1 \end{cases} \tag{6}$$

Note that transposition $(r\ s)$ requires 1 star operation if $r = 1$ and 3 star operations otherwise. This result can be extended to the following Lemma [13]:

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

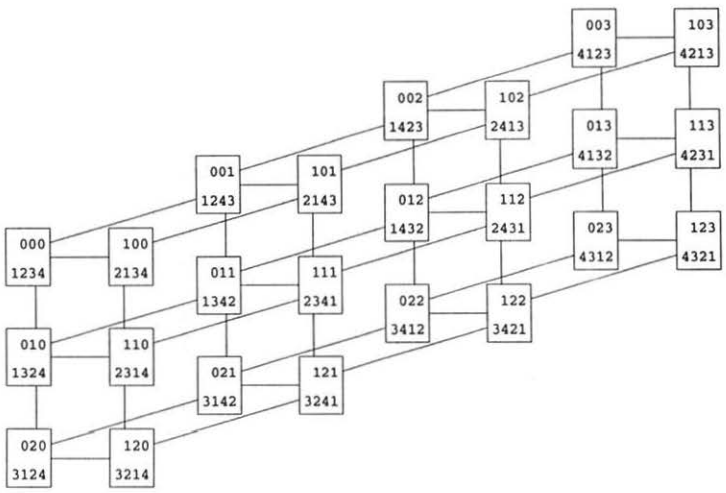

Figure 3: Mapping of $M(3)$ onto $S(4)$

3 Packing Hypercubes into Star Graphs

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 processors of $S(n)$. An important result upon 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 due the fact that 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 $s_1 \times s_2 \times \ldots \times s_k = 2 \times 2 \times \ldots \times 2$, which corresponds to using a limited range of the coordinates available in $M(n-1)$. □

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 will be presenting next. Two distinct cases are considered in the remainder of this section: packing $Q(n-1)$ into $M(n-1)$ and packing $Q(n-2)$ into $M(n-1)$. Our results are then extended via a subsequent embedding of $M(n-1)$ into $S(n)$ using the mapping given by Algorithm 1.

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

Theorem 1 *It is possible to 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*

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

Proof: The size of $M(n-1)$ along the ith dimension is $s_i = i+1$ (i.e., the ith 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

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

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. Hence, it is possible to 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 [13]. 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 $|Q(n-1)| = 2^{n-1}$.□

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

Algorithm 3 (Packing $Q(n-1)$ into $S(n)$):

```
pack_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++) {
    $offset[i] = 2\left(\left\lfloor \frac{C}{\lfloor \frac{i}{2} \rfloor ! \times \lfloor \frac{i-1}{2} \rfloor !} \right\rfloor \bmod \left\lfloor \frac{i+1}{2} \right\rfloor\right)$
    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 Cth 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 Cth 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 pack 12 copies of $Q(5)$ into $S(6)$.

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

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

Theorem 2 *It is possible to 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*

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

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

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

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. 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.□

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

Algorithm 4 (Packing $Q(n-2)$ into $S(n)$):

```
pack_cube_n_2 (int $q[\,]$, int $C$, int $n$, int $p[\,]$)
{
  int $i$, $offset[\,]$, $m[\,]$;
  $m[1] = q[1]$;
  $m[2] = C \bmod 3$;
  for ($i = 3$; $i < n$; $i++$) {
    $offset[i] = 2\left(\left\lfloor \frac{C}{3 \times \lfloor \frac{i}{2} \rfloor! \times \lfloor \frac{i-1}{2} \rfloor!} \right\rfloor \bmod \left\lfloor \frac{i+1}{2} \right\rfloor\right)$
    $m[i] = q[i-1] + offset[i]$;
  }
  mesh_to_star ($m[\,]$, $n$, $p[\,]$)
}
```

3.3 Results on packing $Q(n-1)$ and $Q(n-2)$ into $S(n)$

Table 3 depicts the number of packed hypercubes and the expansion ratios resulting from the techniques described in this section. Note that low expansion ratios are obtained, meaning that a large portion of the nodes of $S(n)$ are used in the packing. Smaller expansion ratios are obtained when packing $Q(n-2)$ into $S(n)$, since a more efficient partitioning strategy of $M(n-1)$ is possible in this case. In addition, note that a slight increase in the expansion ratios occurs whenever n is incremented to an odd number, which can also be explained by the partitioning strategy described in the proofs of Theorems 1 and 2. Finally, we observe that the unused nodes resulting from the partitioning process could be used for packing additional hypercubes at the expense of higher dilation.

$S(n)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$	$S(8)$	$S(9)$	$S(10)$
No. of packed $Q(n-1)$*'s* (p_{n-1})	1	2	4	12	36	144	576	2,880
Expansion ratio ($X(1,3,p_{n-1})$)	1.50	1.50	1.88	1.88	2.19	2.19	2.46	2.46
No. of packed $Q(n-2)$*'s* (p_{n-2})	3	6	12	36	108	432	1,728	8,640
Expansion ratio ($X(1,3,p_{n-1})$)	1.00	1.00	1.25	1.25	1.46	1.46	1.64	1.64

Table 3: Results on packing $Q(n-1)$ and $Q(n-2)$ into $S(n)$

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

In this section, we describe how an $(n-1+\ell)$-dimensional hypercube can be embedded with variable dilation into $S(n)$. 2^ℓ packed copies of $Q(n-1)$ are required for such an embedding, where ℓ is limited by

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

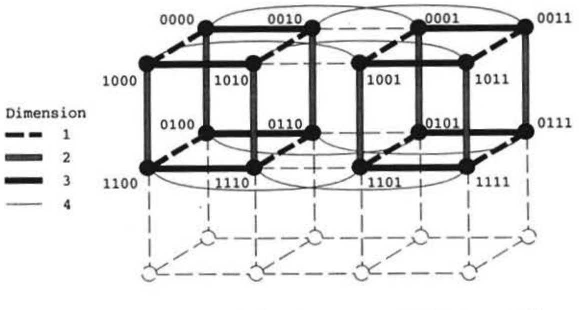

Figure 4: A variable-dilation embedding of $Q(4)$ into $M(3)$

Figure 4 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 ith dimension of $Q(4)$ is

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

As defined in section 1, Figure 4 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 $\omega = m_1 m_2 \ldots m_i \ldots m_{n-1}$, $\omega_{i+2} = m_1 m_2 \ldots (m_i + 2) \ldots m_{n-1} \in V(M)$ be a pair of nodes separated by 2 links along the ith dimension of $M(n-1)$, $2 \leq i \leq n-1$. In the corresponding embedding of $M(n-1)$ into $S(n)$, ω and ω_{i+2} are connected by a path containing at most 4 links.*

Proof: By inspection of Algorithm 1 and Table 1, we note that the mapping of $\omega_{i+2} \in V(M)$ onto $\pi_{i+2} \in V(S)$ uses a sequence of transpositions of the form:

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

Accordingly, ω is mapped onto π via a sequence of transpositions of the form:

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

Although identical transpositions are used in σ and σ_{i+2}, the mapping sequence of ω_{i+2} has two transpositions more than the mapping sequence of ω_i. An inspection of Table 1 reveals that the extra transpositions $(k\ l)$ and $(m\ n)$ are actually two consecutive transpositions along the ith dimension, whose generic format is $(x\ x+1)(x-1\ x) = (s\ t)(r\ s), r = s - 1 = t - 2$. According to Equation 6, these two transpositions can be accomplished via a sequence of star operations as follows:

$$(s\ t)(r\ s) \equiv \begin{cases} s \rightarrow t \rightarrow \not{s} \rightarrow \not{s} = s \rightarrow t & \text{, if } r = 1 \\ s \rightarrow t \rightarrow \not{s} \rightarrow \not{s} \rightarrow r \rightarrow s = s \rightarrow t \rightarrow r \rightarrow s & \text{, if } r \neq 1 \end{cases} \tag{8}$$

Note that the execution of the two transpositions require either 2 or 4 star operations. We can actually concatenate $(s\ t)(r\ s)$ into a single *permutation cycle* [14] of the form $(r\ s\ t)$. Let the digits occupying the rth, the sth and the tth positions of the permutation resulting from applying transpositions $(a\ b)(c\ d)\ldots(i\ j) \in \sigma, \sigma_{i+2}$ to the identity $123\ldots n$ be respectively d_1, d_2 and d_3. Therefore, cycle $(r\ s\ t)$ moves d_1 into d_2's position, d_2 into d_3's position, and d_3 into d_1's position. Since σ and σ_{i+2} have the same transpositions except for the cycle $(r\ s\ t)$, π and π_{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. Therefore, if in π we have $p_\alpha = d_1$, $p_\beta = d_2$, and $p_\gamma = d_3$, in π_{i+2} we have $p_\alpha = d_3$, $p_\beta = d_1$ and $p_\gamma = d_2$. Therefore, routing from π to π_{i+2} requires a single cycle $(\alpha\ \beta\ \gamma)$, which corresponds to a path containing at most 4 links in $S(n)$ as indicated by Equation 8. □

We now state without proof the following lemma (proof is actually analogous to that of Lemma 3):

Lemma 4 *Let $\omega = m_1 m_2 \ldots m_i \ldots m_{n-1}$, $\omega_{i+2} = m_1 m_2 \ldots (m_i + 4) \ldots m_{n-1} \in V(M)$ be a pair of nodes separated by 4 links along the ith dimension of $M(n-1)$, $4 \le i \le n-1$. In the corresponding embedding of $M(n-1)$ into $S(n)$, ω and ω_{i+4} are connected by a path containing at most 6 links.*

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

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

Proof: Due to the partitioning process described in the proof of Theorem 1, at least 2 equal-sized sets of packed $Q(n-1)$'s can be found if $M(n-1)$ is traversed along dimension i, $i \ge 3$. If we connect the packed hypercubes along ℓ of these dimensions as shown in Figure 4, a variable-dilation embedding of $Q(n-1+\ell)$ into $M(n-1)$ results. The dilation vector of such an embedding is:

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

Note that an extra hypercube dimension can be created with dilation 2 along each dimension $i \ge 3$ via the technique depicted in Figure 4. Since in $M(n-1)$ there are $(n-3)$ such dimensions, the embedding must observe the constraint $0 \le \ell \le n-3$.

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 5. □

Theorem 4 *For $n \ge 8$, there is a load 1, variable-dilation embedding of $Q(n-1+\ell)$ into $S(n)$, $0 < \ell \le 2n-10$, whose dilation vector and expansion ratio are respectively*

$$d_i = \begin{cases} 3 & , \textit{if } i \le n-1 \\ 4 & , \textit{if } n \le i \le 2n-4 \textit{ and } i \le \ell \\ 6 & , \textit{if } 2n-3 \le i \le \ell \end{cases} \qquad \textit{and} \qquad X(1, \overline{d_{n-1+\ell}}) = \frac{n!}{2^{n-1-\ell}} \ge \frac{n!}{2^{3n-11}}$$

Proof: As described in the proof of Theorem 3, for $n \ge 4$ up to 2^{n-3} packed $Q(n-1)$'s can be connected in $M(n-1)$, resulting in an $(2n-4)$-dimensional hypercube whose dilation vector is $\overline{d_{2n-4}} = [1, 1, \ldots, 1, 2, 2, \ldots, 2]$.

For $n \ge 8$, there are $(n-7)$ dimensions along which $M(n-1)$'s size is $s_i \ge 8$. In addition, at most one half of the packed $Q(n-1)$'s along these dimensions is used by the technique depicted in Figure 4 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, higher dimensional hypercubes result. In fact, a new hypercube dimension can be created with dilation 4 by connecting $Q(2n-4)$'s along each dimension $i \ge 7$ in $M(n-1)$, as shown in Figure 5. 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 $M(n-1)$:

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

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 5. □

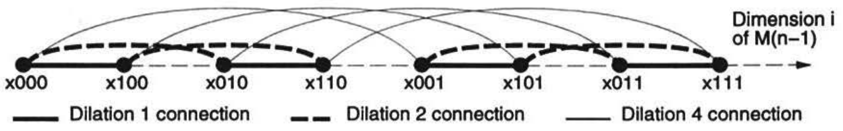

Figure 5: Embedding 3 dimensions of $Q(n-1+\ell)$ along the ith dimension of $M(n-1)$, $i \geq 7$

Communication slowdown resulting from variable-dilation embeddings

As explained in Section 1, variable-dilation embeddings can achieve a considerably smaller communication slowdown for certain classes of algorithms. Figure 6 gives an estimate for the communication slowdown $\eta(Q(k) \mapsto S(n))$, which was computed from Equation 3 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.

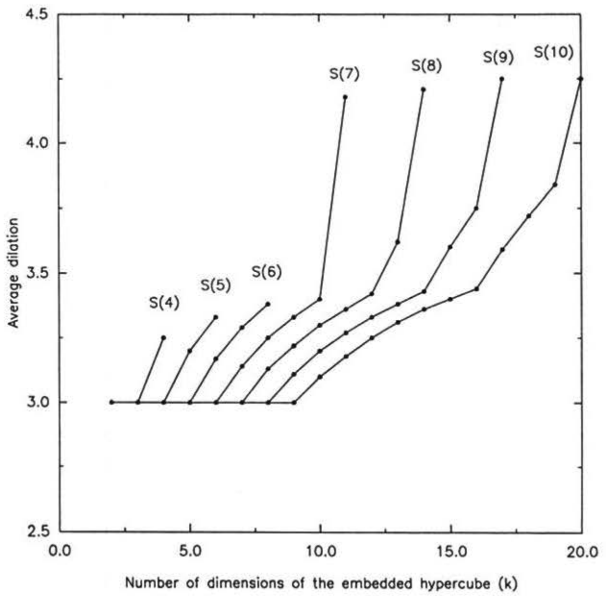

Figure 6: Average dilation of embeddings of $Q(n-1+\ell)$ into $S(n)$

In the case of $S(10)$, for example, Figure 6 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 3. 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 ratio up to 3.84. The dilation vectors used in the case $17 \leq k \leq 19$ are as defined in Theorem 4.

Finally, if Equation 7 is compared with the highest dimension k obtainable with the technique of Theorem 4 ($k = 3n - 11$), we note that it is possible to embed $Q(20)$ into $S(10)$ with variable dilation by connecting packed $Q(9)$'s properly. A special connection technique allows such an embedding, whose dilation vector is given by:

$$d_i = \begin{cases} 3 & \text{, if } i \leq n-1 \\ 4 & \text{, if } n \leq i \leq 2n-4 \\ 6 & \text{, if } 2n-3 \leq i \leq 3n-12 \\ 8 & \text{, if } 3n-11 \leq i \leq 3n-10 \end{cases} \tag{9}$$

Although a dilation of 8 results for the two highest dimensions of $Q(20)$, the average dilation is nearly one half of that value (4.25). For conciseness, we will not present the connection arrangement required to embed $Q(20)$ into $S(10)$ with variable dilation here. In fact, such an arrangement is also required for variable-dilation embeddings of $Q(11)$ into $S(7)$, $Q(14)$ into $S(8)$, and $Q(17)$ into $S(9)$.

5 Multiple-Sized Packings

Table 4 depicts packings of hypercubes of various sizes into $S(n)$. These packings result from connecting $Q(n-1)$'s, such that a variable-dilation embedding is actually required for each packed $Q(k)$, $k > n-1$, as defined in Equation 9 and Theorems 3 and 4.

Note that Table 4 simply lists how many $Q(k)$'s can be packed into $S(n)$, without truly reflecting the flexibility existing in a multiple-sized packing. It is possible, for instance, to simultaneously pack 1 $Q(16)$, 1 $Q(15)$, 2 $Q(13)$'s and 8 $Q(12)$'s into $S(9)$ via simple partitioning mechanisms. Notably, processor allocation strategies can use the concept of multiple-sized packings to efficiently map hypercube-compatible algorithms with different computational requirements into $S(n)$.

6 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 significantly smaller communication slowdown when the star graph is used for hypercube simulation purposes. Finally, we also introduced the concept of multiple-sized packings, which can provide the required support for processor allocation and task migration strategies in applications where $S(n)$ must handle a workload of parallel programs originally written for a hypercube-configured MPP.

$S(n)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$	$S(8)$	$S(9)$
No. of $Q(n-1)$*'s*	1	2	4	12	36	144	576
No. of $Q(n)$*'s*	-	1	2	6	18	72	288
No. of $Q(n+1)$*'s*	-	-	1	3	9	36	144
No. of $Q(n+2)$*'s*	-	-	-	1	4	18	72
No. of $Q(n+3)$*'s*	-	-	-	-	2	9	36
No. of $Q(n+4)$*'s*	-	-	-	-	1	4	18
No. of $Q(n+5)$*'s*	-	-	-	-	-	2	9
No. of $Q(n+6)$*'s*	-	-	-	-	-	1	4
No. of $Q(n+7)$*'s*	-	-	-	-	-	-	2
No. of $Q(n+8)$*'s*	-	-	-	-	-	-	1

Table 4: Some possible packings of $Q(k)$ into $S(n)$, $k \geq n-1$

References

[1] S. B. Akers, D. Horel 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," *Proc. Int'l Conf. on Parallel Processing*, 1986, pp. 216-223.

[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] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kauffmann Publishers, San Mateo, California, 1992, pp. 466-469.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. E. Knuth, *The Art of Computer Programming, Vol. 1*, Addison-Wesley, 1968, pp. 73, pp. 176-177.