2 to 1 Embeddings of Grids into Hypercubes

by

Dennis L. Manke

B.Sc., University of Victoria, Victoria, B.C., Canada, 1989

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the Department
of
Computer Science

We accept this thesis as conforming
to the required standard

[redacted]
Dr. John A. Ellis, Supervisor (Computer Science)

[redacted]
Dr. Michael R. Fellows, Departmental Member (Computer Science)

[redacted]
Dr. Donald J. Miller, Outside Member (Mathematics)

[redacted]
Dr. Fayez El Guibaly, External (Elec. and Computer Engineering)

Supervisor: Dr. John A. Ellis

Abstract

We consider two-to-one embeddings of grids into the next smaller Hypercube and derive novel two-to-one embedding techniques that achieve optimal dilation 1 for many grids, where in some cases no previous solutions were known. In particular, dilation 1, two-to-one embeddings into the next smaller Hypercube can be found for grids that are:

- square.
- close to square.
- of height within one of a power of 2.

Examiners:

Dr. John A. Ellis, Supervisor (Computer Science)

Dr. Michael R. Fellows, Departmental Member (Computer Science)

Dr. Donald J. Miller, Outside Member (Mathematics)

Dr. Fayez El Guibaly, External (Elec. and Computer Engineering)

Contents

List of Figures

Acknowledgements

I would like to thank my supervisor, professor John A. Ellis, for his guidance and financial support throughout my studies. I also wish to thank the department for providing excellent computing resources, which made possible the CPU intensive searches in the early stages of my research.

Chapter 1

Introduction

Over a decade ago, Rosenberg [22] proposed five motivations for studying graph embeddings. They were:

- Finding efficient storage representations for data structures.
- Laying circuits out on VLSI chips.
- Structuring programs.
- Organizing computations on a network of processors.
- Determining the bandwidth and cutwidth of a sparse matrix (dilation and congestion of an embedding into the line graph).

Of these five, "finding efficient storage representations for data structures" and "structuring programs" have received little attention. More recently, "organizing computations on a network of processors" has become the most compelling reason for studying graph embeddings. So much so, that the terms "graph" and "network" are often freely

interchanged. Indeed, much of this interest stems from the fact that many network embedding problems are now finding applications in parallel computer architectures of today [17, 15, 13] and will only play an increasing role as parallel machines continue to evolve.

1.1 Definitions and Terminology

1.1.1 Graph Framework

Before introducing definitions of interest in the field of graph embedding, we shall define our fundamental graph terminology.

- A **graph** G, denoted by $G = (V, E)$, is a set of **vertices** V and a set of **edges** E, where E is a set of unordered pairs of not necessarily distinct elements of V. The **edge set** E may be denoted by $E(G)$ and similarly the **vertex set** V may be denoted by $V(G)$.
- A **path** from vertex $v_1 \in V$ to vertex $v_n \in V$ is a sequence of vertices $(v_1, v_2, \ldots v_n)$, where $v_i \in V$ for $1 \leq i \leq n$ and $\{v_i, v_{i+1}\} \in E$ for $1 \leq i < n$.
- A path $(v_1, v_2, \ldots v_n)$ has a **cycle** if there exists $i \neq j$ such that $v_i = v_j$.
- A **simple path** is a path with no cycle.
- A path $(v_1, v_2, \ldots v_n)$ is **closed** if $v_1 = v_n$.
- The **length** of a path is the number of vertices $- 1$.
- An ***n*-cycle** is a closed simple path of length n.
- A graph G is **connected** if for all $v_i, v_j \in V$, there exists a path from v_i to v_j.

- In a graph G, vertices $v_i, v_j \in V$ are **adjacent**, denoted by v_i adj v_j, iff they are connected by an edge.

$$v_i \text{ adj } v_j \Leftrightarrow \{v_i, v_j\} \in E$$

- In a graph G, the edge $e \in E$ is **incident** to a vertex $v \in V$ iff $e = \{v, x\}$ for some $x \in V$.

- **deg(v)** denotes the **degree** of a vertex $v \in V$, and is the number of edges incident to v.

- The **number of vertices** in G is denoted by $|G|$ or by $|V|$.

- The **number of edges** in G is denoted by $|E|$.

- **dist$_G$(v_i, v_j)** denotes the length of a shortest path between vertex $v_i \in V(G)$ and vertex $v_j \in V(G)$.

- The **Diameter** of a graph G is:

$$\max_{v_i, v_j \in V(G)} \text{dist}_G(v_i, v_j)$$

- The **Bisection Width** of a graph G is the minimum number of edges that must be removed from G to split G into two unconnected graphs, G_1 and G_2, where the number of vertices in G_1 and G_2 differ by no more than 1.

1.1.2 General Graph Assumptions

We shall assume that paths are **simple** and that graphs are **connected**.

1.1.3 Embedding Framework

- An **embedding** of a (**Guest**) graph G into a (**Host**) graph H is a mapping $\phi_v : V(G) \rightarrow V(H)$ and a mapping $\phi_e : E(G) \rightarrow$ paths in H. The mappings ϕ_v and ϕ_e need not be 1-to-1 or onto.

- For an edge $e = \{v_i, v_j\} \in E(G)$ and an embedding $\phi_v : V(G) \rightarrow V(H)$, we use the notation $\mathbf{dist_H(e)}$ to denote $\text{dist}_H(\phi_v(v_i), \phi_v(v_j))$.

1.1.4 Embedding Parameters

Various **cost** measures for evaluating the effectiveness of an embedding have been investigated throughout the literature.

- The **Congestion of a Host Edge** is the number of paths in the Host (images of Guest edges) sharing the Host edge in question.

- The **Congestion of an Embedding** is the maximum congestion of an edge, over all Host edges.

- The **Dilation of an Edge** is the length of the path that is the image of a Guest edge in the Host.

- The **Dilation of an Embedding** is the maximum edge dilation over all edges in the Guest.

- The **Expansion** of an embedding is the ratio $|H|/|G|$.

- The **Load Factor of a Host Vertex** is the number of Guest vertices mapped to the Host vertex in question.

- The **Load Factor of an Embedding** is the maximum load factor over all Host vertices. We note that by the Pigeon Hole Principal:

$$\text{Load Factor} \geq \lceil 1/\text{Expansion} \rceil$$

In a parallel processing application:

- congestion of an embedding corresponds to the maximum number of messages that may have to pass over a common network link at any given time.
- dilation of an embedding corresponds to the maximum number of network links that a message may have to pass over to arrive at its destination.
- load factor of an embedding corresponds to the maximum number of processes that may have to run on a single processor.

1.1.5 General Embedding Assumptions

For embedding costs that are defined using a related "edge or vertex" cost, we are normally interested in the overall embedding cost. Therefore when the following costs are used without qualifying them, we make the following **assumptions**:

- Dilation – refers to the dilation of an embedding.
- Load – refers to the load factor of an embedding.
- Congestion – refers to the edge congestion of an embedding.

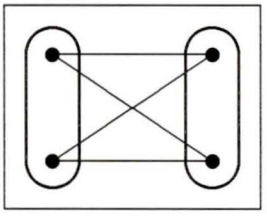
In the definition of an embedding, ϕ_e defines a mapping from Guest edges to Host paths. However, the literature on graph embeddings will in many cases avoid a specification of the mapping ϕ_e, in which case one of the **shortest paths is assumed**. This is usually done for the following reasons:

- Congestion may not be the focus of the investigation.
- Minimizing dilation (path lengths) can tend to minimize congestion.

The latter assumption is not always valid, since minimizing dilation and/or expansion while ignoring congestion can result in large congestion. Indeed, Rosenberg [23] mentions a result where an embedding construction gives constant dilation and expansion, but has horrendous congestion.

We note that since our work is also not concerned with congestion, we also follow the practice of making no references to mapped paths. Without as yet defining our problem, we point out that in our case, being oblivious to congestion will not lead to unnecessarily large congestion. Our work will deal with dilation 1, load 2 embeddings, and hence the maximum congestion will be small since congestion is bounded from above by the square of the load (4 in our case).

To show small congestion, we note that dilation 1 implies all Host paths are just single edges or vertices and that two vertices defining any edge in the Host will have a maximum of "load" Guest vertices mapped to each of them. Hence, in the worst case the Host edge will be used by all possible pairs of vertices (square of the load).



Adjacent Host Nodes
with
Potential Guest Edges

1.2 Graph Families / Parallel Networks

In parallel computer architectures, graphs are used to represent the interconnections between nodes in a network. Nodes may represent individual processors, switches for routing information through the network, or a combination of the two. Various

parallel algorithms also use graphs for modeling their processes and inter-process communication. Generally different classes of problems lend themselves to different types of graphs. These graph types are referred to as **Graph Families**. For example, matrix applications and image processing applications will often be modeled by a network where the interconnections resemble a mesh organization [15]. In contrast, problems that are solved by decomposing them in a recursive way may more readily be modeled by interconnections resembling tree shaped organizations.

1.2.1 Determining Factors of Interesting Families

How can these different graph families be simulated on a universal parallel network? We note that it is infeasible to build a network based on a complete graph. The challenge is to design networks (Graph Families) that can efficiently **simulate** the various graph families resulting from problems we see in practice. In general, this task is not easy, since the techniques for finding an efficient way of embedding one graph family into another are not well understood, and are largely composed of adhoc techniques. This is further complicated since we know that the problem of embedding any given graph into a specific graph family can be NP-complete [21]. Therefore the practice has been to develop a network that one believes will be an efficient Host for the large number of graph families we see in practice, and once having this general Host network, to then devise ways of embedding common graph families into the Host network, with some measure of efficiency placed on the embedding.

Research on general purpose networks is diverse, with numerous graph families being studied for their ability to:

- play efficient Hosts for a variety of Guests.
- be easily constructed in hardware.

It is generally assumed that the following parameters are the most significant factors influencing these traits:

- Diameter – The diameter of the network typically represents a lower bound on the communication delay incurred by a network. This is generally realized because in the design of a parallel algorithm, the two or more network nodes separated by "diameter" links in the network may be required to communicate with each other (e.g. Sorting [15], where a pair of nodes represents a pair of elements which must ultimately be compared).
- Bisection Width – In many parallel algorithms, half of the network will be required to communicate with the other half and hence a small bisection width causes communication delays due to the bottle neck (congestion).
- Max Node Degree – Affects the ability to implement the network in hardware, since there is a direct correspondence between node degree and the number of wires one must connect to a network node.

1.2.2 2-dimensional Grids (Mesh)

A **2-dimensional Grid** is denoted by $G(h, w)$, where h is called the **height** and w is called the **width**. The vertices V are labeled v_{x_1,x_2}, where $1 \leq x_1 \leq h$ and $1 \leq x_2 \leq w$. An edge

$$e = (v_{x_1,x_2}, v_{x_1',x_2'}) \in E \Leftrightarrow |x_1' - x_1| + |x_2' - x_2| = 1$$

Some properties of 2-dimensional grids are:

$$|G| = hw$$

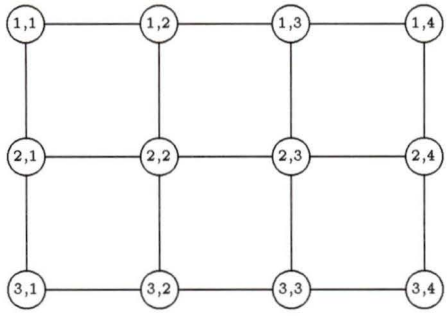


Figure 1.1: $G(3,4)$ with Vertex Labeling

$$\begin{aligned} |E| &= w(h-1) + h(w-1) \\ \text{Diameter}(G) &= h + w \\ \text{Max Degree} &= 4 \end{aligned}$$

Other general parameters that are sometimes used in describing Grids and their embeddings are:

- The **Aspect Ratio** of $G(h,w)$ is the ratio w/h.
- The **Compression Ratio** of a Grid to Grid embedding is the ratio $w(H)/w(G)$, where G is the Guest and H is the Host.

1.2.3 r-dimensional Grids (Mesh)

An **r-dimensional Grid** is denoted by $G(d_1, d_2, \ldots, d_r)$, where d_i is the size of the ith dimension. The vertices V are labeled $v_{x_1,x_2,\ldots,x_r}$, where $1 \leq x_i \leq d_i$. An edge

$$e = (v_{x_1,\ldots,x_r}, v_{x'_1,\ldots,x'_r}) \in E \Leftrightarrow \sum_{i=1}^{i=r} |x'_i - x_i| = 1$$

Some properties of r-dimensional grids are:

$$\begin{aligned} |G| &= \prod_{1 \le i \le r} d_i \\ |E| &= \sum_{1 \le i \le r} \left((d_i - 1) \prod_{i \ne j, 1 \le j \le r} d_j \right) \\ \text{Diameter}(G) &= \sum_{1 \le i \le r} d_i \\ \text{Max Degree} &= 2r \end{aligned}$$

1.2.4 Hypercubes

A **Hypercube** of dimension r is denoted by $Q(r)$. The vertices V are labeled with a binary string of length r. An edge $(v_1,\ v_2) \in E$ iff the label of v_1 and v_2 differ in exactly 1 bit. Edge $e = (v_1,\ v_2)$ is said to **cross dimension i** iff the label of v_1 and v_2 differ in the ith bit.

Some properties of Hypercubes are:

$$\begin{aligned} |G| &= 2^r \\ |E| &= 2^{r-1} r \\ \text{Diameter}(G) &= r \\ \text{Degree} &= r \end{aligned}$$

See Figure 1.2 for an illustration of $Q(3)$ with vertex labeling in both binary and decimal.

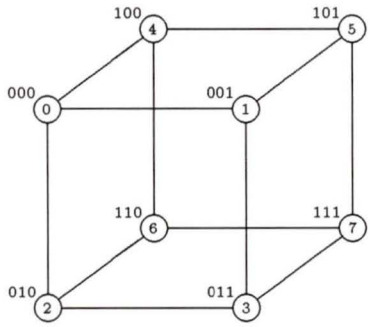


Figure 1.2: $Q(3)$ with Vertex Labeling

1.2.5 Other Families Relevant to Parallel Architectures

Many families of graphs have been and are actively studied throughout the literature. Here are some of them:

- **Tree Networks:** Trees provide a natural organization for recursive algorithms. They include variants [4, 20, 15], such as those having additional links (e.g. x-trees), or those with tree nodes being complex graphs themselves.

- **Butterfly Networks:** Butterfly networks (sometimes called FFT networks) provide a structure initially derived for efficiently solving parallel implementations of the radix 2 Fast Fourier Transformation (FFT). Butterfly nodes have fixed degree 4, thereby enhancing the networks scalability to larger networks. Despite the networks fixed degree nodes, butterflies can efficiently simulate many of the graph families that a Hypercube can [3, 12, 20, 15].

- **Shuffle-Exchange Networks:** Shuffle-exchange networks are a fixed degree variant of the Hypercube. Though shuffle-exchange networks appear dissimilar

from butterfly networks, they can be shown to be computationally equivalent [20, 15].

1.3 Embedding Representations

We formulate two ways of representing an embedding:

- Edge Mappings
- Vertex Mappings

1.3.1 Edge Mappings

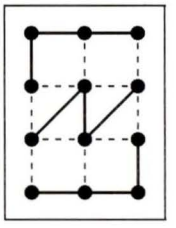


Figure 1.3: 3×4 to 4×3

An **edge mapping** is portrayed by superposing the mapped Guest edges over a diagram of the Host. Figure 1.3 illustrates a dilation 2, Load 1 embedding of $G(3,4)$ into $G(4,3)$. When the Guest is a Grid, the convention is to illustrate only the horizontal (row) edges of the Grid as they appear mapped in the Host. This results in no loss of information, since the mapped vertical (column) edges of the Grid can be inferred, and in this way, figures are kept less cluttered and more readable. The diagonal edges of Figure 1.3 are dilation 2 edges with no particular path illustrated (there are two length 2 paths that traverse a 4-cycle to the opposite vertex).

1.3.2 Vertex Mappings

A **vertex mapping** is portrayed by a diagram of the Guest, with Guest vertices containing the labels of the Host vertices that each Guest vertex maps to (image of the Guest vertices). Figure 1.4 illustrates a vertex mapping equivalent to the edge mapping of Figure 1.3. Under this representation, when the Guest is a Grid, we specify where vertices of the Grid are mapped in a matrix form. Since no edges are drawn, the diagram is uncluttered regardless of how complex the embedding may be.

2,1	1,1	1,2	1,3
2,3	2,2	3,2	2,3
4,1	4,2	4,3	3,4

Figure 1.4: 3×4 to 4×3

1.4 Embedding Types

The goal we have when embedding a Guest network into a Host network is to find an efficient way of simulating the Guest on the Host. Quantifying the efficiency of an embedding is usually done with respect to:

- Dilation — influences the delays in communication due to the expansion of Guest edges under the Host simulation.
- Congestion — influences the communication delays due to the multiplexing of several Guest edges on single Host edges under the Host simulation.
- Expansion — governs the efficient use of Host vertices. Larger expansion means more Host vertices go unused.
- Dilation/Congestion/Expansion Tradeoffs — it is sometimes impossible to minimize one cost without increasing the other(s). Indeed, algorithms exist that optimize dilation by increasing the expansion.

We must also be aware of the size of the Guest with respect to a particular Host (embedding expansion). It is standard practice to develop parallel algorithms and their associated networks by assuming we have as many processors available as required. This is an unrealistic assumption, since in practice we have a fixed size machine. However, algorithms assuming a 1-to-1 mapping can usually be adapted to many-to-1 mappings [15], but designing an embedding strategy tailored towards a specific load factor can result in optimizations otherwise not realized. Therefore, when looking for optimal embedding schemes, we pursue different strategies for various load factors and the associated smallest Host. Hence, embeddings are usually studied in the following ways:

- One-to-One Embeddings
- Two-to-One Embeddings
- Many-to-One Embeddings
- One-to-Many Embeddings

We use the following terminology for specifying a Host:

- The **Smallest Hypercube** is the Hypercube $Q(d)$, where d is the smallest such that $|\text{Guest}| \leq \text{load} \cdot |Q(d)|$.
- The **Next Smaller Hypercube** is the Hypercube $Q(d)$, where d is the smallest such that $|\text{Guest}| \leq 2|Q(d)|$. We use this terminology to make clear the size of the Hypercube under a load 2 embedding.
- The **Smallest Grid** is for a given width w, the Grid $G(h, w)$, where h is the smallest such that $|\text{Guest}| \leq hw$.

1.4.1 One to One Embeddings

A one to one embedding is a one-to-one (i.e. Load 1) mapping from Guest vertices to Host vertices. The challenge is to minimize dilation and/or congestion.

With many graph families, lower bounds on dilation and congestion have been shown and in some cases, these lower bounds have been achieved. Figure 1.5 illustrates such an example, where dilation 2 is optimal and achievable under a one to one embedding of $G(3,5)$ into $G(4,4)$. Dilation 2 is optimal in Figure 1.5, since $G(3,5)$ is not a subgraph of $G(4,4)$.

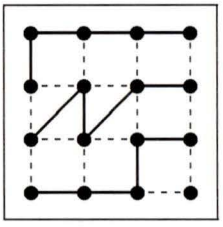


Figure 1.5: 3×5 to 4×4

1.4.2 Two to One Embeddings

In a two to one embedding strategy, we search for a two-to-one mapping (i.e. Load 2 embedding) for the following reasons:

- To cover the situation when $1/2 \leq$ Expansion < 1.
- To provide insight into solving the general problem where the Host is many times smaller than the Guest. For a given Host having fewer vertices than the Guest (Expansion < 1), we wish to solve for the minimum possible load (i.e. Load $= \lceil 1/\text{Expansion} \rceil$).
- To find embeddings where dilation and/or congestion is less than the best known dilation and/or congestion of the corresponding one-to-one embeddings.

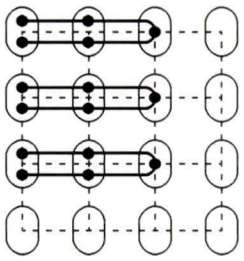


Figure 1.6: 3×5 to 4×4

In a two-to-one embedding, Guest vertices that were adjacent in the Host (under a one-to-one strategy) can be mapped to the same Host vertex, resulting in reduced path lengths (dilation) for any Host paths using that original edge. Hence, the overall dilation and/or congestion of the embedding may be reduced. For example, in Figure 1.6 we illustrate a two-to-one embedding from/to the same Guest/Host of Figure 1.5, which has improved the dilation from 2, to 1. The technique used is referred to as a **fold**, since the Grid was geometrically folded to reduce the width of the Guest to a point where it could fit into the Host.

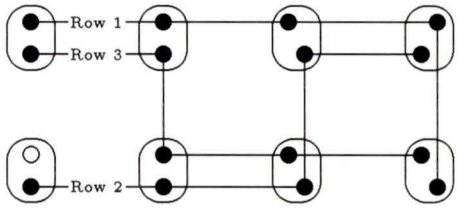
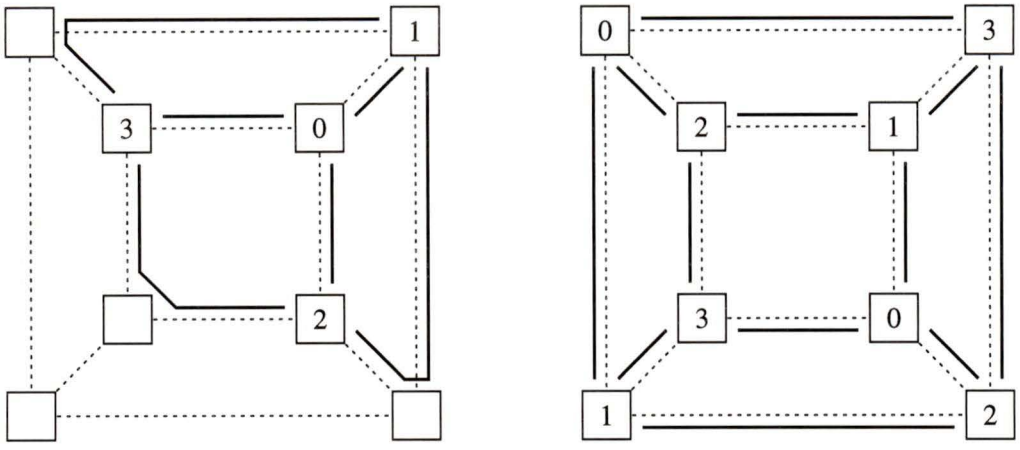


Figure 1.7: 3×5 to 2×4

We observe that by mapping two-to-one into $G(4,4)$, which is a subgraph of $Q(4)$, we use less than $1/2$ of the available Host. Therefore, we should try for a two-to-one embedding of $G(3,5)$ into $G(2,4)$, which is a subgraph of $Q(3)$, and where in this case, $Q(3)$ would be the next smaller Hypercube Host under a load 2 embedding. Figure 1.7 illustrates a dilation 1 solution to this problem.

1.4.3 Many to One Embeddings

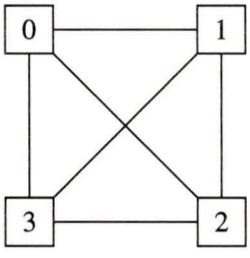
Many-to-one embeddings are an extension of two to one embeddings (i.e. we wish to solve for the smallest load factor given a Host of size smaller than the size of the Guest). These embeddings would have applications for a fixed sized Host — unlimited size Guest.



dilation 2 (1:1 embedding) dilation 1 (1:2 embedding)

Figure 1.8: K_4 Embedded in $Q(3)$

1.4.4 One to Many Embeddings



K_4

This area of research has only recently been been explored [16, 11]. The problem, referred to as **Redundancy**, looks at embeddings where a Guest vertex can be mapped to many Host vertices. The motivation is to find load factor 1 embeddings where the duplication of Guest vertices in the Host allows an improvement over the best known dilation of any other load factor 1 embedding. The major disadvantage of this embedding scheme is that it requires large Hosts. In Figure 1.8 we illustrate an example (by Fellows [11]), where K_4 is one-to-one embedded with at best dilation 2 into Q(3), but is also one-to-two embedded with dilation 1 into $Q(3)$.

1.5 Our Problem

We have previously mentioned that Grid networks are a natural structure for some parallel algorithms. However, they are inefficient for others. Also, the Hypercube with the following properties

- Rich in symmetry.
- Many edges.
- Small diameter.
- Vertex degree which grows slowly (logarithmically).

has proven to be able to efficiently simulate a vast number of parallel structures and to be realizable in hardware [15, 13]. Finding hardware based Hypercube structures is one reason why some of the research on Grids has pursued optimizing the costs of Grid to Hypercube embeddings [5, 8, 6, 7].

We note that load 1, dilation 1, Grid to Hypercube embeddings are not always possible [5]. We also know that in the instances where dilation 1 is not attainable, we can achieve dilation 2 [8]. Consequently, we are motivated by the potential of reducing the best achievable dilation 2 instances of a load 1 Grid to Hypercube embedding, to optimal dilation 1, under a load 2 embedding.

Therefore, we pursue the problem of finding dilation 1, load 2 embeddings of Grids into the next smaller Hypercube.

We derive ways of solving dilation 1, load 2 embeddings for several classes of Grids that under a load 1 embedding, would not always be embeddable with dilation 1. As justified in 1.4.2, this is an improvement over a load 1, dilation 2 embedding.

1.5.1 General Assumptions

When there is no ambiguity and unless otherwise stated, we use the following assumptions:

- Our Guest graph is a Grid and a our Host graph is a Hypercube.
- The Host Hypercube will be the smallest Hypercube for the appropriate load factor.

Chapter 2

Previous Results

2.1 Grids to Grids

Research on embedding Grids into Grids has generally been in the context of one-to-one embeddings into square Grids (the motivation being that a square Grid may be the model of Grid based, parallel computer architectures). Initially, Aleliunas and Rosenberg [1] studied embeddings of Grids with aspect ratio greater than one, into the smallest square Grid. In some cases they could not guarantee dilation less than 18 (the larger the aspect ratio, the harder the problem appeared), and they speculated that there was an enherent tradeoff between dilation and expansion. There has since been a tightening of the dilation and expansion bounds for small and large aspect ratios, showing that dilation and expansion can both be almost optimized simultaneously [10, 9].

Some recent results show that:

- If the compression ratio is ≤ 2, then a Grid can be embedded with dilation 2 into the smallest Grid [10].

- If the compression ratio is sufficiently large, then a Grid can be embedded with dilation 2 into the smallest Grid [9].
- If the expansion is less than 1.2, then a Grid can be embedded with dilation 2 into the square grid [18].

However, the problem of embedding a Guest Grid into a smallest Host Grid of a given width, is in general unsolved.

Of relevance to our work is a theorem by Ellis [10], that provides a constructive approach to finding optimal dilation 2 embeddings of Guest Grids into some Host Grids.

Theorem 1 [10] *If the compression ratio is* ≤ 2*, then a rectangular Grid can be embedded into any of its smallest rectangular Grids with dilation 2.*

Without loss of generality, Theorem 1 assumes that $h \leq w$. Then we have that a Grid $G(h, w)$ can be embedded into any Host Grid $G(h', w')$ with dilation 2, if given w', $w/w' \leq 2$ and h' is the smallest such that $hw \leq h'w'$.

2.2 Grids to Hypercubes

2.2.1 One-to-One Embeddings

Theorem 2 [14, 8]

$$G(h, w) \text{ is a subgraph of } Q(d) \Leftrightarrow d \geq \lceil \log_2 h \rceil + \lceil \log_2 w \rceil$$

Proof: We first note that there exists a subgraph iff there exists a dilation 1 embedding.

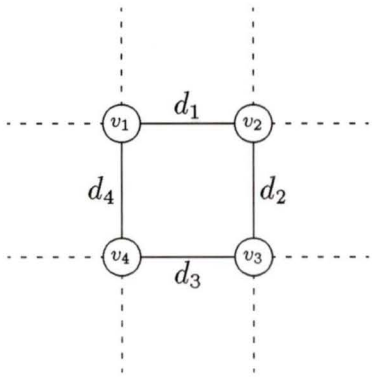


Figure 2.1: Embedded 4-cycle $(v_1, v_2, v_3, v_4, v_1)$

In Figure 2.1 we show an embedded 4-cycle of a Grid into a Hypercube cycle. The values d_1, d_2, d_3, d_4 denote the dimension that the illustrated edges cross in the Hypercube.

We note that since the embedding is 1-to-1, vertices in the Grid are mapped to distinct vertices in the Hypercube. Hence, since an edge in the Hypercube crosses precisely one dimension, we have that 2 distinct edges incident to the same vertex must cross different dimensions. Therefore in our 4-cycle, we have:

$$\begin{aligned} d_1 &\neq d_2 \\ d_2 &\neq d_3 \\ d_3 &\neq d_4 \\ d_4 &\neq d_1 \end{aligned}$$

If we traverse the 4-cycle once, starting and ending at vertex v_i we must cross like dimensions an even number of times to return to the initial vertex. Therefore, by the

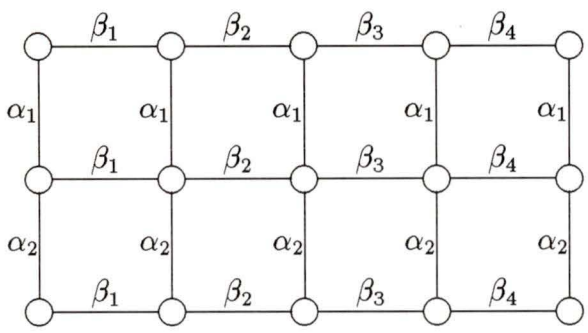


Figure 2.2: $G(3,5)$ Edge Dimensions Crossed

---

previous inequalities we have:

$$d_1 = d_3$$
$$d_2 = d_4$$

(i.e. parallel edges of the 4-cycle cross the same dimension).

In Figure 2.2, we show the only way that $G(3,5)$ can be embedded into the Hypercube. The α_i's denote the dimension that the illustrated vertical edges cross. Likewise the β_j's denote the dimension that the horizontal edges cross. By applying the previous result repeatedly on adjacent 4-cycles, we have that all parallel edges in the same row or column (vertical/horizontal edges crossing between the same rows/columns) of a Grid embedded in the Hypercube, must cross the same dimension. In Figure 2.2, we have subscripted the α_i's and β_j's appropriately to illustrate this property.

Suppose a horizontal and vertical edge cross the same dimension — then because all the edges in the same row/column as the vertical/horizontal edge must cross the same dimension, there exists a horizontal and vertical edge incident to the same vertex and crossing the same dimension $\Rightarrow\Leftarrow$.

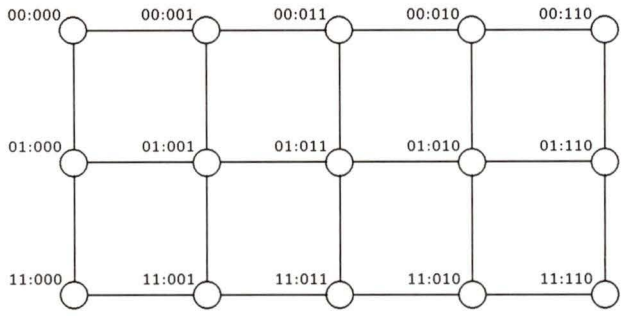


Figure 2.3: $G(3,5)$ Map Labeling

Therefore, the set of dimensions crossed by horizontal edges is disjoint from the set of dimensions crossed by vertical edges.

Since each dimension corresponds to a single bit in the label of the Hypercube vertices, and since there are w distinct vertices in a row, we require

$$d_w = \lceil \log_2 w \rceil \tag{2.1}$$

bits in the vertex labels to distinguish between the vertices in any row, and likewise since there are h distinct vertices in a column, we require

$$d_h = \lceil \log_2 h \rceil \tag{2.2}$$

bits in the vertex labels to distinguish between vertices in any column.

Since these dimension sets are disjoint:

$$d < \lceil \log_2 h \rceil + \lceil \log_2 w \rceil \Rightarrow Q(d) \text{ is not a subgraph of } G(h,w)$$

To show that

$$d \geq \lceil \log_2 h \rceil + \lceil \log_2 w \rceil \Rightarrow Q(d) \text{ is a subgraph of } G(h,w)$$

we label the vertices of $G(h, w)$ with a $d_h + d_w$ length binary string. Assign the first d_h bits of the vertex label for $G[i, j]$ with the ith string from a Gray code sequence of d_h bits, and assign the next d_w bits with the jth string from a Gray code sequence of d_w bits. For example, we have labeled the vertices in Figure 2.3 appropriately, where the first 2 bits indicate the row and the next 3 bits indicate the column. If we use these labels as the Hypercube labels we map to, then it is clear that adjacent vertices in the Grid are adjacent in the Hypercube, since their string labels differ in only one bit. □

Note that dilation 1 is not always possible, since the smallest Hypercube is of dimension

$$d = \lceil \log_2 hw \rceil = \lceil \log_2 h + \log_2 w \rceil$$

which may be less than the required dimension for dilation 1

$$\lceil \log_2 h \rceil + \lceil \log_2 w \rceil$$

Chan [5] has shown that in all other cases, dilation 2 is possible. There is one other adhoc proof developed independently by Sudborough [2]. We illustrate another proof being the direct result of Ellis [10].

Theorem 3 *$G(h, w)$ is dilation 2 embeddable into the smallest Hypercube.*

Proof: We wish to find a dilation 2 embedding of $G(h, w)$ into $Q(d)$, where $Q(d)$ is smallest Hypercube with respect to G and we assume without loss of generality that $h \leq w$. Note that neither h nor w is a power of 2. Otherwise $G(h, w)$ is dilation 1 embeddable into the smallest Hypercube.

Let w' be the largest power of 2 less than w and h' be the smallest integer such that $hw \leq h'w'$. Therefore we have the following:

$$w' = 2^{\lfloor \log_2 w \rfloor} \tag{2.3}$$

$$(h'-1)w' < hw \tag{2.4}$$

Note that $G(h', w')$ is a subgraph of $Q(d')$, where $d' = \lceil \log_2 h'w' \rceil$, because

$$\begin{aligned} \lceil \log_2 h'w' \rceil &= \lceil \log_2 h' + \log_2 w' \rceil \\ &= \lceil \log_2 h' \rceil + \lceil \log_2 w' \rceil \end{aligned}$$

By Theorem 1, since $w/w' < 2$, $G(h, w)$ is dilation 2 embeddable into $G(h', w')$. Therefore,

$$G(h, w) \text{ is dilation 2 embeddable into } Q(d') \tag{2.5}$$

Since $hw \leq h'w'$, we have:

$$d \leq d' \tag{2.6}$$

We can write:

$$\begin{aligned} w &= 2^{i+\alpha}, \quad 0 < \alpha < 1, \quad i \text{ an integer} \\ h &= 2^{j+\beta}, \quad 0 < \beta < 1, \quad j \text{ an integer} \end{aligned}$$

Therefore

$$\begin{aligned} w' &= 2^i, && \text{(by Equation 2.3)} \\ h' &< h\left(\tfrac{w}{w'}\right) + 1, && \text{(by Equation 2.4)} \\ &< 2^{j+\alpha+\beta} + 1 \\ &< 2^{\lceil j+\alpha+\beta \rceil} + 1 \\ &\leq 2^{\lceil j+\alpha+\beta \rceil}, && \text{(since } h' \text{ and } 2^{\lceil \cdots \rceil} \text{ are integers)} \\ &\leq 2^{j+\lceil \alpha+\beta \rceil} \end{aligned}$$

Therefore

$$\begin{aligned} h'w' &\leq 2^{i+j+\lceil \alpha+\beta \rceil} \\ &\leq 2^{\lceil i+j+\alpha+\beta \rceil} \\ &\leq 2^{\lceil \log_2 hw \rceil} \end{aligned}$$

Hence

$$\begin{aligned} d' &= \lceil \log_2 h'w' \rceil \\ &\leq \lceil \log_2 hw \rceil \\ &\leq d \end{aligned} \tag{2.7}$$

Therefore, by Eqn 2.5, 2.6 and 2.7, $G(h, w)$ is dilation 2 embeddable into $Q(d)$. □

2.2.2 Two-to-One Embeddings

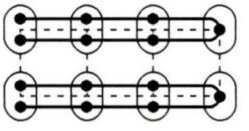


Figure 2.4:
2×7 to 2×4

When a one-to-one, dilation 1 embedding into the smallest Hypercube exists, a solution for a two-to-one, dilation 1 embedding into the smallest Hypercube can be found by a simple fold about a vertical or horizontal line of the embedding (see Figure 2.4, where we fold in half a 2×7 Grid embedding into a map of $Q(4)$). The fold does not change the dilation of the embedding (except perhaps edge dilation about the fold, where an even width implies that adjacent vertices immediately about the fold boundary are mapped to the same vertex), but cuts in half the number of vertices required by the Host.

We know of no fold technique that will convert a one-to-one, dilation 2 embedding into a two-to-one, dilation 1 embedding (i.e. where the fold removes the dilation 2 components). The technique does however illustrate that we can always find a two-to-one, dilation 2 embedding from a one-to-one, dilation 2 embedding.

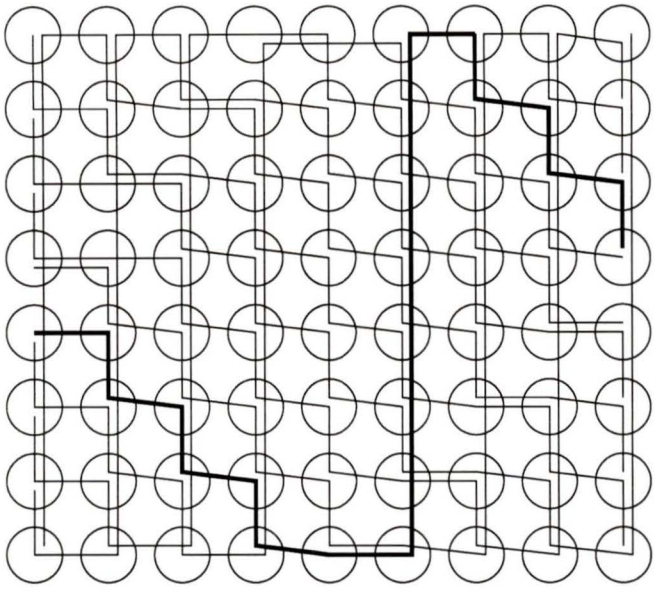


Figure 2.5: Sudborough & Miller Height 9 Braiding

2.2.3 Many-to-One Embeddings

Sudborough and Miller [19] have studied a variety of load factors for Grids to Hypercube embeddings. Their technique relies upon finding edge patterns (called "braidings") of a given height that use in a greedy way (left to right, first come first serve) all vertex labels in each column of a Hypercube map as one follows the braiding from left to right. For a height $h = 2^j + k$, a braiding must map all vertices of the Grid $G(2^j + k, 2^{j+1})$ with a pattern of height 2^j and width $2^j + k$ (i.e. uses the first $2^j + k$ columns of a height 2^j Hypercube map). Since a braiding is a mapping pattern that uses the Hypercube map columns in a greedy way, the pattern can be replicated (by reflection) any number of times to provide solutions for all Grids of that height. For example, Figure 2.5 illustrates their two-to-one, dilation 1 braiding for a height 9 Grid. We have highlighted one edge for illustration purposes. In this case, the braid-

ing has an apparent pattern which does not make use of Hypercube edges, but we note that braidings are usually much more complex.

Although braidings for Grids of every height have not been found, numerous braidings do exist, since Sudborough and Miller also derived techniques that merge braidings of different height to create new braidings of composite height, but also larger dilation. Their results have improved upon the dilation of existing techniques, but do not guarantee optimal results. They can guarantee a dilation 1 embedding into the next smaller Hypercube with load factor 3.

Chapter 3

Various Techniques

3.1 Subgraph Searching

Solving for two-to-one, dilation 1 embeddings of Grids into the next smaller Hypercube, might also be solved by extending the Hypercube (inserting additional vertices and edges) such that the problem is transformed into searching for a subgraph (i.e. a one-to-one, dilation 1 embedding). This approach requires transforming each Hypercube vertex into two adjacent vertices (replace vertices by K_2) and inserting edges between each new vertex and the vertices adjacent to the original vertex (see Figure 3.1, where the transformation of $Q(2)$ is illustrated). We could also view the transformation as replacing each edge by K_4. Unfortunately, this transformation causes the problem size to become larger and since it does not seem to yield an easier way of solving our problem, we did not pursue this approach extensively.

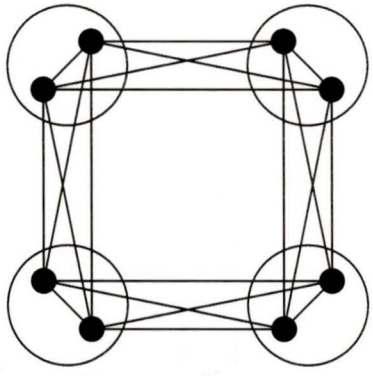


Figure 3.1: $Q(2)$ Supergraph Transformation

3.2 Computer Searching

Also available is a technique in which we examine computer generated embeddings of a small embedding problem, with the intent to observe patterns leading to an algorithm for solving larger instances of the problem. This method has successfully been used in solving Grid to Grid embedding problems [10, 9] and it may be that our problem of embedding into the Hypercube is also amenable to such an approach.

Our embedding program takes as input, the dimensions of the Guest Grid and the maximum number of embeddings to search for. It exhaustively searches for valid embeddings and produces the following output:

- A unique file with the extension ".CGO" for each embedding that is found. The CGO (Common Graph Output) file conforms to a standard format used by a graph editing/viewing/testing utility called GED. The CGO format specifies a graph as a set of adjacency lists (for each vertex, a list of the vertices adjacent to it). Our CGO graphs are adjacency lists specifying the horizontal row edges of the Guest Grid as they appear embedded in the Host Hypercube (i.e. an edge map of the embedding which follows the convention of illustrating only the Hypercube edges that are used by the horizontal edges of the Grid).

- Standard screen output, which in addition to general processing messages, for each solution found produces two unique matrix representations (to be described later) of the computed embedding, a message indicating the name of the CGO output file, and the CPU time used to compute the solution.

In order for this technique to promote the recognition of patterns, it requires that the set of solutions be small. We found however that since the Hypercube is so

rich in symmetry and number of edges, for even small instances of the problem our exhaustive search algorithm had the detrimental effect of computing:

- large numbers of nearly identical embeddings.
- complex embedding structures when we knew simple ones existed.

Due to these complexities, we modified the approach by adding the ability to constrain the structure of solutions (by specifying where a vertex of the Guest may or may not be mapped to in the Host), resulting in a more manageable human goal of recognizing patterns within a solution. However, an inherent drawback to constraining solutions is that we must foresee which partial embedding patterns to selectively investigate while taking care not to overlook a decisive pattern.

There are two ways embeddings are presented so their structure can be analyzed for a pattern based solution:

- Edge Mappings
- Vertex Mappings

3.2.1 Pattern Search of Edge Mappings

With this technique we draw upon the original method used by Ellis [10, 9], which searches for a patterning within the edge mapping of sample embeddings (the way Guest edges are embedded in the Host). Our algorithm computes these embeddings and stores the results as edge mappings in CGO files. Any potential patterns that exist in the map are then explored by viewing the embeddings via GED. For example, in Figure 3.2 we use GED to illustrate a two-to-one, dilation 1 computer generated embedding of $G(3,5)$ into $G(2,4)$. Unfortunately, edge mappings of Grid to Hypercube embeddings are usually complex in nature because of the larger number of edges in the Hypercube and therefore, picking out patterns from the edge mappings of Grid to Hypercube embeddings, usually presents an unmanageable task. For example, Figure 3.3 (a) illustrates a typical solution when embedding $G(7,9)$ into the next smaller Hypercube. Despite this problem, we did experiment further by constraining solutions to more simple geometric forms, but were unable to find recognizable patterns. Indeed as illustrated in Figure 3.3 (b), the simple constraint of forcing Grid corner vertices to corner vertices of the illustrated Host created a more complex embedding structure than Figure 3.3 (a).

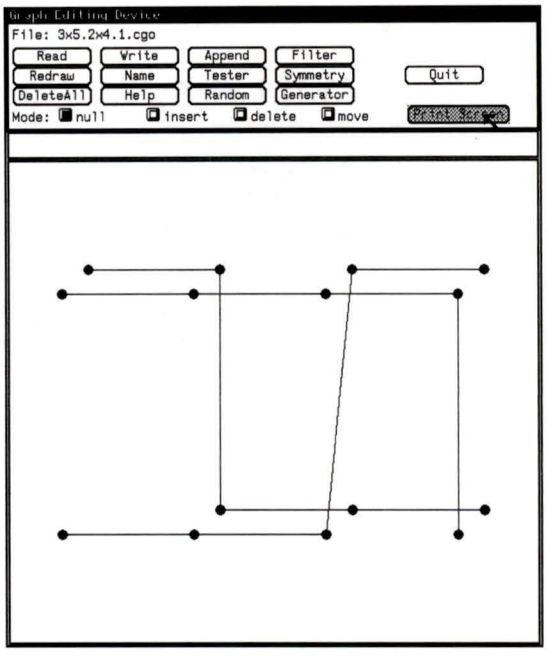
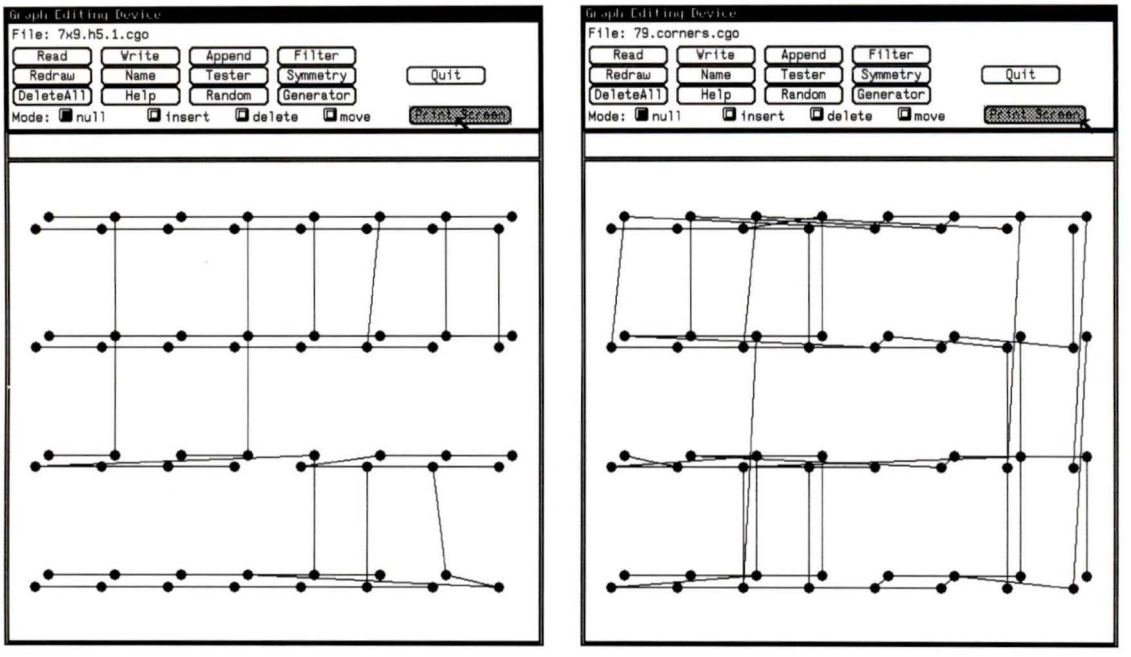


Figure 3.2: 3×5 into 2×4 Computer Generated Embedding



(a) (b)

Figure 3.3: 7×9 Sample Solutions

Since our embeddings do not appear to produce recognizable structures, we have so far shown no example where patterns can be found via this technique. Therefore, although not applicable to our problem, in Figure 3.4 we illustrate via a computer generated embedding of $G(6,8)$ into $G(5,5)$, that using this technique can produce recognizable patterns. In this instance, the two middle rows exhibit an identical pattern, while the top two rows appear identical to the bottom two rows under reflection.

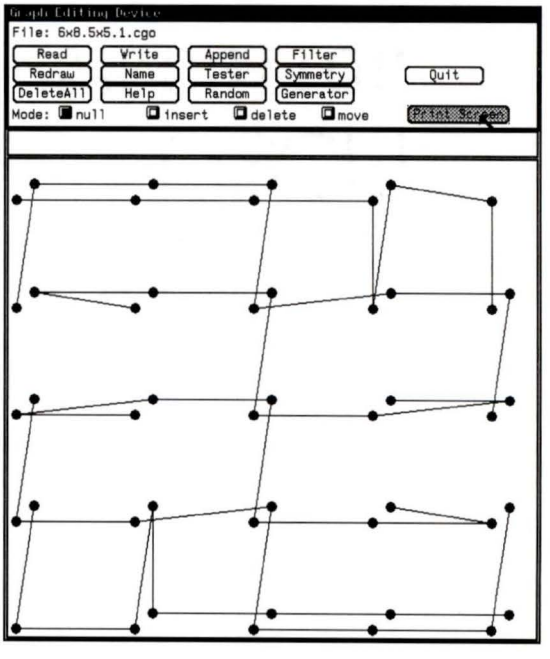


Figure 3.4: 6×8 into 5×5 Computer Generated Embedding

```
Solution 1:
   00 01 03 02 06 04 05 07 15
   08 09 11 10 14 12 13 05 07
   24 25 27 26 30 28 29 21 23
   26 27 31 30 22 20 21 23 22
   24 25 29 28 20 16 17 19 18
   08 09 13 12 04 00 01 17 16
   10 11 15 14 06 02 03 19 18

   00:50  01:51  03:59  02:60  05:49  06:16  04:58  07:17
   09:45  10:46  12:54  11:55  14:48  15:47  13:57  08:56
   41:53  42:52  44:62  43:61  32:40  25:33  31:35  26:34
   18:36  19:37  21:27  20:28  23:39  24:38  22:30  29:

output to CGO file:7x9.h5.1.cgo  time=12.820 seconds
```

Figure 3.5: Sample Computer Output of $G(7,9)$ Embeddings

3.2.2 Pattern Search of Vertex Mappings

In our effort to recognize patterns in the complex edge mappings created by our embedding problem (see Section 3.2.1), we realized that patterns in the structure of a vertex map representation of our embeddings might be easier to recognize. With a vertex map representation, the patterns we are looking for are no longer geometrical patterns, but relationships between the values of elements or blocks of elements in the map. Although this is a more difficult task than recognizing patterns of a more geometrical form (edge mappings), it has the distinct advantage that the complexity of the embedding bears no relationship to the complexity of the mapping (the mapping is just a matrix of integers).

Figure 3.5 shows the relevant portion of our computer programs screen output. The two matrices represent in unique ways the equivalent embedding of $G(7,9)$ into $Q(5)$ as illustrated in Figure 3.3 (a). The first matrix is a vertex map of the Guest

Grid which shows the label of the Host Hypercube vertex that each Guest vertex is mapped to and the second matrix is a representation of the Hypercube where elements are two integers (separated by a ":") which specify the label of the Grid vertices that are mapped to the Hypercube vertex in question (an inverse mapping). In our matrix representation of the vertex map and inverse image map, the label our algorithm assigns to the vertex associated with a position in the matrix, is a row major ordering (left to right, top to bottom) using the integers from 0 to the size of the graph.

3.3 Permuting Vertex Mappings

Although in Section 3.2.2 we found no clear patterns in our vertex mappings, a few ideas were drawn from their analysis:

- Vertex Numberings are Gray codes (obvious from the definition of a Hypercube).
- Duplicate vertex mappings frequently occur along a diagonal.
- We can often observe localized patterns. For example, we sometimes see blocks of elements shifted along diagonals.

Because of the order in which the algorithm computes solutions (via a permutation generator), we assume that the localized patterns are local because we only examined the first few solutions and were we able to examine all the solutions, we would likely find these patterns on a larger scale. Therefore, we pursued the idea that it may be possible to re-arrange the elements of a pre-initialized (simple mapping into the next smaller Hypercube) vertex map, using operations that maintain a Gray code invariant and produce larger scale patterns similar to our localized patterns. In the process of applying these operations, if we can reshape the host vertex map into a map having

the same dimensions as the Guest Grid, we will have a recipe for computing a vertex map specification for a valid embedding. We note that there is a considerable number of ways we can manipulate the vertex map, since the map represents a graph with many edges (the Hypercube) and at some stage(s) in the rearrangement elements of the Hypercube map may be doubled to account for the two-to-one nature of the embedding.

This technique was used to derive all our major results. The details are developed in Chapter 4, where a pre-initialized vertex map we often make use of has elements that are assigned numberings from a reflective Gray code sequence in the fashion illustrated by Figure 3.6, where $Q(4)$ is represented in this way.

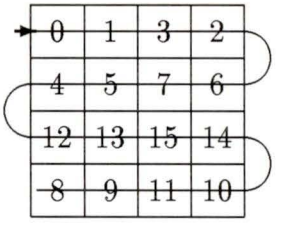


Figure 3.6: 4×4 Hypercube Vertex Mapping

3.4 Subgraphs of Hypercubes

We are searching for two-to-one, dilation 1 embeddings into the next smaller Hypercube. In this respect, we are not aware of a result indicating that Hypercube Hosts are sufficient. Neither is there a result indicating that Grid Hosts are insufficient. It may be that a graph family both necessary and sufficient to accept any Grid is somewhere between a Grid and a Hypercube in complexity.

3.4.1 Grid to Grid

By Theorem 3, dilation 2 is the best we can hope for under a one-to-one Grid to Hypercube embedding. As illustrated in the proof of Theorem 3, of particular importance was the compression of the Guest Grid into a Host Grid with dimensions such

that the Host is a subgraph of the next smaller Hypercube by Theorem 1 (width of the compressed Grid is a power of 2). Because we know little about two-to-one embeddings of Grids into Grids with similar compression characteristics, and since the additional flexibility of a two-to-one embedding may allow dilation 1 by contraction of the worst case length 2 paths, we thought it possible that an analogous dilation 1 result of Theorem 1 could exist for two-to-one embeddings, thereby providing us with the ammunition to find a two-to-one, dilation 1 embedding into the next smaller Hypercube via an analogue to the proof of Theorem 3. To further enforce this conjecture we refer the reader back to Figure 1.7, which illustrates a circumstance where indeed, a Guest "$G(3,5)$" which could not be one-to-one, dilation 1 embedded into $G(4,4)$ and hence the smallest Hypercube $Q(4)$, could be two-to-one, dilation 1 embedded into the Host $G(2,4)$ and hence the next smaller Hypercube $Q(3)$.

3.4.2 Results

3.4.2.1 Grid Hosts

We briefly explored the possibility of two-to-one, dilation 1 embedding Grids into smaller Grids. Our approach was by attempting to find patterns in computer generated solutions. Initially, computer searching generated a wealth of solutions for all the instances of the problem that we tried. However, since searching takes factorial time, the problem sizes we looked at were small.

For a given Hypercube dimension and an associated Host Grid ($G(2^{\lfloor d/2 \rfloor}, 2^{\lceil d/2 \rceil})$, for dimension d), we searched for solutions where the Guest Grids were of maximum height/width and by Theorem 3, whose best one-to-one embedding was dilation 2. The embeddings of $G(3,5)$ into $G(2,4)$ and $G(7,9)$ into $G(4,8)$ are two such cases. In the case of $G(7,9)$ being embedded into $G(4,8)$, computer searching found no

solution. We have no other proof, but due to our success in searches with other graphs, we are reasonably certain of our program's correctness. We are therefore confident that no embedding exists.

Since we observed the existence of two-to-one, dilation 1 embeddings into a smallest Host Grid, where no corresponding one-to-one, dilation 1 embedding existed, of interest would be a characterization of when an improvement in dilation is possible. Currently no such characterizations are known.

Chapter 4

Square Grids

4.1 Introduction

In this section we solve the problem of two-to-one embedding a square grid into the next smaller Hypercube. The problem is solved by two-to-one embedding the largest possible square grid into a Hypercube of any given dimension. This is sufficient to solve for all square grids, since any square grid can be embedded via a subgraph of the largest possible square grid embedding. Definitions and terminology required are given below.

4.2 Definitions

Definition 1 *A* **Map** *M is a 2-dimensional matrix whose elements are integers or null and where non-null adjacent elements of the map are within Hamming distance 1 of each other.*

- The **height of M**, denoted by $\boldsymbol{h}(\boldsymbol{M})$ is the number of rows in M.
- The **width of M**, denoted by $\boldsymbol{w}(\boldsymbol{M})$ is the number of columns in M.
- $\boldsymbol{M}$**[row, column]** denotes an individual element of the map M, where the left-most upper element is denoted by $M[1,1]$.
- **cols**$(\boldsymbol{M}, \boldsymbol{i}, \boldsymbol{j})$ is used to denote the map composed of column i through column j (inclusive) of the map M.
- **rows**$(\boldsymbol{M}, \boldsymbol{i}, \boldsymbol{j})$ is used to denote the map composed of row i through row j (inclusive) of the map M.

Definition 2 *The* **Vertical Separator** *of a Map M, denoted by $\boldsymbol{S}_v(\boldsymbol{M})$, is the median column index and is used to partition M based on column index, into left and right parts, denoted by* **left**$(\boldsymbol{M})$ *and* **right**$(\boldsymbol{M})$.

Therefore we define:

- $S_v(M) = (w(M)+1)/2$, since columns are in the range $[1, \ldots, w(M)]$.
- $\text{left}(M) = \text{cols}(M, 1, \lfloor S_v(M) \rfloor)$
- $\text{right}(M) = \begin{cases} \text{null} & w(M) = 1 \\ \text{cols}(M, \lfloor S_v(M) \rfloor + 1, w(M)) & w(M) > 1 \end{cases}$

Definition 3 *The* **Horizontal Separator** *of a Map M, denoted by $\boldsymbol{S}_h(\boldsymbol{M})$, is the median row index and is used to partition M based on row index, into top and bottom parts, denoted by* **top**$(\boldsymbol{M})$ *and* **bottom**$(\boldsymbol{M})$.

Therefore we define:

- $S_h(M) = (h(M) + 1)/2$, since rows are in the range $[1, \ldots, h(M)]$.
- $\text{top}(M) = \text{rows}(M, 1, \lfloor S_h(M) \rfloor)$
- $\text{bottom}(M) = \begin{cases} \text{null} & h(M) = 1 \\ \text{rows}(M, \lfloor S_h(M) \rfloor + 1, h(M)) & h(M) > 1 \end{cases}$

We note that in many cases, but not all, map dimensions will be powers of 2. When $h(M)$ is a power of 2, then $h(M)$ is either even or 1. In the case that $h(M)$ is even, $h(\text{top}(M)) = h(\text{bottom}(M)) = h(M)/2$. Otherwise $h(M) = 1$ implying $\text{top}(M) = M$ and $\text{bottom}(M)$ is null. Likewise when $w(M)$ is a power of 2, a similar result will hold for $\text{left}(M)$ and $\text{right}(M)$.

Definition 4 *A Map M is said to have* **Reflective Edges about the Horizontal Separator** *if M has the property that elements in the same column with a row index equidistant from the Horizontal Separator are Hamming distance 1 apart.*

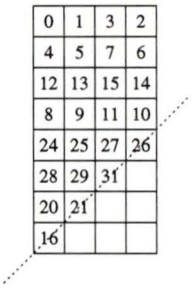
Definition 5 *A Map M is said to have* **Reflective Edges about the Vertical Separator** *if M has the property that elements in the same row with a column index equidistant from the Vertical Separator are Hamming distance 1 apart.*

Definition 6 *A* **Trapezoid** *T is a map with the following properties:*

1. $w(T) \leq h(T)$
2. *The elements $T[i, j]$, for $1 \leq i < h(T) + 1 - j$ and $1 \leq j \leq w(T)$ are non-null and unique.*
3. *The elements $T[i, j]$, for $i = h(T) + 1 - j$ and $1 \leq j \leq w(T)$ are non-null and unique from the elements of property 2.*

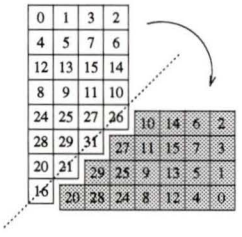
4. The elements $T[i,j]$, for $h(T)+1-j < i \leq h(T)$ and $1 \leq j \leq w(T)$ are null.

This definition describes a map where the non-null elements take the form of a trapezoid and are unique, with the exception that elements along the diagonal may be duplicates of other elements along the diagonal.



Trapezoid

Operation 1 *A* **Diagonal Fold** *of a trapezoid T, denoted by $\boldsymbol{F_d(T)}$, duplicates the elements from above the diagonal of T by reflecting T in a plane through the diagonal.*



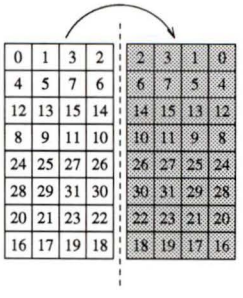
Diagonal Fold

Formally:

- $T[h(T)+1-j, h(T)+1-i] \leftarrow T[i,j]$, for $1 \leq i < h(T)+1-j$ and $1 \leq j \leq w(T)$
- $w(T) \leftarrow h(T)$

We note that off diagonal elements are the only elements duplicated by $F_d(T)$.

Operation 2 *A* **Vertical Fold** *of a map M, denoted by $\boldsymbol{F_v(M)}$, duplicates the elements of M by reflecting M in a plane through a vertical axis to the right of M.*

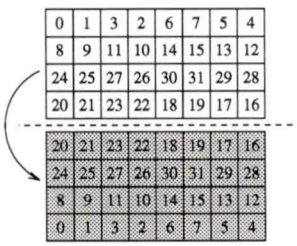


Vertical Fold

Formally:

- $T[i, 2w(T) + 1 - j] \leftarrow T[i, j]$, for $1 \leq i \leq h(T)$ and $1 \leq j \leq w(T)$
- $w(T) \leftarrow 2w(T)$

Operation 3 *A* **Horizontal Fold** *of a map M, denoted by $\boldsymbol{F_h(M)}$, duplicates the elements of M by reflecting M in a plane through a horizontal axis below M.*



Horizontal Fold

Formally:

- $T[2h(T) + 1 - i, j] \leftarrow T[i, j]$, for $1 \leq i \leq h(T)$ and $1 \leq j \leq w(T)$
- $h(T) \leftarrow 2h(T)$

Definition 7 *A* **Hypercube Map** *H of dimension d, denoted by* $\mathbf{HM}(\boldsymbol{d})$ *is a map where $h(H) = 2^{k_1}$ and $w(H) = 2^{k_2}$ for some $k_1 + k_2 = d$ and where elements are unique and in the range $[0, \ldots, 2^d - 1]$. A Hypercube Map may also be denoted by* $\mathbf{HM}(\boldsymbol{h}, \boldsymbol{w})$*, where the height h and width w are specified directly, but must be powers of 2, and in which case the dimension $d = \log_2 hw$.*

In a Hypercube Map, $H = \mathrm{HM}(d) = \mathrm{HM}(h, w)$, $|Q(d)| = 2^d$ and elements are unique and in the range $[0, \ldots, 2^d - 1]$. Therefore, elements of the map can be used to represent the vertices of a Hypercube. In this analogy, many pairs of elements which are not adjacent in the map, are adjacent in the corresponding Hypercube (within Hamming distance 1 of each other). This observation allows for manipulations to the map that may change the dimensions and uniqueness of elements, but preserve the property that adjacent elements in the resulting map are within Hamming distance 1 and hence, are adjacent in the corresponding Hypercube. Using this idea, grid

mappings into Hypercubes can be realized in a natural way by associating a reshaped Hypercube Map H' of H, with height h' and width w', as a mapping of the $h' \times w'$ grid vertices into the Hypercube of dimension d.

Definition 8 *A* **Partial Hypercube Map** H *of dimension* d*, is a map with unique elements in the range* $[0, \ldots, 2^d - 1]$*, and where* H *has the following properties:*

1. *If* $w(H) > 1$ *then*
   $right(H)$ *adj* $F_h(left(H))$
2. *If* $h(H) > 1$ *then*
   $top(H)$ *is a Partial Hypercube Map.*
   $bottom(H)$ *adj* $F_v(top(H))$

Some observations we can make are:

- $w(H)$ is even or 1, by property 1.
- $h(H)$ is a power of 2, by property 2.

Definition 9 *A* **Reflective Hypercube Map** H*, denoted by* $\boldsymbol{H} = \mathbf{HM}_r(\boldsymbol{d})$*, or by* $\boldsymbol{H} = \mathbf{HM}_r(\boldsymbol{h}, \boldsymbol{w})$*, is a specific Hypercube Map of dimension* d*, defined recursively by the following:*

1. $HM_r(0) = HM_r(1,1) = [0]$
2. *If* $h = 1$ *and* $w \geq 2$ *then*
   $right(H) = F_v(left(H)) + |H|/2$

3. *If* $h > 1$ *then*

$$bottom(H) = F_h(top(H)) + |H|/2$$
$$top(H) = HM_r(h/2, w)$$

For example,

$$\mathrm{HM}_r(1) = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathrm{HM}_r(2) = \begin{bmatrix} 0 & 1 & 3 & 2 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

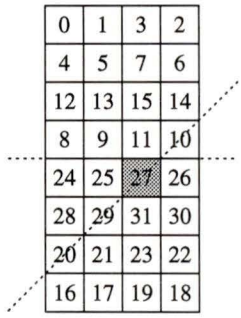
$$\mathrm{HM}_r(2,4) = \begin{bmatrix} 0 & 1 & 3 & 2 \\ 4 & 5 & 7 & 6 \end{bmatrix}$$

Some observations we can make are:

- $\mathrm{HM}_r(d)$ is a Partial Hypercube Map of dimension d.
- By property 2, a Reflective Hypercube Map of height 1 is a Binary Reflected Gray Code, and hence the definition forces reflective edges about the Vertical Separator of any Reflective Hypercube Map.
- Similarly, by property 3, a Reflective Hypercube Map of height greater than 1, forces reflective edges about the Horizontal Separator.
- Since the definition is recursive, Reflective edges will exist about the Horizontal and Vertical Separators of all equal sub-divisions of H.

Definition 10 *The* **Diagonal Division Column** *of a Partial Hypercube Map H, for $i > S_h(H) > 1$, is denoted by* $\mathbf{ddc}(\boldsymbol{H}, \boldsymbol{i}) = i - \lfloor S_h(H) \rfloor$.

Visually, $\mathrm{ddc}(H, i)$ is the column index of H where the row index of H points to row 1 in $\mathrm{bottom}(H)$ when advancing diagonally up and right from $H[i, 1]$. That is, progressing from $H[i, 1]$, to $H[i-1, 2]$, ..., to $H[\lfloor S_h(H) \rfloor + 1, i - \lfloor S_h(H) \rfloor]$.



$\mathrm{ddc}(H, 7) = 3$

Rotations are operations used to cut and paste blocks of elements (visualized as rotations of the blocks) of various Maps, and create new types of Maps as a result. There are three rotation operators defined below and illustrated in Figure 4.1 through Figure 4.3.

Operation 4 $\mathbf{R_{ccw}(M)}$ *cuts and pastes the map M by rotating $\mathrm{bottom}(M)$ counter clockwise 180 degrees and appending the result to the right of $\mathrm{top}(M)$, where $M = \mathrm{cols}(H, i, j)$, for $1 \leq i \leq j \leq w(H)$ and H a Partial Hypercube Map of height > 1.*

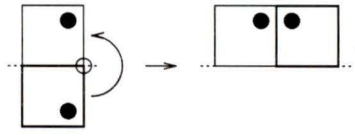


Figure 4.1: $R_{ccw}(M)$

Formally, we can define:

$$R_{ccw}(M) \equiv \mathrm{top}(M) \,||\, F_h(F_v(\mathrm{bottom}(M)))$$

Some observations we can make are:

- $h(R_{ccw}(M) = \lfloor S_h(M) \rfloor$
- $w(R_{ccw}(M) = 2w(M)$

Lemma 1 *The rotation operator* $\boldsymbol{R}_{ccw}(\boldsymbol{M})$ *creates a Partial Hypercube Map.*

Proof: Let

$$\begin{aligned} L &\leftarrow \text{top}(M) \\ R &\leftarrow \text{bottom}(M) \\ L_i &\leftarrow \text{cols}(L, i, i), \text{ for } 1 \leq i \leq w(M) \\ R_i &\leftarrow \text{cols}(R, i, i), \text{ for } 1 \leq i \leq w(M) \end{aligned}$$

Therefore, we can write,

$$\begin{aligned} L &= L_1 \,||\, L_2 \,||\, \ldots \,||\, L_{w(M)} \\ R &= R_1 \,||\, R_2 \,||\, \ldots \,||\, R_{w(M)} \end{aligned}$$

and therefore,

$$\begin{aligned} F_h(F_v(R)) &= F_h(F_v(R_1 \,||\, R_2 \,||\, \ldots \,||\, R_{w(M)})) \\ &= F_h(R_{w(M)} \,||\, R_{w(M)-1} \,||\, \ldots \,||\, R_1) \\ &= \underbrace{F_h(R_{w(M)}) \,||\, F_h(R_{w(M)-1}) \,||\, \ldots \,||\, F_h(R_1)}_{w(M)} \end{aligned} \tag{4.1}$$

Recall that,

$$L_i \quad \text{adj} \quad F_h(R_i)$$

since the columns of M are the columns of a Partial Hypercube Map of height > 1
which implies that M has reflective edges about the Horizontal Separator "top(M) adj F_h(bottom(M

Hence, the solution

$$L \,||\, F_h(F_v(R)) \;=\; \underbrace{\overbrace{L_1 \,||\, \ldots \,||\, L_{w(M)}}^{w(M)} \,||\, \overbrace{F_h(R_{w(M)}) \,||\, \ldots \,||\, F_h(R_1)}^{w(M)}}_{2w(M)} \tag{4.2}$$

is formed with a valid concatenation where reflective edges exist about the resulting Vertical Separator and hence, the result is a Partial Hypercube Map. □

Operation 5 $\mathbf{R_{cw}(M)}$ *cuts and pastes the map M by rotating bottom(M) clockwise 180 degrees and appending the result to the left of top(M), where $M = cols(H, i, j)$, for $1 \leq i \leq j \leq w(H)$ and H a Partial Hypercube Map of height > 1.*

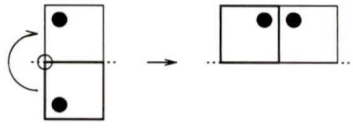


Figure 4.2: $R_{cw}(M)$

Formally, we can define:

$$R_{cw}(M) \equiv F_h(F_v(\text{bottom}(M))) \,||\, \text{top}(M)$$

Some observations we can make are:

- $h(R_{cw}(M) = \lfloor S_h(M) \rfloor$
- $w(R_{cw}(M) = 2w(M)$

Lemma 2 *The rotation operator $\boldsymbol{R_{cw}(M)}$ creates a Partial Hypercube Map.*

Proof: Similar to Lemma 1.

Operation 6 $\mathbf{R_d(M)}$ *removes a Trapezoid from the Map M, where $h(M) \geq 2w(M)$ and $M = cols(H, i, j)$, for $1 \leq i \leq j \leq w(H)$ and H a Partial Hypercube Map of height > 1, by cutting along a 45 degree diagonal that*

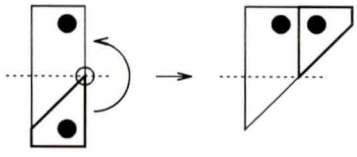


Figure 4.3: $R_d(M)$

intersects $M[\lfloor S_h(M) \rfloor + 1, w(M)]$, *and then appends the remainder of the map (including the diagonal Trapezoid boundary), rotated counter clockwise 180 degrees, to the Trapezoid.*

Formally, for $1 \leq j \leq w(M)$, the solution T can be computed as follows:

$$\begin{aligned} L[i,j] &\leftarrow M[i,j], \text{ for } 1 \leq i \leq \lfloor S_h(M) \rfloor + w(M) + 1 - j \\ R[i,j] &\leftarrow M[i + \lfloor S_h(M) \rfloor, j], \text{ for } 1 \leq i \leq w(M) + 1 - j \\ T &\leftarrow L \,||\, F_h(F_v(R)) \end{aligned}$$

Some observations we can make are:

- elements on the diagonal are in both interim maps, and hence, are duplicated by this operation.
- $h(\text{top}(M)) = h(\text{bottom}(M)) = \lfloor S_h(M) \rfloor$
- $w(M) \leq \lfloor S_h(M) \rfloor$, since $w(M) \leq h(M)/2 \leq \lfloor \frac{1+h(M)}{2} \rfloor = \lfloor S_h(M) \rfloor$
- $h(T) = \lfloor S_h(M) \rfloor + w(M)$
- $w(T) = 2w(M)$

Lemma 3 *The rotation operator* $\boldsymbol{R_d(M)}$ *creates a Trapezoid* T *of height* $\lfloor S_h(M) \rfloor + w(M)$ *and width* $2w(M)$*, where the* $2w(M)$ *elements on the diagonal are used exactly twice.*

Proof: $T \leftarrow L \,||\, F_h(F_v(R))$ is a valid concatenation, since non-null adjacent elements at the boundary of the concatenation are identical to the boundary when forming $R_{ccw}(M)$, which by Lemma 1 is valid.

Therefore, we must show that T has the properties of a Trapezoid.

By the definition of L and R, the diagonal used to break M into L and R is

$$M[\lfloor S_h(M) \rfloor + w(M) + 1 - j, j], \quad \text{for} \quad 1 \leq j \leq w(M)$$

Therefore, the end vertices ($j = 1, w(M)$) of the diagonal are

$$M[\lfloor S_h(M) \rfloor + w(M)] \quad \text{and} \quad M[\lfloor S_h(M) \rfloor + 1]$$

and we have

$$\begin{aligned} h(M) &\geq 2w(M) \\ &\geq \lfloor S_h(M) \rfloor + w(M) \\ &\geq \lfloor S_h(M) \rfloor + 1 \\ &\geq 2 \end{aligned}$$

Therefore, the diagonal defines L as a Trapezoid of height $= \lfloor S_h(M) \rfloor + w(M)$.

We note that indices of the diagonal are the same for L and M, but are different for R, since R is a subgraph of bottom(M). The row index for R is computed by subtracting $\lfloor S_h(M) \rfloor$ from the row index for M. The column index is unchanged.

Therefore, the diagonal boundary of R is

$$R[w(M) + 1 - j, j], \quad \text{for} \quad 1 \leq j \leq w(M)$$

If we let $R' \equiv F_h(F_v(R))$, the diagonal boundary of R' is

$$R'[\lfloor S_h(M) \rfloor + 1 - j', j'] \quad \text{for} \quad 1 \leq j' \leq w(M)$$

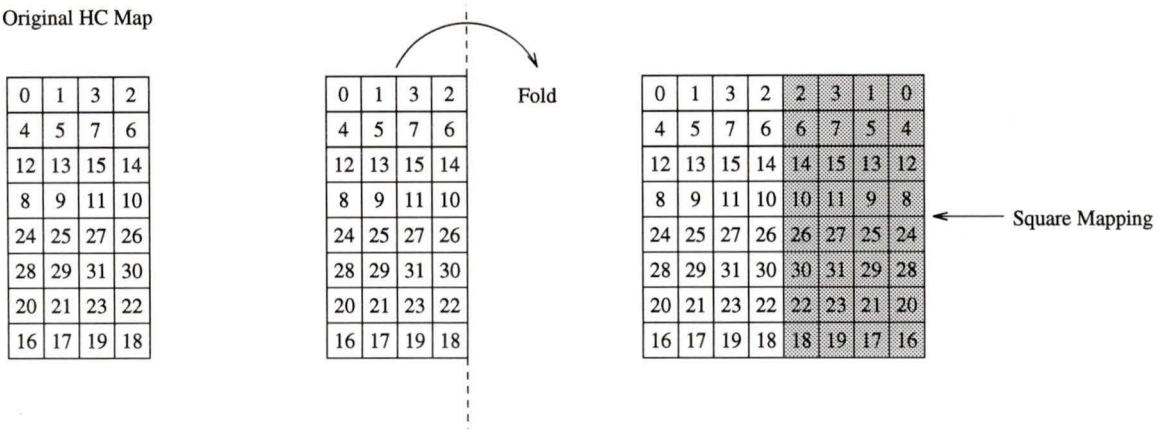


Figure 4.4: Simple Embedding into Odd Dimensional $H(5)$

Therefore, the diagonals form one diagonal of length $2w(M)$ after concatenation since

$$\begin{aligned} h(R') = h(\text{cols}(R', 1, 1)) &= \lfloor S_h(M) \rfloor \\ h(\text{cols}(L, w(M), w(M))) &= \lfloor S_h(M) \rfloor + 1 \\ &= h(R') + 1 \end{aligned}$$

By the decomposition of M into L and R, the diagonal elements are the only elements used more than once (exactly twice) and they correspond to elements in property 3 of a Trapezoid definition. Therefore a Trapezoid is formed.

4.3 Examples

In the examples that follow, illustrations of various maps use highlighted regions to represent parts to be moved or parts that have been moved and shaded regions to represent parts to be copied or parts that have been copied.

Example 1

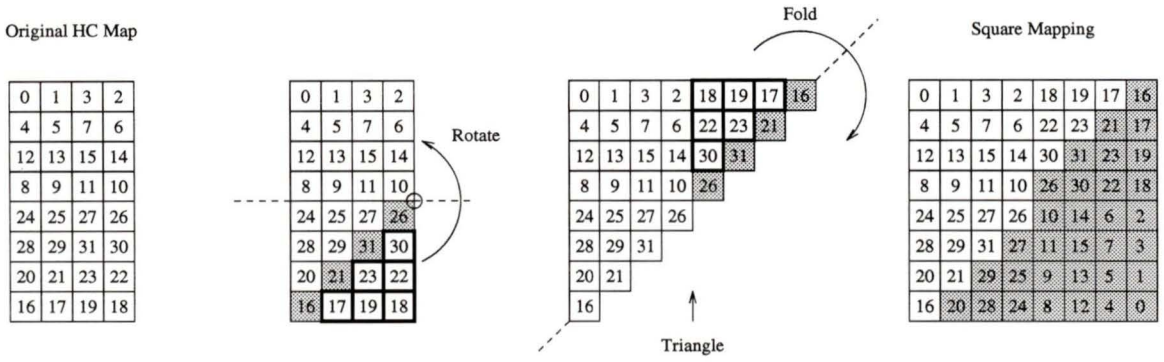


Figure 4.5: General Embedding into Odd Dimensional $H(5)$

When a Host Hypercube is of odd dimension, the solution is easy. Figure 4.4 illustrates an embedding of the largest possible square grid into an odd dimensioned Hypercube with the sample embedding of $G(8,8)$ into $Q(5)$. In so doing, we have illustrated a way of solving the problem for every square grid requiring an odd dimensioned Host.

The technique requires an initial Hypercube Map H, of height h and width w, where $h = 2w$ ($H = \mathrm{HM}(8,4)$ in Figure 4.4). H is then doubled with the vertical fold operation "$F_v(H)$", creating a square map of integers which we use to define a two-to-one mapping from the $h \times h$ grid into the next smaller Hypercube.

This technique works for odd dimensioned Hosts because when d is odd, we can always create an initial Hypercube Map where

$$\begin{aligned} w(H) &= 2^{\lfloor d/2 \rfloor} \\ h(H) &= 2^{\lceil d/2 \rceil} = 2^{\lfloor d/2 \rfloor + 1} = 2w(H) \end{aligned}$$

and since $2|Q(d)| = 2^{d+1}$ is a perfect square when d is odd

$$\sqrt{2^{d+1}} = 2^{\frac{d+1}{2}} = 2^{\lfloor d/2 \rfloor + 1} = h(H)$$

we know the embedding uses every Host vertex twice and hence, represents the largest possible square embedding.

Example 2

In Figure 4.5 we illustrate a different technique that also embeds $G(8,8)$ into $Q(5)$. Initially, we require a Reflective Hypercube Map H, of height h and width w, where $h = 2w$ ($H = \mathrm{HM}_r(8,4)$ in Figure 4.5). Although Example 1 and Example 2 use the same initial Reflective Hypercube Map, we note that Example 1 does not require any restrictions on the form of the Hypercube Map. We then rotate diagonally "$R_d(H)$", forming a triangle T, where every element on the diagonal of the triangle is a duplicate of another element on the diagonal. Finally, we diagonally fold the triangle "$F_d(T)$", forming a square map with a dimension the same as the dimension of the maximum square grid. Since the diagonal fold only duplicates the elements off the diagonal, we have the required two-to-one embedding.

Example 3

We can extend the diagonal fold technique of Example 2 for any Hypercube dimension. Consider for example, the problem of embedding $G(11,11)$ into $Q(6)$. The solution to this problem is illustrated via a modified technique, in Figure 4.6 through Figure 4.11.

Initially, we start with a Reflective Hypercube Map H. The height of this map (call it h_c) is the smallest power of 2 that is greater than or equal to the dimension of the square (call this h). In Figure 4.6, $h = \lfloor\sqrt{2^{6+1}}\rfloor = 11$, $h_c = 2^{\lceil \log_2 h \rceil} = 16$ and $H = \mathrm{HM}_r(16,8)$.

The value "$w = \mathrm{ddc}(H,h)$" is computed, and H is then split into two maps, H_l and H_r, by grouping all columns to the left(inclusive) of column w and all columns to the

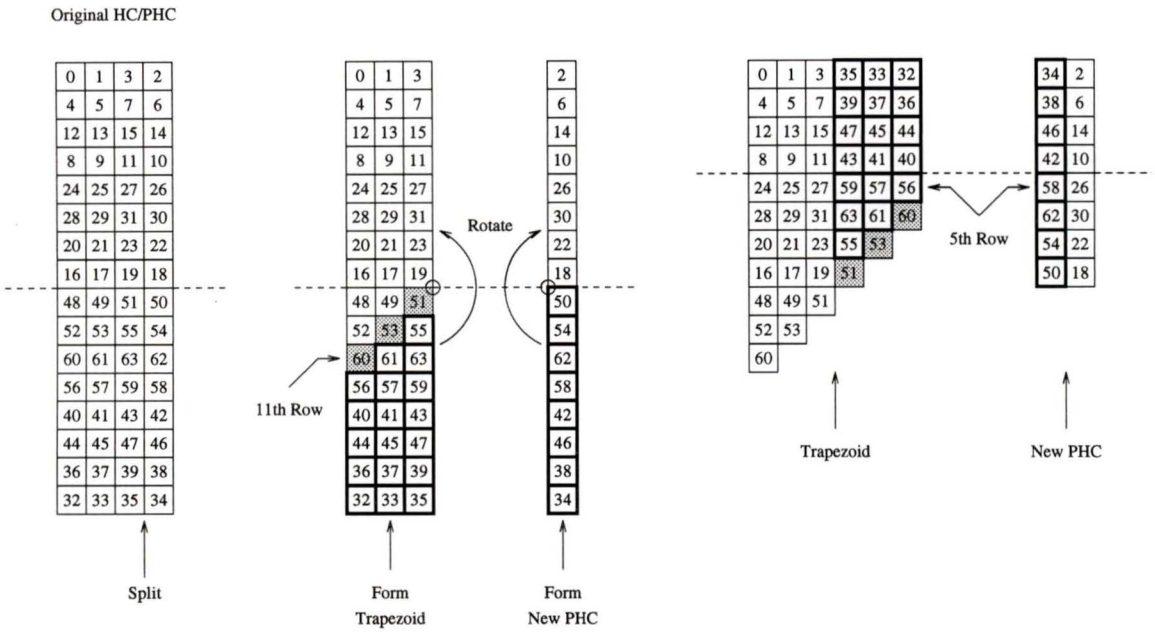


Figure 4.6: Embedding of $G(11,11)$ into $H(6)$ — Iteration 1 / Step 2

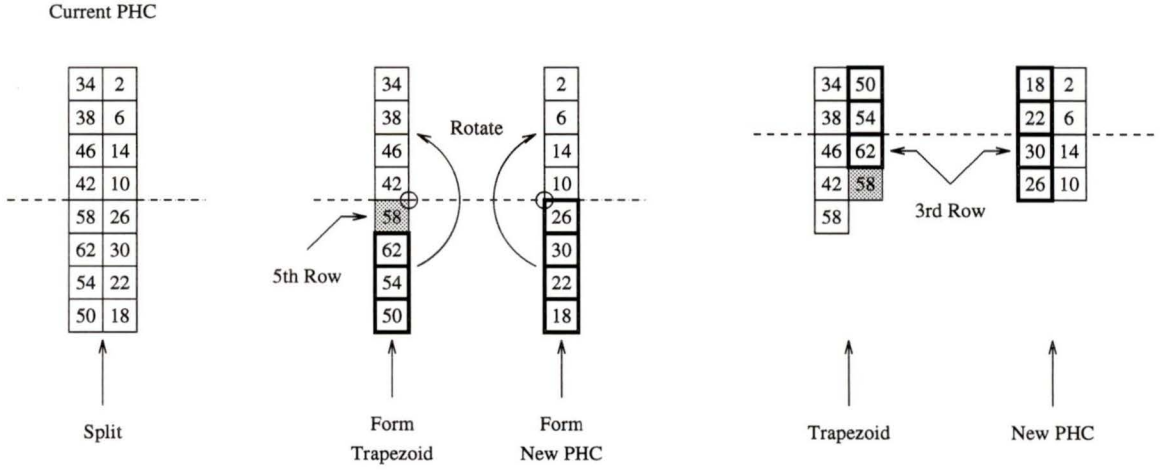


Figure 4.7: Embedding of $G(11,11)$ into $H(6)$ — Iteration 2 / Step 2

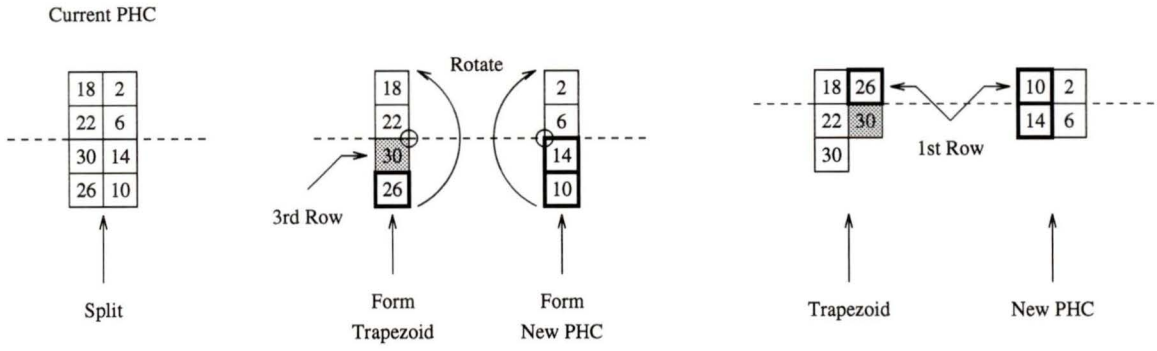


Figure 4.8: Embedding of $G(11, 11)$ into $H(6)$ — Iteration 3 / Step 2

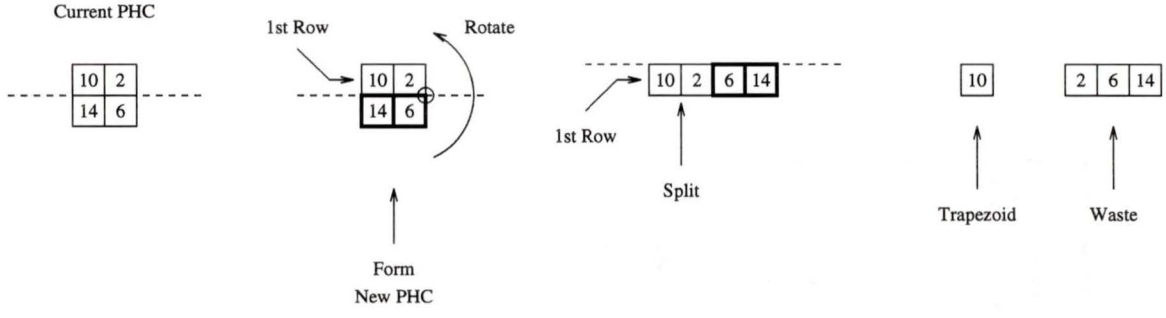


Figure 4.9: Embedding of $G(11, 11)$ into $H(6)$ — Iteration 3 / Step 2

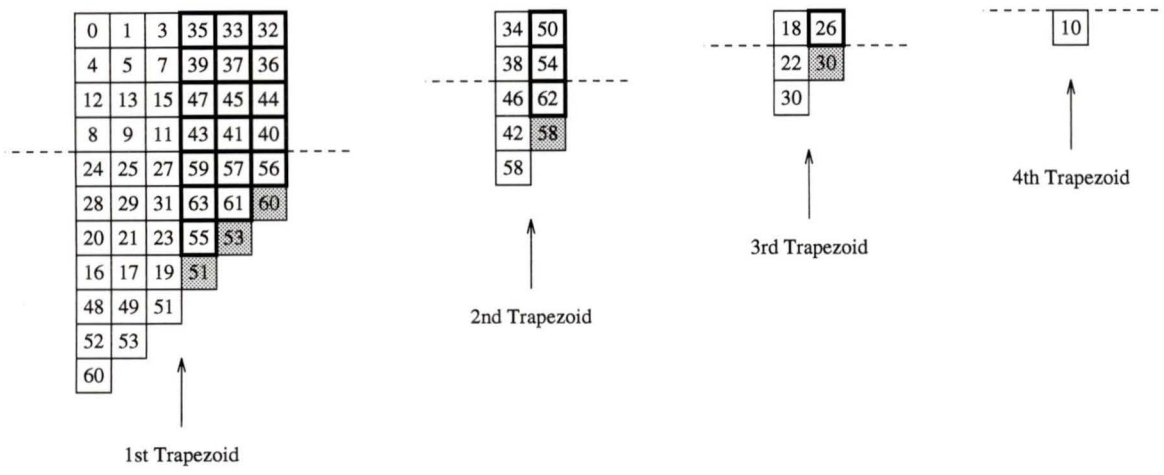


Figure 4.10: Embedding of $G(11, 11)$ into $H(6)$ — Summary of Step 2

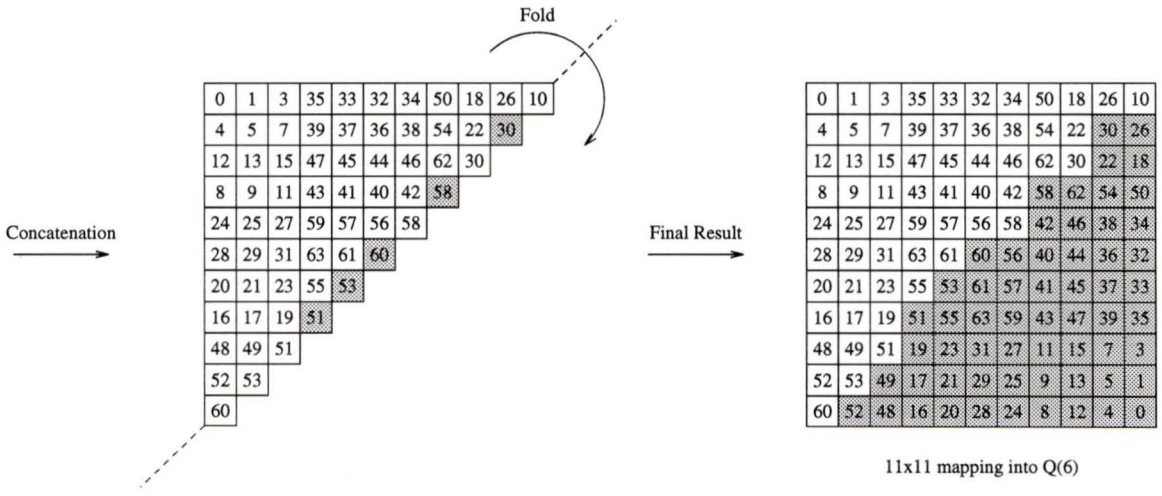


Figure 4.11: Embedding of $G(11, 11)$ into $H(6)$ — Step 3 and 4

right(exclusive) of column w. A diagonal rotation is then applied to the leftmost map "$R_d(H_l)$" and a clockwise rotation to the rightmost map "$R_{cw}(H_r)$", as illustrated in Figure 4.6. The result of $R_d(H_l)$ is a Trapezoid H'_l of height h, since w was computed in a way to guarantee this height. We note that if H'_l were folded diagonally "$F_d(H'_l)$" to create a square $h \times h$ map, there would exist an $h' \times h'$ null region in the upper right, where $h' = h - 2w$ ($h = 5$ in this case). The result of $R_{cw}(H_r)$, by Lemma 2, is a Partial Hypercube Map H'_r of $1/2$ the height of the previous Partial Hypercube Map H. The problem has now been reduced into solving for this smaller region using the remaining map H'_r.

Figure 4.7 illustrates the next step. Using the remaining Partial Hypercube Map, we find a smaller Trapezoid having matching edges between its leftmost column and the rightmost column of the Trapezoid from the previous step. We start by reassigning $H \leftarrow H'_r$, $h \leftarrow h' = 5$ and $h_c \leftarrow h_c/2 = 8$ and simply repeat our previous steps with these new assignments. This time, since $\mathrm{ddc}(H, 5) = 1$, we split the Partial Hypercube Map into two separate maps/columns and apply the rotations "$R_d(H_l)$ and $R_{cw}(H_r)$" as in the previous step. The result of the diagonal rotation is a Trapezoid H'_l. Similarly, if we fold this diagonally "$F_d(H'_l)$", we create a square 5×5 map whose leftmost column has matching edges with the rightmost column of the previous Trapezoid, and with a 3×3 null region in the upper right. The result of the clockwise rotation is a Partial Hypercube Map H'_r of $1/2$ the height.

Figure 4.8 illustrates another repetition of the previous steps with the new smaller Partial Hypercube Map for a height 3 square grid ($H \leftarrow H'_r$, $h_c = 4$ and $h = 3$). The result is a Trapezoid with an incomplete height $h' = h - 2w = 1$ Trapezoid remaining to be solved, and a smaller Partial Hypercube Map of $1/2$ the height.

Figure 4.9 illustrates the choosing of the final 1×1 Trapezoid/vertex. Here, $h_c = 2$ and $h = 1$. Various choices can work in this example, however, in general we

must change the Partial Hypercube Map via the illustrated counter clockwise rotation "$R_{ccw}(H)$" so that ddc(H, h) is guaranteed to exist (i.e. force $h \geq S_h(H)$). In this case, the new Partial Hypercube Map is split and the leftmost map is a single vertex which we use as the final Trapezoid.

Figure 4.10 and Figure 4.11 illustrate the final step of the technique. All the Trapezoids from every stage of the technique are concatenated in the order they occurred, forming a triangle T with all but 1 number along the diagonal occurring twice. The square grid is then completed by diagonally folding the triangle, "$F_d(T)$", and results in each number being used no more than twice.

4.4 Results

Theorem 4 *Any square grid is two-to-one embeddable with dilation 1 into the next smaller Hypercube.*

Proof: The Proof is constructive and follows from the correctness of Algorithm 4.12, which two-to-one translates a Reflective Hypercube Map of dimension "d" into a square map with dimension identical to the dimension of the largest possible square grid that can be embedded into a Hypercube of dimension d. The Algorithm is a formalization of the technique described in Example 3, with only slight differences in the procedure.

Variables used by the Algorithm are:

- d denotes the known dimension of the Host Hypercube Map.

Step 1: **INITIALIZE**
{ Solve for $G(h,h) \rightarrow Q(d)$ }
Set $h \leftarrow \lfloor\sqrt{2^{d+1}}\rfloor$
Set $H \leftarrow \mathrm{HM}_r(d)$, of height $2^{\lceil\log_2 h\rceil}$ **or** $\mathrm{HM}_r(2^{\lfloor d/2\rfloor+1}, 2^{\lceil d/2\rceil-1})$
Set $T \leftarrow$ empty

Step 2: **BODY**
while $(h > 1)$ **do begin**
Set $w \leftarrow \mathrm{ddc}(H,h)$
Set $H_l \leftarrow \mathrm{cols}(H,1,w)$
Set $H_r \leftarrow \mathrm{cols}(H,w+1,w(H))$
Set $T \leftarrow T \,||\, R_d(H_l)$
Set $H \leftarrow R_{cw}(H_r)$

Set $h \leftarrow h - 2w$
if $(h > 1)$ **then begin**
{ while row $h \in \mathrm{top}(H)$ do }
while $(h \leq S_h(H))$ **do begin**
Set $H \leftarrow R_{ccw}(H)$
end
end
end

Step 3: **COMPLETE TRIANGLE**
if $(h = 1)$ **then begin**
Set $T \leftarrow T \,||\, H[1,1]$
end

Step 4: **FINALIZE**
Return $F_d(T)$

Figure 4.12: Square Embedding Algorithm

- h denotes the dimension of the square map the algorithm is trying to compute $(G(h,h) \rightarrow H)$.
- H denotes the Partial Hypercube Map the algorithm is trying to translate.
- w denotes a column boundary used to partition H into the maps H_l and H_r.
- H_l denotes the left partition of H.
- H_r denotes the right partition of H.
- T denotes a trapezoid representing the mapping so far computed. As the algorithm progresses, smaller trapezoids are concatenated with T, until T is a triangle.

Step 1: We initialize the variables, where h is the size of the largest square grid that can be embedded into $Q(d)$. Since $|Q(d)| = 2^d$ and in a two-to-one mapping, we can map 2 vertices to each of the 2^d Host nodes, we can have at most 2^{d+1} vertices in the square, which implies the largest square has dimension $h = \lfloor\sqrt{2^{d+1}}\rfloor$.

Step 2: We recognize the main loop contains an invariant as follows:

1. $h \times h \leq 2|H|$
2. H is a partial Hypercube map.
3. $S_h(H) < h \leq h(H)$, unless $0 \leq h \leq 1$.
4. If T and H are not empty, the right hand column of T has no mismatches along the shared border with the left hand column of H.

At entry to the outer loop, property 1 of the invariant holds because of the initialization. Property 2 also holds, since H is a reflective Hypercube Map and is therefore a Partial Hypercube Map. To show property 3, we have

$$h = 2^{\log_2 h} \leq 2^{\lceil \log_2 h \rceil} = h(H)$$

Therefore

$$h \leq h(H) \tag{4.3}$$

Also

$$h = \frac{2^{(log_2 h)+1} + 1}{2} - 1/2 \geq \frac{2^{(log_2 h)+1} + 1}{2}$$

But

$$(log_2 h) + 1 > \lceil \log_2 h \rceil$$

Therefore

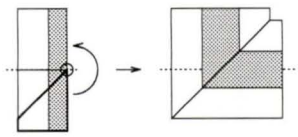
$$h > \frac{2^{\lceil \log_2 h \rceil} + 1}{2} = \frac{h(H) + 1}{2} = S_h(H)$$

and hence

$$h > S_h(H) \tag{4.4}$$

Therefore $S_h(H) < h \leq h(H)$ and hence, property 3 of the invariant holds. Property 4 holds since T is initially empty. Hence, the invariant holds at initial entry to the loop.

Let us show that the properties are invariant throughout the loop. We must show that given H and h, $w = \mathrm{ddc}(H, h)$ exists. Since $h > S_h(H)$ by property 3 of the invariant, we have that $\mathrm{ddc}(H)$ exists only if the map H is wide enough. That is, $w \leq w(H)$, or $w > w(H)$ in the case that H is not wide enough. The latter would contradict property 1.



To illustrate the contradiction, we could do the rotation by padding just enough columns to the right of the map (call this new map H'), so that ddc(H') exists. The result can be diagonally folded to achieve a $h \times h$ map with a $(h(H') - h) \times (h(H') - h)$ missing square map in the top righthand corner. Excluding the missing square portion, we would still be using every vertex from H' exactly twice (including vertices on the diagonal). This contradicts property 1, since $|H'| > |H|$.

By property 4, the rightmost column of T has matching edges with the leftmost column of H. Since the leftmost column of H_l is equal to the leftmost column of H, we can concatenate $R_d(H_l)$ with T. After concatenation, we have constructed a trapezoid T with the rightmost column being the bottom $h(H) + 1 - h$ elements (in reverse order) from the leftmost column of H_l. We also have constructed a new Partial Hypercube Map with the leftmost column being the bottom half of the rightmost column of H_r (in reverse order). Vertical reflective edges exist in H, which implies that the leftmost column of H_l has matching edges with the rightmost column of H_r. Hence, the rightmost column of T has matching edges with the leftmost column of the new Partial Hypercube Map H. Hence, property 4 is maintained.

The height of the rightmost column of $T = h - 2w + 1$. This implies that the height of the remaining Trapezoid is $h - 2w$. If $h - 2w > 1$, to ensure property 3 remains true, the Algorithm cuts the height of H in half repeatedly until the assumption is true. The operation $R_{ccw}(H)$ used to achieve this does not change the vertices in the top half of column 1 of H, which implies that matching edges between the rightmost column of T and the leftmost column of H still exist. Therefore property 3 and

property 4 hold.

To show that the outer while loop terminates, we note that $h(H)$ is reduced by at least $1/2$ with each iteration. Since property 3 is maintained, we have $h \leq h(H)$, or $h \in \{0, 1\}$, so eventually the outer while loop will terminate.

Step 3: If upon exit from the outer while loop of Step 2, $h = 1$, we can concatenate one final corner element of H with T to ensure the formation of a triangle. This is possible since property 1 and property 4 are true after leaving Step 2.

The end result of Step 3 is a trapezoid taking the form of a triangle, where off diagonal elements are unique, and with the height being the dimension of the square we are trying to embed.

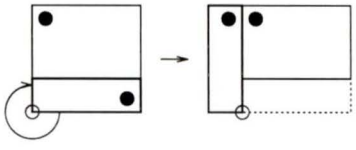
Step 4: We diagonally fold the triangle T and achieve our required two-to-one mapping. □

Chapter 5

Nearly Square Grids

5.1 Introduction

We have shown how to embed a square grid into the next smaller Hypercube. In this section we look at grids where the height and width differ slightly from the dimension of the largest embeddable square grid (we call these **nearly square** grids). For many of these, we find that we can embed them into the next smaller Hypercube using a development from the square grid embedding technique. We will need the following definition.



Operation 7 *A* **Row Rotation** *of a square map M, denoted by $\boldsymbol{R_r(M, i)}$, $1 \leq i < h(M)$, rotates the bottom i rows of M by 270 degrees about the bottom left corner of the map, and appends them to the leftmost column of M in their new orientation. The result of this operation is a $(h(M)-i) \times (h(M)+i)$ grid, plus an $i \times i$ grid below the lower left corner.*

Formally, for $1 \leq c \leq i$, the solution S can be computed as follows:

$$\begin{aligned} L[r,c] &\leftarrow M[h(M)-i+c, w(M)+1-r], \text{ for } 1 \leq r \leq w(M) \\ R[r,c] &\leftarrow \begin{cases} M[r,c], & \text{for } 1 \leq r \leq h(M)-i \\ \text{null}, & \text{for } h(M)-i < r \leq h(M) \end{cases} \\ S &\leftarrow L \,||\, R \end{aligned}$$

We note that the solution could also be viewed as a $h(M) \times (h(M)+i)$ grid with $i \times h(M)$ null elements in the bottom right. See for example, Figure 5.1 and Figure 5.2 for an illustration of $R_r(M,1)$ and $R_r(M,2)$, respectively.

Lemma 4 *$R_r(M,i)$ returns a map when applied to a square map created by Algorithm 4.12.*

Proof: We need to show the bottom row of M is within Hamming distance 1 of the leftmost column of M, since this is where the seam occurs (i.e. where the rotated rows are appended to the remaining part of the columns). Since the square grid from Algorithm 4.12 was created by a diagonal fold of a triangle T, "$F_d(T)$", the bottom row is identical to the leftmost column, and therefore the Hamming distance between the bottom row and the leftmost column is 0. □

5.2 Examples

Example 4

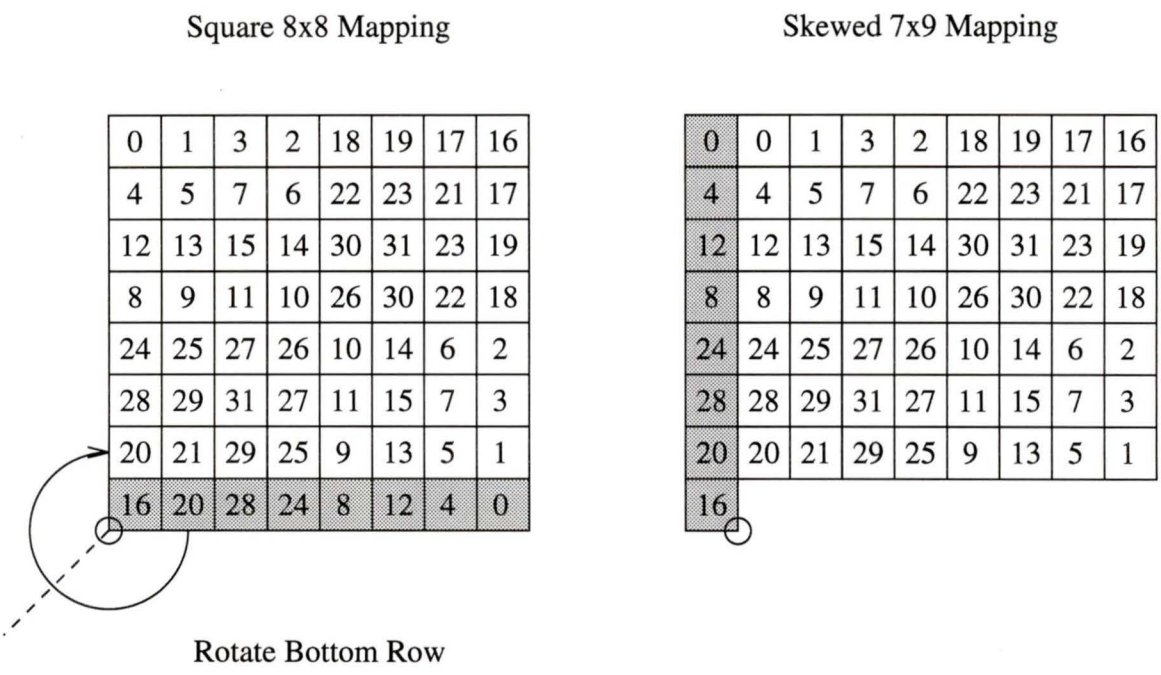


Figure 5.1: General Embedding of $G(8,8)$ into $G(7,9)$

In Figure 5.1, we illustrate how to embed $G(7,9)$ into $Q(5)$ via the solution for $G(8,8)$ into $Q(5)$. Since the square map M is generated by Algorithm 4.12, by Lemma 4 we can apply a row rotation to transform the shape of the map. As illustrated, we turn the bottom row of this map into the leftmost column by rotating the bottom row 270 degrees, "$R_r(M,1)$". The result is a map of height 7 and width 9, with one element outside of these dimensions "M[8,1]", which we call the **waste**.

Example 5

As illustrated in Figure 5.2, we embed $G(6,10)$ into $Q(5)$ by applying the row rotation to the bottom 2 rows of the 8×8 map. The final result is a 6×10 map with a waste of $2^2 = 4$ elements.

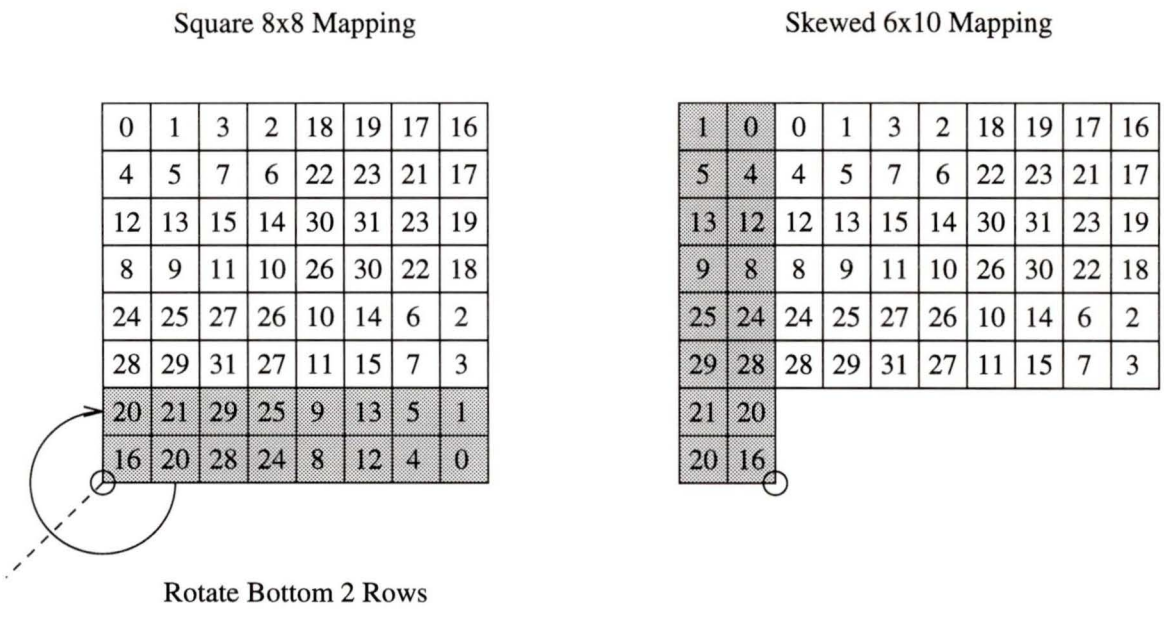


Figure 5.2: General Embedding of $G(8,8)$ into $G(6,10)$

5.3 Results

The two previous examples have demonstrated a technique for solving nearly square grids. In the following we describe a technique for embedding grids of the form $G(b-a, b+a)$ into the next smaller Hypercube, for some $0 \leq a < f(b)$.

Lemma 5 *Grids of the form $G(2^n - 1, 2^n + 1)$, $n > 0$, can be embedded into the next smaller Hypercube.*

Proof: From Example 4, we show how to embed $G(2^n - 1, 2^n + 1)$ into $Q(2n-1)$. Therefore, we need to show the Hypercube of dimension $d = 2n-1$ is the next smaller Hypercube. The next smaller Hypercube is of dimension

$$d = \lceil \log_2 |G| \rceil - 1$$

$$= \lceil \log_2 (2^{2n} - 1) \rceil - 1$$
$$= 2n - 1$$

□

Lemma 6 *Grids of the form $G(b-a, b+a)$, $a < b$, can be embedded into the Hypercube of dimension $d = \lceil 2 \log_2 b \rceil - 1$.*

Proof: We can construct a square embedding map M for a $b \times b$ grid using Algorithm 4.12. By Lemma 4, we can perform a row rotation on the bottom $b - a$ rows of the map, "$R_r(M, a)$", forming the required mapping. The $b \times b$ map requires a Hypercube of dimension $d = \lceil \log_2 b^2 \rceil - 1 = \lceil 2 \log_2 b \rceil - 1$.

Theorem 5 *If $0 \leq a < \sqrt{b^2 - 2^{\lceil 2 \log_2 b \rceil - 1}}$, then grids of the form $G(b-a, b+a)$, can be embedded into the next smaller Hypercube.*

Proof: By Lemma 6, there exists an embedding into the Hypercube of dimension $d = \lceil 2 \log_2 b \rceil - 1$. We will show that if $0 \leq a < \sqrt{b^2 - 2^{\lceil 2 \log_2 b \rceil - 1}}$, then $Q(d)$ is the next smaller Hypercube. By Theorem 4, we know that $Q(d)$ is the next smaller Hypercube when $a = 0$. Therefore, we need to determine when $Q(d)$ is still the next smaller Hypercube for other non-zero values of a. This is true as long as $|G| > |Q(d)|$. Therefore since

$$|G| = (b - a)(b + a) = b^2 - a^2$$

and

$$|Q(d)| = 2^d = 2^{\lceil 2 \log_2 b \rceil - 1}$$

we have

$$b^2 - a^2 > 2^{\lceil 2 \log_2 b \rceil - 1}$$

which is true, iff

$$a < \sqrt{b^2 - 2^{\lceil 2\log_2 b \rceil - 1}}$$

□

Corollary 1 *If* $0 \leq a < 2^{n-1/2}$, *then grids of the form* $G(2^n - a, 2^n + a)$ *can be embedded into the next smaller Hypercube.*

Proof: By applying Theorem 5 with $b = 2^n$, we get:

$$\begin{aligned} a^2 &< b^2 - 2^{\lceil 2\log_2 b \rceil - 1} \\ &= (2^n)^2 - 2^{\lceil 2\log_2 (2^n) \rceil - 1} \\ &= 2^{2n} - 2^{2n-1} \\ &= 2^{2n-1} \end{aligned}$$

Therefore we have, $a < 2^{n-1/2}$. □

Chapter 6

Grids of Height $(2^n - 1)$

6.1 Introduction

We have shown how to embed grids of the form $G(2^n - 1, 2^n + 1)$. We can use part of this result to derive an algorithm that solves for all grids of height $= 2^n - 1$ and width $>$ height. We do so by collecting the one wasted element (unused vertex) in replicated solutions to embeddings of Guests, such as $G(2^n - 1, 2^n + 1)$ — using that waste in solutions for smaller Guests — and piecing together the various solutions to form one large solution.

Definition 11 *A* **Base Grid** *of height h is a grid $G(h, w)$ such that $hw = 2^i - 1$ and $w > h$, for some integer i.*

Note that a base grid is a grid for which the embedding into a Hypercube yields a waste of 1 (for example, $G(3, 5)$ and $G(3, 21)$ are base grids).

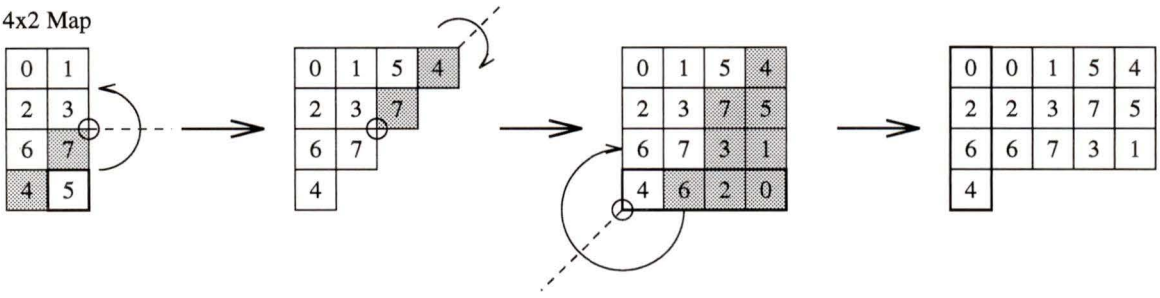


Figure 6.1: $G(3,5)$ into $HM(4,2)$ Embedding

6.2 Examples

Example 6

In Figure 6.1 we show an embedding of $G(3,5)$ into $Q(3)$ that illustrates the simplest height 3 grid that we solve for. $G(3,5)$ is also a base grid which generates a waste of 1.

Example 7

In Figure 6.2 we show an embedding of $G(3,10)$ into $Q(4)$ that illustrates how an embedding of the largest height 3 Grid into the next smaller Hypercube $Q(4)$, can be solved by using the result in Example 6. This demonstrates that we only need to replicate the base grid solution $G(3,5)$ when the total waste is not large enough to allow another column in the Grid. Here the total waste is 2, and we need a total waste of at least 3 for a larger width Grid to be embeddable into $Q(4)$.

Example 8

In Figure 6.3 we show an embedding of $G(3,8)$ into $Q(4)$ that uses 3 of the 4 columns from HM(4,4). There are $3 \times 8 = 24$ vertices embedded into $3 \times 4 = 12$

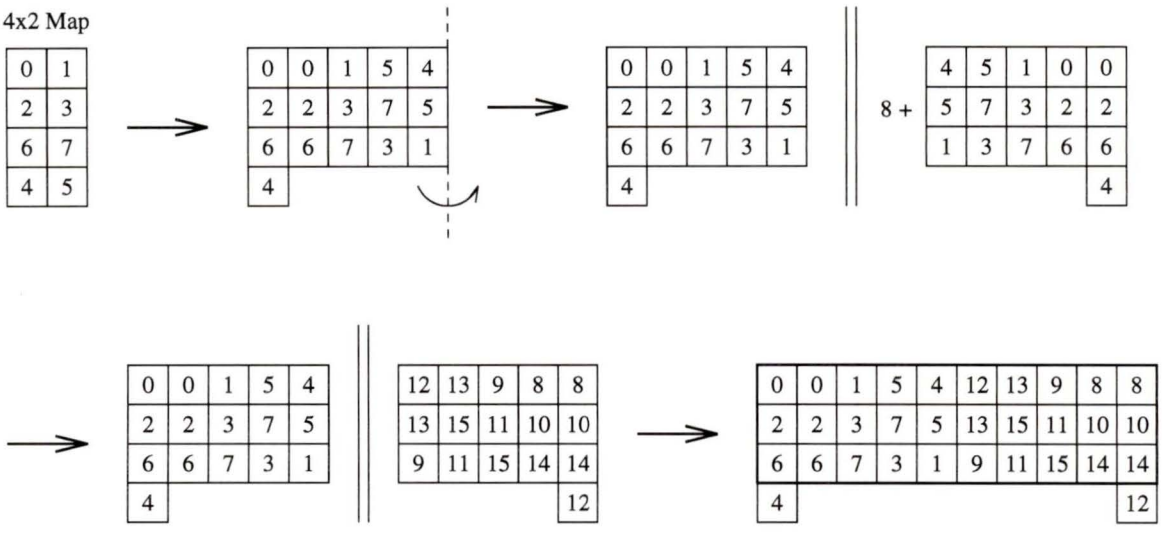


Figure 6.2: $G(3, 10)$ into $HM(4, 4)$ Embedding

nodes and therefore produces no waste. Although this is not the pattern we ultimately use for embedding $G(3, 8)$ into $Q(4)$, it illustrates a technique that reuses the waste from an embedding of $G(3, 5)$ into $Q(3)$ (label 11 in this case) and which forms the basis of our complete solution. We formalize this result in Lemma 7.

Example 9

In Figure 6.4 we show an embedding of $G(3, 21)$ into $Q(5)$. Notice that we don't simply replicate the base solution as in Example 7, since this can only solve for $G(3, 20)$ which is not the largest height 3 Grid. $G(3, 21)$ is the largest height 3 Grid and is an example of a base grid since the waste of an embedding would be 1.

In our example, we decompose the 4×8 map into two 4×3 maps and one 4×2 map. The two 4×3 maps are solved the same way as in Example 8, forming two 3×8 solutions with zero waste and the 4×2 map is solved the same way as in Example 6,

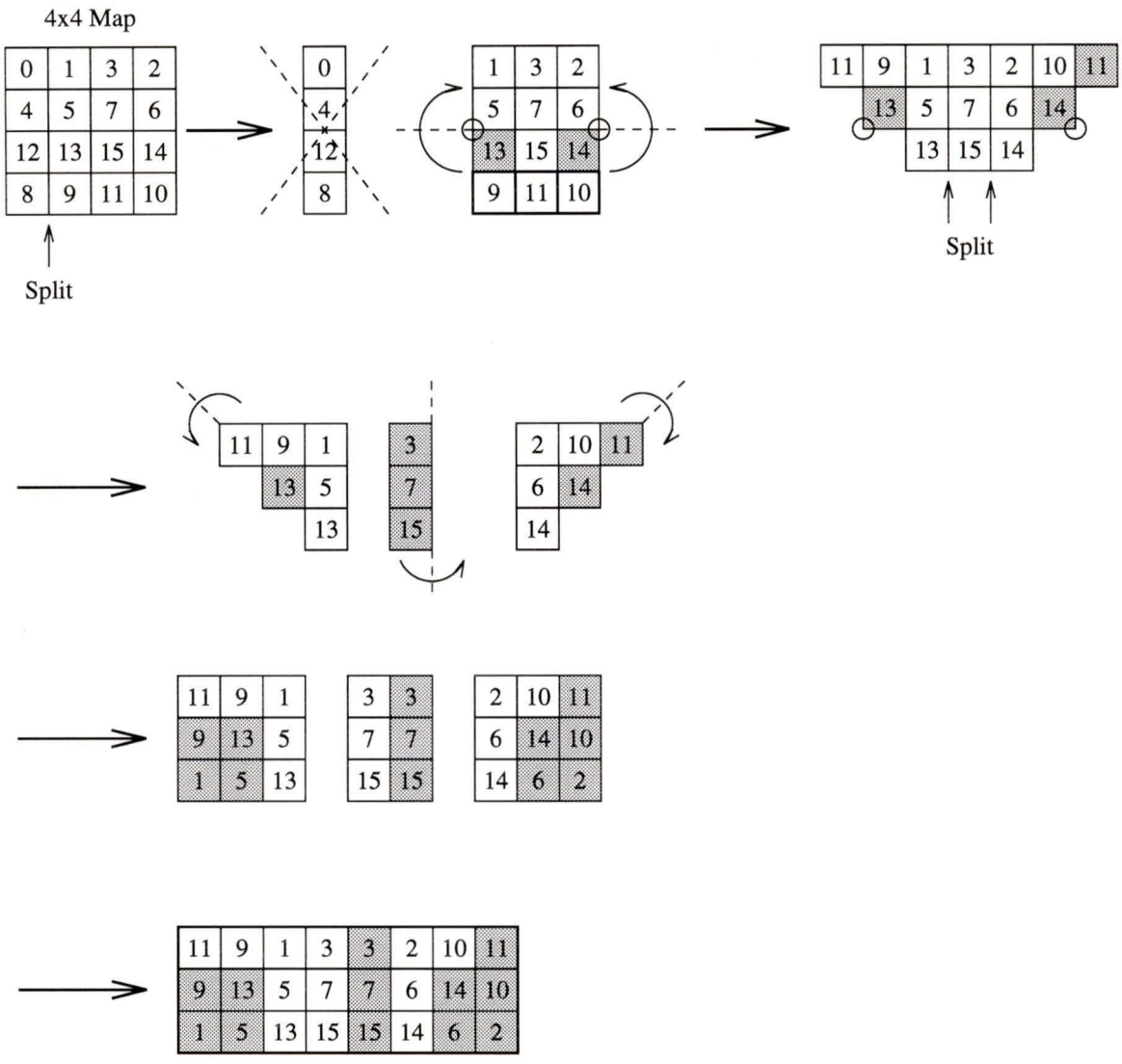


Figure 6.3: $G(3, 8)$ into $HM(4, 4)$ Embedding Pattern

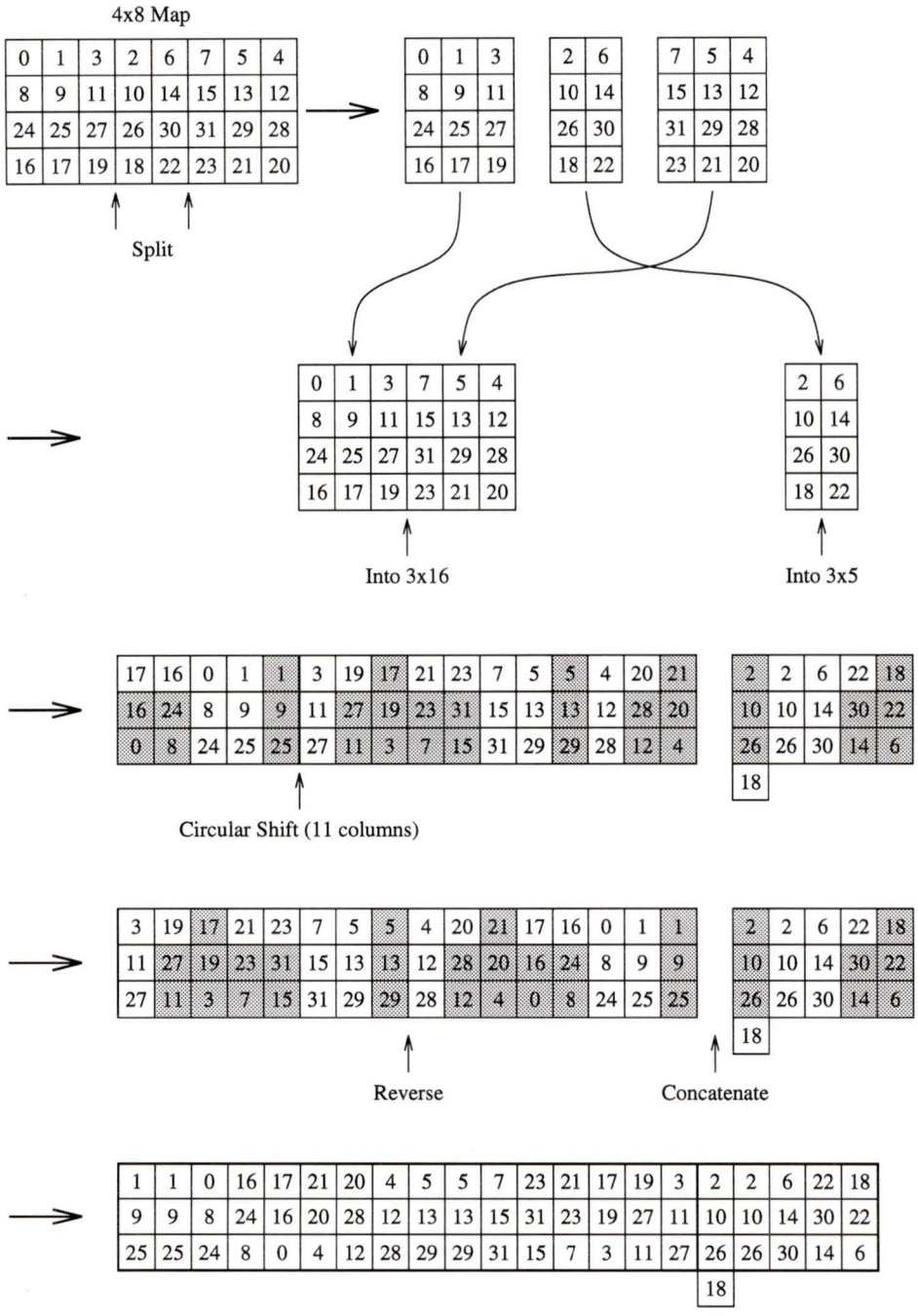


Figure 6.4: $G(3, 21)$ into $HM(4, 8)$ Base $k = 2$ Embedding

forming a 3×5 solution with a waste of 1. These smaller solutions are then pieced together, creating a solution where adjacent vertices are Hamming distance 1 apart.

Notice that when we split the 4×8 map up and solved for $G(3,5)$ into HM$(4,2)$, we decomposed the problem of solving for the base grid $G(3,21)$ into solving for the next smaller base grid $G(3,5)$. This is the insight we use to derive a recursive algorithm that solves for Grids of height $2^n - 1$ and any width > height.

6.3 Results

Lemma 7 *$G(2^n - 1, 2^{n+1})$ can be embedded into $Q(2n)$, using the first $2^n - 1$ columns of the Hypercube map HM$(2^n, 2^n)$.*

Note that an embedding of this form will produce no waste, other than the unused rightmost column (column 2^n) of the Hypercube map. Recall that Figure 6.3 illustrates such an embedding, but where the leftmost column is unused.

Proof: Let $H \leftarrow$ HM$(2^n, 2^n)$ and form the following two solutions by the technique illustrated in Figure 6.1

$S_l \leftarrow$ solution for $G(2^n - 1, 2^n + 1)$ using the map "reverse(cols$(H, 1, 2^{n-1})$)"

$S_r \leftarrow$ solution for $G(2^n - 1, 2^n + 1)$ using the map "cols$(H, 2^{n-1}, 2^n - 1)$"

We note that column 2^{n-1} of H is used for both S_l and S_r. As a consequence, the first and second column of each solution (solutions S_l and S_r) are identical. This implies the top $2^n - 1$ vertices in column 2^{n-1} of H are mapped four times, so we remove two of these columns (first column from S_l and S_r) and form the solution

$$S = \text{reverse}(\text{cols}(S_l, 2, 2^n + 1)) \,||\, \text{cols}(S_r, 2, 2^n + 1)$$

which is a two-to-one mapping to the problem and has zero waste, since

$$|S| = (2^n - 1)(2^{n+1}) = 2(2^n \times (2^n - 1))$$

□

Corollary 2 *$G(2^n-1, 2^{n+1})$ can be embedded into $Q(2n)$, using the last 2^n-1 columns of the Hypercube map $HM(2^n, 2^n)$.*

Note that an embedding of this form will produce no waste, other than the unused leftmost column (column 1) of the Hypercube map.

Proof: By symmetry of the map $\mathrm{HM}(2^n, 2^n)$, this Corollary is identical to Lemma 7.
□

Lemma 8 *If we can solve*

$$G\left(2^n - 1, \frac{2^{n(k+1)} - 1}{2^n - 1}\right) \rightarrow Q(n(k+1) - 1), \text{ for } k \geq 1$$

then we can solve for $G(h, w)$, where $h = 2^n - 1 < w$.

Proof: By Lemma 5 we can solve for $G(2^n - 1, 2^n + 1) \rightarrow Q(2n - 1)$. For larger instances, we can replicate this base solution by doubling/quadrupling/etc. as long as the accumulated waste after the ith replication (call this W_i, where $i = 0$ for the base 1 solution) is insufficient to allow for another column (i.e. $W_i < h$). Since $W_0 = 1$ for the base 1 solution, we have $W_i = 2^i$. Therefore, for $n > 1$, $(W_i = 2^i) < (h = 2^n - 1)$, iff $i < n$. For $i = n$, the new (base 2) solution will have waste $2^i - h = 2^n - (2^n - 1) = 1$. Therefore, we can replicate the base 2 solution up to $n-1$ times also. Similarly, every base k solution can be replicated up to $n - 1$ times before requiring a new base $k + 1$ solution and every base k solution yields a waste of 1. Therefore we only need to

solve for dimensions $d = \{2n-1, 3n-1, 4n-1, \ldots\}$. That is, each base k solution requires a Hypercube of dimension $d_k = n(k+1) - 1$, for $k \geq 1$.

We note that:

$$\begin{aligned} w(\text{base } 1) &= 1 + 2^n \\ w(\text{base } 2) &= 1 + 2^n(1 + 2^n) = 1 + 2^n + 2^{2n} \\ w(\text{base } 3) &= 1 + 2^n w(\text{base } 2) = 1 + 2^n + 2^{2n} + 2^{3n} \end{aligned}$$

Therefore,

$$w(\text{base } k) = \sum_{i=0}^{k} (2^n)^i = \frac{(2^n)^{k+1} - 1}{2^n - 1}, \text{ for } k \geq 1$$

□

We have shown that we only have to compute an embedding for base grids of height $2^n - 1$ in order to compute the embedding for grids of height $2^n - 1$ and any width > height. In Algorithm 6.5, we solve the embedding problem for grids of height $2^n - 1$ and any width > height, assuming there exists a way of computing the base grid embeddings. We present this Algorithm only to show the problem decomposes simply into solving for a base grid and further explanation is not necessary. The remainder of this section will be dedicated to solving for base grids.

Theorem 6 *Grids of height* $(2^n - 1)$ *can be embedded into the next smaller Hypercube for width > height.*

The proof is constructive and follows from the correctness of Algorithm 6.6, which constructs a solution for base grids of height $2^n - 1$, width $\frac{2^{n(k+1)}-1}{2^n-1}$. The Algorithm is recursive and to solve for a height $(2^n - 1)$ base k grid, we invoke the function by:

$$\text{BASE\_SQUEEZE}(H, n, k, \text{dummy}),$$

Step 1: **INITIALIZE**

{Assume d is dimension of next smaller Hypercube for $G(2^n - 1, w) \rightarrow Q(d)$}

Set $k \leftarrow$ largest integer such that $n(k+1) - 1 \leq d$

Set $H \leftarrow$ solution map for $G\left(2^n - 1, \frac{2^{n(k+1)}-1}{2^n-1}\right) \rightarrow Q(n(k+1) - 1)$

Step 2: **LOOP — double solution until solved**

While $w(H) < w$ do

Set $H \leftarrow H \,||\, (\text{reverse}(H) + |H|)$

Figure 6.5: Height $2^n - 1$ Embedding Algorithm

where $H = \text{HM}(2^n, 2^{nk-1})$ is the next smaller Hypercube map required for a base k solution.

The parameters used in Algorithm 6.6 are:

- ***H*** denotes a Host Partial Hypercube Map. Note that initially H is a Hypercube map, but subsequent calls have H passed in as a Partial Hypercube Map with the dimension of the original Hypercube map.
- ***n*** denotes the $\log_2$ of the height of the Hypercube map H (i.e. $2^n = h(H)$).
- ***k*** denotes base k grid we are solving ($k \geq 1$).

The variables used in Algorithm 6.6 are:

- ***L*** and ***R*** denote **left/right** partitions of H formed by extracting and concatenating column segments of H.

{Return a map defining the embedding of the largest grid of height $2^n - 1$ into H. H is a Hypercube map of dimension $n(k+1) - 1$ and height 2^n. "f" is the column index indicating the column of H whose top $2^n - 1$ elements are the same as the first column of the returned map.}

Function BASE_SQUEEZE($H, n, k,$ var f)

Step 1: **BASE CASE (k=1)** { Solve $G(2^n - 1, 2^n + 1) \rightarrow Q(2n-1) \equiv \mathrm{HM}(2^n, 2^{n-1})$ }
if (k=1) then begin
Set $f \leftarrow 1$
return (Solution for $G(2^n - 1, 2^n + 1) \rightarrow H$)
end

Step 2: **DIVIDE**
Let H_i denote "cols($(i-1)2^n + 1, i2^n$)", for $1 \leq i \leq (p = 2^{n(k-1)-1})$
Set $L \leftarrow \mathrm{cols}(H_1, 1, 2^n - 1) \,||\, \mathrm{cols}(H_2, 2, 2^n) \,||\, \mathrm{cols}(H_3, 1, 2^n - 1) \,||\, \ldots \,||\, \mathrm{cols}(H_p, 2, 2^n)$
Set $R \leftarrow \mathrm{cols}(H_1, 2^n, 2^n) \,||\, \mathrm{cols}(H_2, 1, 1) \,||\, \mathrm{cols}(H_3, 2^n, 2^n) \,||\, \ldots \,||\, \mathrm{cols}(H_p, 1, 1)$

Step 3: **CONQUER**
Set $S_L \leftarrow \mathrm{pattern}(L)$
Set $S_R \leftarrow \mathrm{BASE\_SQUEEZE}(R, n, k-1, f)$

Step 4: **MERGE**
if even(f) then begin
a) Set $t \leftarrow f2^{n+1} - 2^n - 1$
b) Set $s \leftarrow w(S_L) - t = w(S_L) - f2^{n+1} + 2^n + 1$
c) Set $S \leftarrow \mathrm{shift}(S_L, s) \,||\, S_R$
d) Set $f \leftarrow f2^n - 2^{n-1} + 1$
end

else begin {odd f}
a) Set $t \leftarrow f2^{n+1} - 2^n + 2$
b) Set $s \leftarrow w(S_L) - t + 1 = w(S_L) - f2^{n+1} + 2^n - 1$
c) Set $S \leftarrow \mathrm{reverse}(\mathrm{shift}(S_L, s)) \,||\, S_R$
d) Set $f \leftarrow f2^n - 2^{n-1}$
end

return (S)

Figure 6.6: Height $2^n - 1$ Base k Embedding Algorithm

- $\boldsymbol{S}_L$ and $\boldsymbol{S}_R$ denote the embedding solutions for the partitions L and R, respectively.
- $\boldsymbol{S}$ denotes the embedding solution for H.
- $\boldsymbol{f}$ denotes the column in H whose top $2^n - 1$ elements are identical to the **first** columns elements in the returned solution map S.
- $\boldsymbol{t}$ denotes the column in solution S_L which is adjacent to the first column in S_R (i.e. the top $2^n - 1$ elements of column f in R).
- $\boldsymbol{s}$ denotes the extent of a right **circular shift** on S_L which results in column t at one end and a column at the opposite end whose elements are the same as the top $2^n - 1$ elements of some column in H.

Proof: Since the procedure is recursive, we only need to show that the base case $(k = 1)$ is correct and for $k > 1$, the Divide, Conquer and Merge stages of the algorithm are correct, using the assumption that the algorithm is correct for smaller instances of k (i.e. proof by induction on k).

At entry, the function parameters obey the following:

1. $k > 0$
2. H's underlying graph is isomorphic to a Hypercube of dimension $d_k = n(k + 1) - 1$.
3. $h(H) = 2^n$
4. $w(H) = 2^{nk-1}$

At exit the following is assumed:

1. The function returns a 2-to-1 mapping for:

$$G\left(2^n - 1, \frac{2^{n(k+1)} - 1}{2^n - 1}\right) \rightarrow Q(n(k+1) - 1)$$

2. Parameter f is a valid column index of H such that the top $2^n - 1$ elements of column f in H are the same as column 1 of the returned solution map.

Step 1: We are required to solve the base case ($k = 1$). We know this can be solved by Lemma 5 (i.e. $G(2^n - 1, 2^n + 1) \rightarrow Q(2n - 1)$), so we must show that f is set correctly. By construction, the the top $2^n - 1$ elements of the first column in H (column $f = 1$) will be identical to the first and second column of the solution.

Step 2: Since $w(H) = 2^{nk-1}$, we have $w(H) = 2^{n(k-1)+n-1} = 2^n\, 2^{n(k-1)-1}$. Hence, $2^n | w(H)$ p times, where $p = 2^{n(k-1)-1}$ and $n, k > 1$. Therefore we can partition H into p smaller Hypercube maps $H_1, H_2, \ldots, H_p$, where H_1 is the first 2^n columns of H, and H_2 is the next 2^n columns of H, and so on.

We then create the partition L, being the concatenation of the first $2^n - 1$ columns from H_1, the last $2^n - 1$ columns from H_2, and so on, alternating from first to last, until we concatenate the last $2^n - 1$ columns from H_p ("last", since p is even). We also create the partition R, being the concatenation of the last column of H_1, the first column of H_2, and so on, alternating between the first and last column until we concatenate the first column of H_p (i.e. the columns not it L).

Since reflective edges exist between the H_i's and since we choose columns equidistant from the vertical separator between adjacent H_i's, the concatenations used to form L and R preserve adjacency of map elements. Since $w(R) = p$ which is a power

of 2 and since the underlying graph of a single column from any H_i is isomorphic to an n dimensional Hypercube, we have that the underlying graph of R is isomorphic to a Hypercube map of dimension $n + \log_2 p = nk - 1 = d_{k-1}$. Note that elements of R are from the H provided on the initial call and hence, may contain elements $\geq 2^{d_{k-1}}$.

Step 3: We can solve S_L by applying the "pattern" of the solution map for $G(2^n - 1, 2^{n+1})$ (solution developed in Lemma 7 — see Figure 6.3) using successive groups of $2^n - 1$ columns from L (see Example 9 — Figure 6.4). Since reflective edges exist about the vertical separators between these groups (positions $i(2^n - 1) + 1/2$, for $1 \leq i < p$), and since the pattern turns bottom and top rows into boundary columns, the column boundary between the pattern of $(H_{i(\mathrm{mod}p)} \cap L)$ and the pattern of $(H_{i+1(\mathrm{mod}p)} \cap L)$ are adjacent and hence the total pattern of L "S_L" preserves adjacency.

We have shown in Step 2 that R is a height 2^n map whose underlying graph is isomorphic to a Hypercube map of dimension $d_{k-1} = nk - 1 = n((k-1) + 1) - 1$. Therefore since R satisfies the input constraint for a base $k - 1$ grid, we can solve by inductive assumption for the next smaller base $k - 1$ grid "S_k" using R as the Partial Hypercube map.

Step 4a: By inductive assumption, f was defined when S_R was solved. Notice that column i in R is from H_i. Let f_f denote the column in H_f that is column f in R, and also define α_f and β_f to be the columns in H_f that are $2^{n-1} - 1$ and 2^{n-1} columns from f_f, respectively. Therefore we have:

$$\text{col } f_f \in H_f \equiv \text{col } f \in R \tag{6.1}$$

$$f_f = \begin{cases} 1, & f \text{ even} \\ 2^n, & f \text{ odd} \end{cases} \tag{6.2}$$

$$\alpha_f = \begin{cases} f_f + (2^{n-1} - 1) = 2^{n-1}, & f \text{ even} \\ f_f - (2^{n-1} - 1) = 2^{n-1} + 1, & f \text{ odd} \end{cases} \tag{6.3}$$

$$\beta_f = \begin{cases} 2^{n-1} + 1 = \alpha_f + 1, & f \text{ even} \\ 2^{n-1} = \alpha_f - 1, & f \text{ odd} \end{cases} \tag{6.4}$$

Or equivalently if we let C_i denote column i in H_f, we have:

$$H_f = \begin{cases} (C_1, \ldots, C_{\beta_f}, \overbrace{C_{\alpha_f}, \ldots, C_{f_f}}^{2^{n-1}}), & f \text{ odd} \\ (\underbrace{C_{f_f}, \ldots, C_{\alpha_f}}_{2^{n-1}}, C_{\beta_f}, \ldots, C_{2^n}), & f \text{ even} \end{cases} \tag{6.5}$$

By reflectivity and Eqn 6.5 we have:

$$\text{col } \alpha_f \in H_f \quad \text{is adjacent to} \quad \text{col } f_f \in H_f \tag{6.6}$$

Let S_f denote the solution map for $G(2^n - 1, 2^{n+1})$ using the $2^n - 1$ columns of

$$(H_f \cap L) \equiv \begin{cases} \text{cols}(H_f, 2, 2^n), & f \text{ even} \\ \text{cols}(H_f, 1, 2^n - 1), & f \text{ odd} \end{cases}$$

We note that by the definition of α_f and β_f in Eqn 6.3 and Eqn 6.4, column α_f always occurs where the diagonal illustrated in Example 8 passes through row $2^n - 1$ and column β_f occurs next to column α_f where the diagonal is at the bottom apex of the alternating diagonals (row 2^n).

This implies the solution map S_f preserves the top $2^n - 1$ elements of column α_f and β_f, since elements on and above the diagonal are unchanged.

Let α'_f and β'_f be the two columns in S_f such that:

$$\text{col } \alpha'_f \in S_f \equiv \text{top } 2^n - 1 \text{ elements of col } \alpha_f \in H_f \tag{6.7}$$

$$\text{col } \beta'_f \in S_f \equiv \text{top } 2^n - 1 \text{ elements of col } \beta_f \in H_f \tag{6.8}$$

Then we have:

$$\alpha'_f = \begin{cases} 2^n - 1, & f \text{ even} \\ 2^n + 2, & f \text{ odd} \end{cases} \tag{6.9}$$

$$\beta'_f = \begin{cases} 2^n = \alpha'_f + 1, & f \text{ even} \\ 2^n + 1 = \alpha'_f - 1, & f \text{ odd} \end{cases} \tag{6.10}$$

Let α'_L and β'_L be columns in S_L such that:

$$\text{col } \alpha'_L \in S_L \equiv \text{col } \alpha'_f \in S_f \tag{6.11}$$

$$\text{col } \beta'_L \in S_L \equiv \text{col } \beta'_f \in S_f \tag{6.12}$$

Since $w(S_f) = 2^{n+1}$, for $1 \leq f \leq p$, we have $\alpha'_L = (f-1)2^{n+1} + \alpha'_f$ and $\beta'_L = (f-1)2^{n+1} + \beta'_f$. Therefore:

$$\alpha'_L = \begin{cases} f2^{n+1} - 2^n - 1, & f \text{ even} \\ f2^{n+1} - 2^n + 2, & f \text{ odd} \end{cases} \tag{6.13}$$

$$\beta'_L = \begin{cases} f2^{n+1} - 2^n = \alpha'_L + 1, & f \text{ even} \\ f2^{n+1} - 2^n + 1 = \alpha'_L - 1, & f \text{ odd} \end{cases} \tag{6.14}$$

Recall, by transitivity on Eqn 6.11, Eqn 6.7, Eqn 6.6 and Eqn 6.1, we have that column $\alpha'_L \in S_L$ is adjacent to the top $2^n - 1$ elements of column $f \in R$. Therefore by inductive assumption, column $f \in R \equiv$ column $1 \in S_R$ and we have α'_L is adjacent to column $1 \in S_R$. Hence Step 4a is correct by letting $t \leftarrow \alpha'_L$.

Step 4b If we let C_i denote column i in S_L, by Eqn 6.14 we can write:

$$S_L = \begin{cases} (C_1, C_2, \ldots, C_{\alpha'_L}, \overbrace{C_{\beta'_L}, \ldots, C_{w(S_L)}}^{w(S_L)-\alpha'_L}) & f \text{ even} \\ (C_1, C_2, \ldots, C_{\beta'_L}, \underbrace{C_{\alpha'_L}, \ldots, C_{w(S_L)}}_{w(S_L)-\alpha'_L+1}) & f \text{ odd} \end{cases} \tag{6.15}$$

Notice that β'_L is the column on the right/left of α'_L when f is even/odd, respectively. This implies we can right circular shift S_L by an appropriate amount to obtain a map where α'_L and β'_L are at the opposite ends of the map.

Let $S'_L \leftarrow \text{shift}(S_L, s)$, where:

$$s = \begin{cases} w(S_L) - \alpha'_L, & f \text{ even} \\ w(S_L) - \alpha'_L + 1, & f \text{ odd} \end{cases} \tag{6.16}$$

By Eqn 6.15 and Eqn 6.16, we can write:

$$S'_L = \begin{cases} (C_{\beta'_L}, \ldots, C_{w(S_L)}, C_1, C_2, \ldots, C_{\alpha'_L}) & f \text{ even} \\ (C_{\alpha'_L}, \ldots, C_{w(S_L)}, C_1, C_2, \ldots, C_{\beta'_L}) & f \text{ odd} \end{cases} \tag{6.17}$$

Therefore by Eqn 6.16, Step 4b is correct.

Step 4c Let us define:

$$S''_L = \begin{cases} S'_L & f \text{ even} \\ \text{reverse}(S'_L) & f \text{ odd} \end{cases} \tag{6.18}$$

Then by Eqn 6.17 and Eqn 6.18, we can write:

$$S''_L = \begin{cases} (C_{\beta'_L}, \ldots, C_{w(S_L)}, C_1, C_2, \ldots, C_{\alpha'_L}) & f \text{ even} \\ (C_{\beta'_L}, \ldots, C_2, C_1, C_{w(S_L)}, \ldots, C_{\alpha'_L}) & f \text{ odd} \end{cases} \tag{6.19}$$

Therefore since column α'_L is adjacent to column $1 \in S_R$ (by Step 4a), Step 4c is correct by letting $S \leftarrow S''_L || S_R$.

Step 4d We now have a solution S, where column $\beta'_L \in S_L$ is the first column in S. Let β_H denote the column in H where $\beta_f \in H_f$ was obtained. Since $w(H_i) = 2^n$, we

have that $\beta_H = (f-1)2^n + \beta_f$ and therefore:

$$\beta_H = \begin{cases} f2^n - 2^{n-1} + 1, & f \text{ even} \\ f2^n - 2^{n-1}, & f \text{ odd} \end{cases} \tag{6.20}$$

Recall, by transitivity on Eqn 6.12 and Eqn 6.8, we have that column $\beta'_L \in S_L \equiv$ top $2^n - 1$ elements of $\beta_f \in H_f$. Therefore Step 4d satisfies the output invariant by letting $f \leftarrow \beta_H$. □

Lemma 9 *Grids of the form $G(h, w)$ can be embedded into the next smallest Hypercube for $h = 2^j - 2^k, 0 \leq k < j$ and $w > 2^{j-k}$.*

Proof: We know by Theorem 6 that we can solve for $h = 2^j - 1 = 2^j - 2^0 < w$. We also know that 2^k divides 2^j, for $k < j$. Therefore, we can solve for $h' = 2^{j-k} - 1$ and replicate the solution 2^k times — stacking these solutions vertically. We then have a solution for grids of height $h'' = 2^k h' = 2^k(2^{j-k} - 2^0) = 2^j - 2^k$. □

Chapter 7

Grids of Height $(2^n + 1)$

7.1 Introduction

We have shown how to embed grids of height $2^n - 1$ using a Hypercube map of height 2^n. We also can solve for all grids of height $2^n + 1$ via a similar approach. By extending an example given by Sudborough & Miller (see Figure 2.5), we observe a general pattern for height $2^n + 1$ grids by examining their height 9 braiding.

7.2 Examples

We shall illustrate a way of transforming the first 9 columns of a height 8 Hypercube map into a 9×16 map, having the properties:

1. Columns from the Hypercube map are used **greedily** (from left to right while minimizing the accumulated waste — elements from a column are used completely before elements from the column on its right). This guaranties that a

0	1	3	2	6	7	5	4	68	69	71	70	66	67	65	64
8	9	11	10	14	15	13	12	76	77	79	78	74	75	73	72
24	25	27	26	30	31	29	28	92	93	95	94	90	91	89	88
16	17	19	18	22	23	21	20	84	85	87	86	82	83	81	80
48	49	51	50	54	55	53	52	116	117	119	118	114	115	113	112
56	57	59	58	62	63	61	60	124	125	127	126	122	123	121	120
40	41	43	42	46	47	45	44	108	109	111	110	106	107	105	104
32	33	35	34	38	39	37	36	100	101	113	112	98	99	97	96

Figure 7.1: Initial 9 Columns of $HM(8, 16)$

sub-map (any number of contiguous columns starting from the leftmost column) will only be composed of columns from the next smaller Hypercube, since a next smaller Hypercube has dimension $d \geq n$ and hence can be expressed as a Hypercube map of height 2^n.

2. The first/last columns of $G(9, 16)$ are mapped to the first/last columns respectively of the 9 column sub-Hypercube map, and hence the pattern may be replicated adinfinitum, in a reflective manner.

The technique works by duplicating elements of the map column by column, then rearranging them to fit the height $2^n + 1$ grid. This is in contrast with previously illustrated techniques which first rearrange elements, then duplicate.

Figure 7.1 illustrates the initial 9 columns of a height 8 Hypercube map and Figure 7.2 illustrates the result of the duplication stage which forms an 8×18 map by duplicating each of the 9 columns in place. In Figure 7.3 we group pairs of columns that are next to each other, but not identical. The first and last column are not paired with another column since they are only next to their duplicate column.

0	0	1	1	3	3	2	2	6	6	7	7	5	5	4	4	68	68
8	8	9	9	11	11	10	10	14	14	15	15	13	13	12	12	76	76
24	24	25	25	27	27	26	26	30	30	31	31	29	29	28	28	92	92
16	16	17	17	19	19	18	18	22	22	23	23	21	21	20	20	84	84
48	48	49	49	51	51	50	50	54	54	55	55	53	53	52	52	116	116
56	56	57	57	59	59	58	58	62	62	63	63	61	61	60	60	124	124
40	40	41	41	43	43	42	42	46	46	47	47	45	45	44	44	108	108
32	32	33	33	35	35	34	34	38	38	39	39	37	37	36	36	100	100

Figure 7.2: $G(8, 18)$ into Initial Map

Figure 7.3 also illustrates the next operation which shifts the groups vertically. We note that after shifting (Figure 7.4), adjacent elements are still Hamming distance 1 apart since the boundary between groups is a boundary between original/duplicate columns, where the shift along a boundary is by one element only (i.e. any element on the right/left boundary of a group in Figure 7.3 is adjacent to the element immediately below/above and hence is adjacent to the element in the next group located to the right/left in the shifted result of Figure 7.4).

The top and bottom 2^{n-1} rows are then moved to the bottom and top respectively, as illustrated in Figure 7.4. This is allowed since reflective edges exist between the top and bottom element of any column. The result is illustrated in Figure 7.5 and may be viewed as two 8×9 maps placed side by side and vertically offset with respect to each other by one element.

The offset between the two maps can be removed by making the following observation — a shift of one diagonal element along any 45 degree diagonal seam ($\nearrow$) can be done while preserving the Hamming distance 1 property. This is valid since elements along these diagonals and their neighboring diagonals are duplicated in a

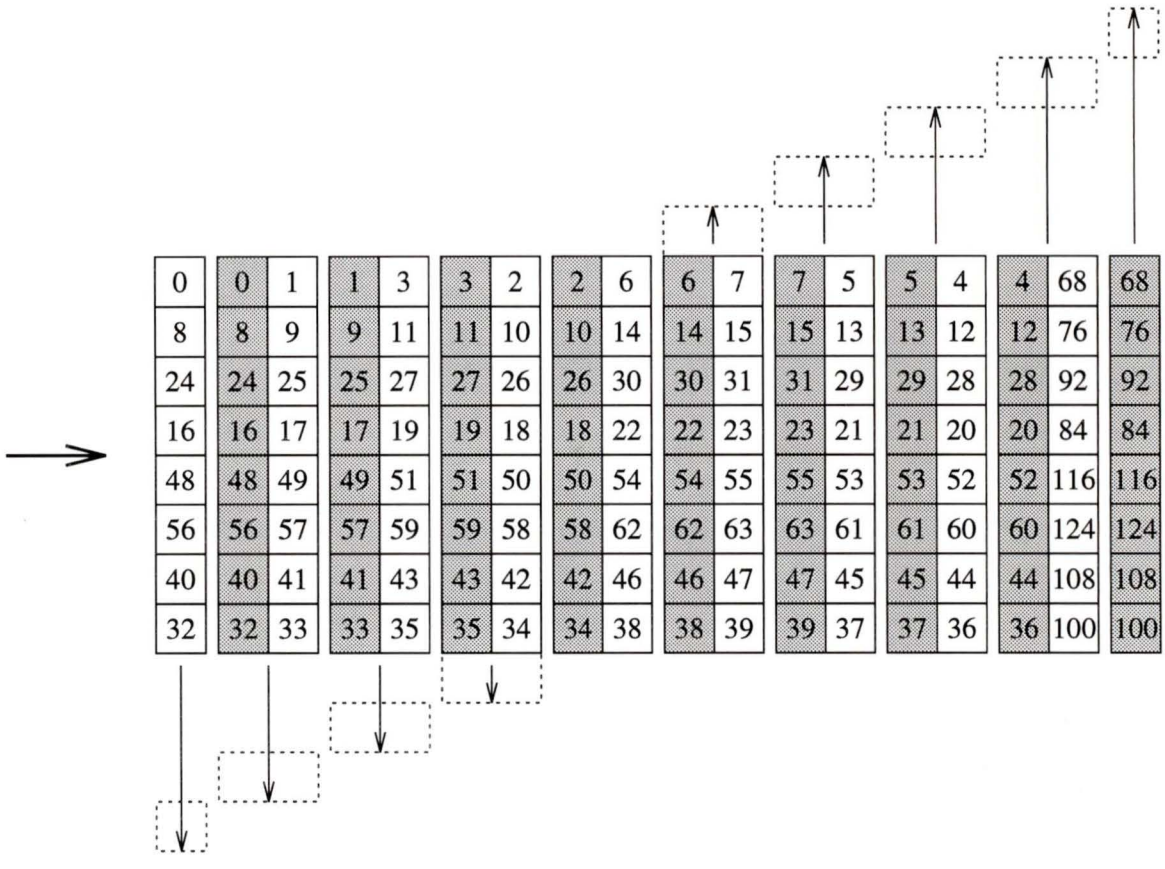


Figure 7.3: Grouping of $G(8, 18)$ and Illustrated Vertical Shift

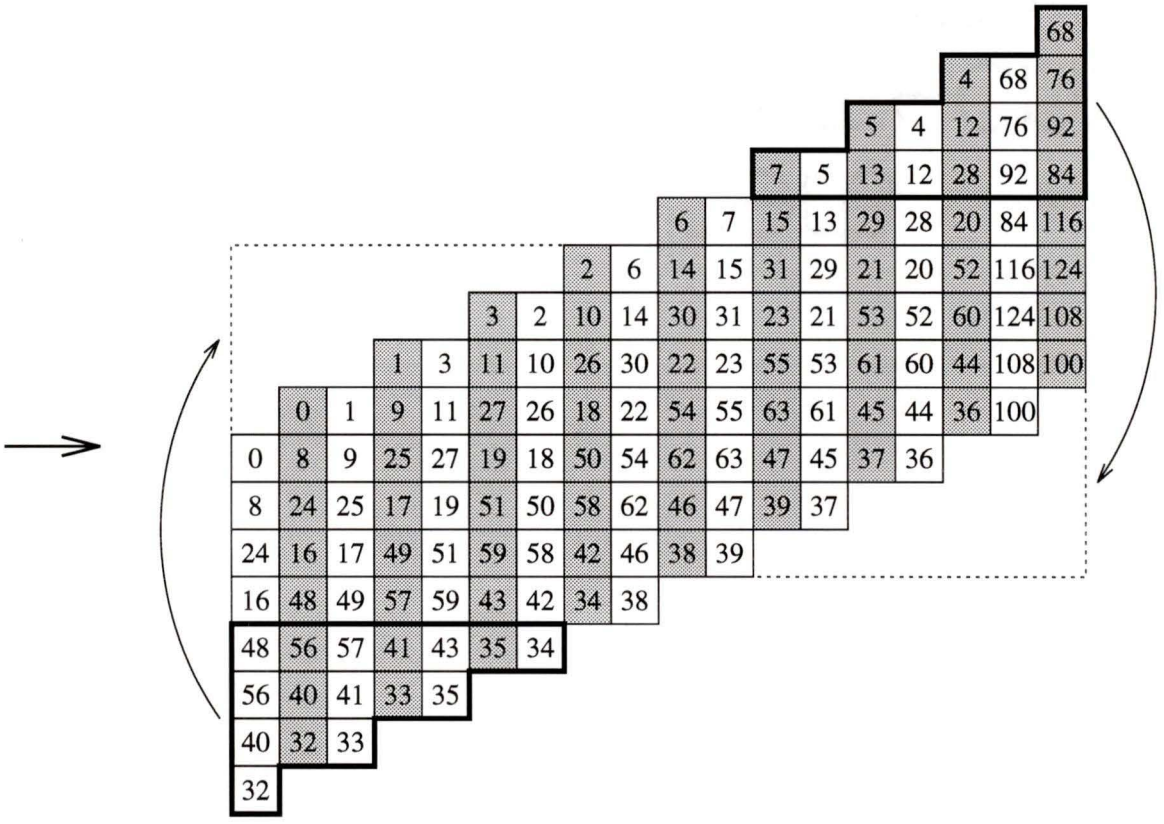


Figure 7.4: Staircase Pattern of $G(8, 18)$ and Illustrated Move

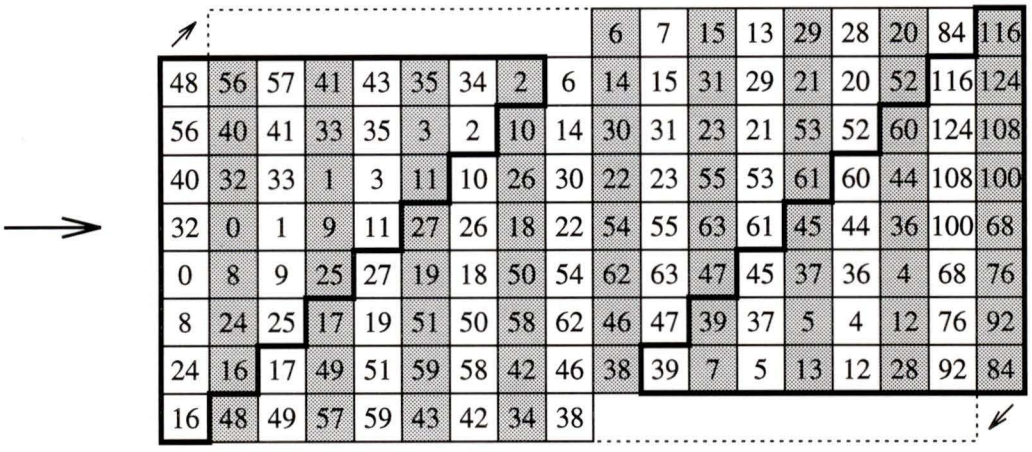


Figure 7.5: Shifted Two Grid Pattern

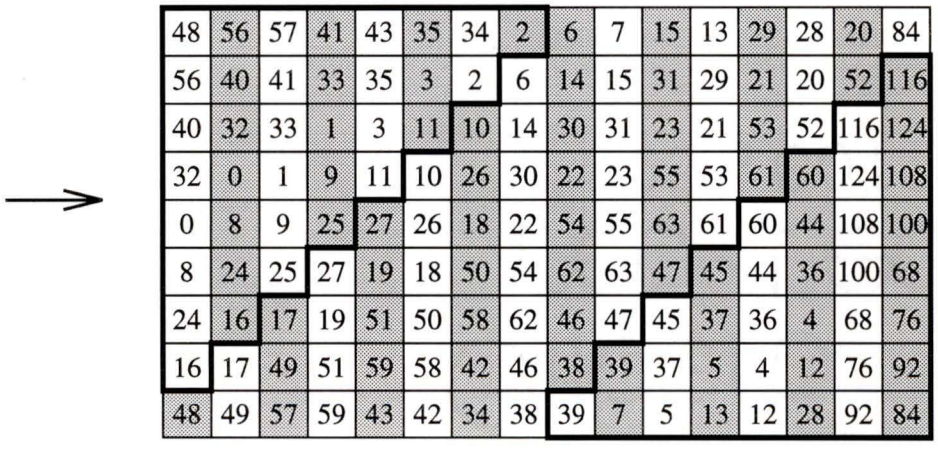


Figure 7.6: $G(9, 16)$ into $HM(8, 9)$ Embedding Pattern

way that allows it. We shift the diagonals as illustrated in Figure 7.5 and achieve the final result as required in Figure 7.6.

It should be clear from this example that we can apply this technique with any height 2^n Hypercube map and that we can apply this pattern similarly to successive groups of $2^n + 1$ columns where the shifts occur in alternately opposite directions.

7.3 Results

Theorem 7 *$G(2^n + 1, 2^{n+1})$ can be embedded using any $2^n + 1$ consecutive columns from a height 2^n Hypercube map.*

Proof: We note that any valid embedding produces no waste, since $|G(2^n+1, 2^{n+1})| = 2(2^n(2^n + 1))$ and hence it fully uses the $2^n + 1$ columns of the Hypercube.

To show that there exists a valid embedding we will show that the construction in Algorithm 7.7 solves for the embedding as required.

Step 1: Initially the column groupings as illustrated by Figure 7.3 are computed. There are $2^n + 2$ of these groups:

$$\begin{aligned} G_1 &= \text{cols}(H, 1, 1) \\ G_2 &= \text{cols}(H, 1, 2) \\ G_3 &= \text{cols}(H, 2, 3) \\ &\vdots \\ G_{2^n+1} &= \text{cols}(H, 2^n, 2^n + 1) \\ G_{2^n+2} &= \text{cols}(H, 2^n + 1, 2^n + 1) \end{aligned}$$

{Return a map for the largest grid of height $2^n + 1$ into the first $2^n + 1$ columns of H. H is a Hypercube map of height 2^n and width at least $2^n + 1$.}

Function EXPAND(H)

Step 1: **INITIALIZE**
Set G_1 $\leftarrow$ cols$(H, 1, 1)$
Set G_i $\leftarrow$ cols$(H, i-1, i)$, for $1 < i < 2^n + 2$
Set G_{2^n+2} $\leftarrow$ cols$(H, 2^n + 1, 2^n + 1)$
Set L $\leftarrow$ empty
Set R $\leftarrow$ empty

Step 2: **SOLVE**
For $i \leftarrow 1$ to $2^{n-1} + 1$ do
$L \leftarrow$ vshift$(L, 1)$ || G_i
$R \leftarrow G_{2^n+3-i}$ || vshift$(R, -1)$

Step 3: **FINALIZE**
Set $L' \leftarrow$ diag_shift$(L, (\nearrow))$
Set $R' \leftarrow$ diag_shift$(R, (\swarrow))$

Step 4: **RETURN**
Set $S \leftarrow (L' \,||\, R')$
Return (S)

Figure 7.7: Height $2^n + 1$ Embedding Algorithm

Step 2: Let us denote the leftmost/rightmost column of G_i as $\mathrm{LF}(G_i)$ and $\mathrm{RT}(G_i)$ respectively. We observe that for $1 \leq i \leq 2^n + 1$ and some $a, b, c \in \{1, 2, \ldots, 2^n + 1\}$ we have:

$$\begin{aligned} G_i &= \mathrm{cols}(H, a, b) \text{ and} \\ G_{i+1} &= \mathrm{cols}(H, b, c) \end{aligned}$$

Therefore we have:

$$\mathrm{RT}(G_i) = \mathrm{LF}(G_{i+1}) \tag{7.1}$$

For $1 \leq j \leq 2^n$, by the definition of a Hypercube map we have:

$$\mathrm{RT}(G_i)[j] \quad \mathrm{adj} \quad \mathrm{RT}(G_i)[1 + ((j-1) \pm 1) \bmod 2^n] \tag{7.2}$$

and by the definition of vshift() we have:

$$\mathrm{vshift}(\mathrm{RT}(G_i), \mp 1)[j] = \mathrm{RT}(G_i)[1 + ((j-1) \pm 1) \bmod 2^n] \tag{7.3}$$

Therefore by Eqn 7.3, Eqn 7.2 and Eqn 7.1, we have:

$$\mathrm{vshift}(\mathrm{RT}(G_i), \mp 1)[j] \quad \mathrm{adj} \quad \mathrm{LF}(G_{i+1})[j]$$

or equivalently:

$$\mathrm{vshift}(\mathrm{RT}(G_i), \pm 1) \quad \mathrm{adj} \quad \mathrm{LF}(G_{i+1}) \tag{7.4}$$

Similarly we can show:

$$\mathrm{RT}(G_i) \quad \mathrm{adj} \quad \mathrm{vshift}(\mathrm{LF}(G_{i+1}), \pm 1) \tag{7.5}$$

At the top of the **for loop** with $i > 1$, we have:

$$\mathrm{RT}(L) = \mathrm{RT}(G_{i-1}) \tag{7.6}$$

$$\mathrm{LF}(R) = \mathrm{LF}(G_{2^n+4-i}) \tag{7.7}$$

since on the previous iteration we concatenated G_i to the end of L and G_{2^n+4-i} to the beginning of R.

Therefore by applying Eqn 7.4 to Eqn 7.6 and Eqn 7.5 to Eqn 7.7, we have:

$$\text{RT}(\text{vshift}(L, 1)) \quad \text{adj} \quad \text{LF}(G_i) \tag{7.8}$$

$$\text{LF}(\text{vshift}(R, 1)) \quad \text{adj} \quad \text{RT}(G_{2^n+3-i}) \tag{7.9}$$

and hence L and R are formed with valid concatenation sequences.

Step 3: We show that a one element upwards ($\nearrow$) shift of a upper triangle along any diagonal is valid and that a one element downwards ($\swarrow$) shift of a lower triangle along any diagonal is valid. We can prove this by looking at all possible 3×3 regions of the map and verifying that the appropriate shift of the upper/lower triangle bounded along the main 45 degree diagonal is always possible within the region. We denote a region symbolically by:

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

By symmetry we need only prove for shifting of the upper triangle. To validate a shift upwards of the upper triangle we must show that g adj f.

By the construction of L and R in Step 2, for the $i + 1$ entry ($i \geq 0$) of the for loop we have:

$$\begin{aligned} L_0 &= \text{empty} \\ L_{i+1} &= \text{vshift}(L_i, 1) \,||\, G_{i+1} \end{aligned}$$

$$\begin{aligned} &= \text{vshift}(\text{vshift}(L_{i-1}, 1) \,||\, G_i, 1) \,||\, G_{i+1},\ i > 0 \\ &= \text{vshift}(L_{i-1}, 2) \,||\, \text{vshift}(G_i, 1) \,||\, G_{i+1} \end{aligned} \tag{7.10}$$

$$\begin{aligned} R_0 &= \text{empty} \\ R_{i+1} &= G_{2^n+2-i} \,||\, \text{vshift}(R_i, -1) \\ &= G_{2^n+2-i} \,||\, \text{vshift}(G_{2^n+3-i} \,||\, \text{vshift}(R_{i-1}, -1), -1),\ i > 0 \\ &= G_j \,||\, \text{vshift}(G_{j+1}, -1) \,||\, \text{vshift}(R_{i-1}, -2),\ j = 2^n + 2 - i \end{aligned} \tag{7.11}$$

Since the loop solves for $L = L_{2^{n-1}+1}$ and $R = R_{2^{n-1}+1}$, for $1 \leq i \leq 2^{n-1}$ and $j = 2^n + 2 - i$, we have:

$$\begin{aligned} \text{vshift}((\text{vshift}(G_i, 1) \,||\, G_{i+1}), i - 2^{n-1}) &\subset L \\ \text{vshift}((G_j \,||\, \text{vshift}(G_{j+1}, -1)), j - (2^{n-1} + 2)) &\subset R \end{aligned}$$

Equivalently, for the appropriate value of x_i and x_j, we can write:

$$\begin{aligned} \text{vshift}((G_i \,||\, \text{vshift}(G_{i+1}, -1)), x_i) &\subset L \\ \text{vshift}((G_j \,||\, \text{vshift}(G_{j+1}, -1)), x_j) &\subset R \end{aligned}$$

Therefore since

$$w(G_i) = \begin{cases} 1 & \text{for } i = 1 \text{ or } i = 2^n + 2 \\ 2 & \text{otherwise} \end{cases} \tag{7.12}$$

for any $M \subset L$ or any $M \subset R$, we have for the appropriate value of i and x

$$\begin{aligned} \alpha &\leftarrow \text{vshift}(G_i, x) \\ \beta &\leftarrow \text{vshift}(G_{i+1}, x) \\ M &\subset \alpha \,||\, \text{vshift}(\beta, -1) \end{aligned}$$

By Eqn 7.12 there are two ways we can form a 3×3 region:

$$\begin{bmatrix} a & b \\ d & e \\ g & h \end{bmatrix} \subset \alpha, \text{ and } \begin{bmatrix} c \\ f \\ i \end{bmatrix} \subset \text{vshift}(\beta, -1)$$

or

$$\begin{bmatrix} a \\ d \\ g \end{bmatrix} \subset \alpha, \text{ and } \begin{bmatrix} b & c \\ e & f \\ h & i \end{bmatrix} \subset \text{vshift}(\beta, -1)$$

By Eqn 7.1, Eqn 7.3 and the definition of α and β, we have

$$\begin{aligned} \text{RT}(\alpha)[j] &= \text{vshift}(\text{LF}(\beta), -1)[1 + ((j-1) - 1) \bmod 2^n] \\ &= \text{vshift}(\text{LF}(\beta), -1)[j - 1], \text{ for } 1 < j \leq 2^n \end{aligned}$$

Therefore the two cases can be written as:

$$\begin{bmatrix} a & b & e \\ d & e & h \\ g & h & i \end{bmatrix} \text{ or } \begin{bmatrix} a & b & c \\ b & e & f \\ e & h & i \end{bmatrix}$$

This is equivalent to g adj $h = f$ or $g = e$ adj f and therefore a shift upwards of a upper triangle is valid.

Step 4: Since L' and R' are $(2^n+1) \times 2^n$ maps, we can form the required $(2^n+1) \times 2^{n+1}$ map "$(L' \,||\, R')$" if and only if

$$\text{RT}(L') \quad \text{adj} \quad \text{LF}(R')$$

By Eqn 7.10 and Eqn 7.11, for some γ and δ, we have

$$L = \gamma \,||\, G_{2^{n-1}+1} \tag{7.13}$$

$$R = G_{2^{n-1}+2} \,||\, \delta \tag{7.14}$$

Therefore

$$\mathrm{RT}(L') = \mathrm{RT}((\nearrow)L) = \begin{bmatrix} G_{2^{n-1}+1}[1,1] \\ \mathrm{RT}(G_{2^{n-1}+1}) \end{bmatrix}$$

$$\mathrm{LF}(R') = \mathrm{LF}((\swarrow)R) = \begin{bmatrix} \mathrm{LF}(G_{2^{n-1}+2}) \\ G_{2^{n-1}+2}[2^n,2] \end{bmatrix}$$

By Eqn 7.1, Eqn 7.13 and Eqn 7.14

$$\mathrm{RT}(G_{2^{n-1}+1}) = \mathrm{RT}(L) = \mathrm{LF}(R) = \mathrm{LF}(G_{2^{n-1}+2})$$

Therefore by Eqn 7.4 and Eqn 7.5

$$\mathrm{LF}(R')[j] \quad \mathrm{adj} \quad \mathrm{vshift}(\mathrm{LF}(R'),+1)[j] = (\mathrm{RT}(L')[j], \text{ for } 1 < j \leq 2^n + 1)$$
$$\mathrm{RT}(L')[j] \quad \mathrm{adj} \quad \mathrm{vshift}(\mathrm{RT}(L'),-1)[j] = (\mathrm{LF}(R')[j], \text{ for } 1 \leq j < 2^n + 1)$$

Therefore

$$\mathrm{RT}(L')[j] \quad \mathrm{adj} \quad \mathrm{LF}(R')[j], \text{ for } 1 \leq j \leq 2^n + 1$$

and hence $L' \,||\, R'$ forms a valid $(2^n + 1) \times 2^n$ map. □

Lemma 10 *The embedding generated by Algorithm 7.7 maps columns of the grid to columns of the Hypercube map, using Hypercube map columns greedily from left to right.*

Proof: We show that the waste of any map formed by taking a contiguous set of columns (starting with column 1) from the map generated by Algorithm 7.7 will imply a greedy use of the Hypercube map columns.

We let

- $L(i)$ be the column index of H mapped to by column i of L.
- $R(i)$ be the column index of H mapped to by column i of R.
- $L'(i)$ be the set of column indices of H mapped to by column i of L'.
- $R'(i)$ be the set of column indices of H mapped to by column i of R'.
- $S(i)$ be the set of column indices of H mapped to by column i of $L' \,||\, R'$.
- $m(i)$ be the maximum value in $S(i)$.
- $w(i)$ be the total waste of the map composed of the first i columns of the solution map.

Since each column of the Hypercube map is capable of having 2^{n+1} grid vertices mapped to it, we need to show that

$$w(i) \;<\; 2^{n+1}, \text{ for } 1 \leq i \leq 2^{n+1} \tag{7.15}$$

By the construction of L and R, we have for $1 \leq i \leq 2^n + 1$

$$\begin{aligned} L(i) &= 1 + \lfloor (i-1)/2 \rfloor \\ R(i) &= 1 + \lceil (i-1)/2 \rceil + 2^{n-1} \end{aligned}$$

Therefore by the construction of L' and R', we have for $1 \leq i \leq 2^n$

$$\begin{aligned} L'(i) &= \{L(i), L(i+1)\} \\ &= \{1 + \lfloor (i-1)/2 \rfloor, 1 + \lfloor i/2 \rfloor\} \end{aligned}$$

$$
\begin{aligned}
&= \begin{cases} \{1 + \lfloor i/2 \rfloor\} & i \text{ odd} \\ \{\lfloor i/2 \rfloor, 1 + \lfloor i/2 \rfloor\} & i \text{ even} \end{cases} \\
R'(i) &= \{R(i), R(i+1)\} \\
&= \{1 + \lceil (i-1)/2 \rceil + 2^{n-1}, 1 + \lceil i/2 \rceil + 2^{n-1}\} \\
&= \begin{cases} \{1 + \lceil i/2 \rceil + 2^{n-1}\} & i \text{ even} \\ \{\lceil i/2 \rceil + 2^{n-1}, 1 + \lceil i/2 \rceil + 2^{n-1}\} & i \text{ odd} \end{cases}
\end{aligned}
$$

Hence for $1 \leq i \leq 2^n$

$$
\begin{aligned}
S(i) &= L'(i) \\
&= \begin{cases} \{1 + \lfloor i/2 \rfloor\} & i \text{ odd} \\ \{\lfloor i/2 \rfloor, 1 + \lfloor i/2 \rfloor\} & i \text{ even} \end{cases}
\end{aligned}
$$

and for $2^n < i \leq 2^{n+1}$

$$
\begin{aligned}
S(i) &= R'(i - 2^n) \\
&= \begin{cases} \{1 + \lceil (i-2^n)/2 \rceil + 2^{n-1}\} & i \text{ even} \\ \{\lceil (i-2^n)/2 \rceil + 2^{n-1}, 1 + \lceil (i-2^n)/2 \rceil + 2^{n-1}\} & i \text{ odd} \end{cases} \\
&= \begin{cases} \{1 + \lceil i/2 \rceil\} & i \text{ even} \\ \{\lceil i/2 \rceil, 1 + \lceil i/2 \rceil\} & i \text{ odd} \end{cases}
\end{aligned}
$$

Therefore we have

$$
m(i) = \begin{cases} 1 + \lfloor i/2 \rfloor & \text{for } 1 \leq i \leq 2^n \\ 1 + \lceil i/2 \rceil & \text{for } 2^n < i \leq 2^{n+1} \end{cases}
$$

We can solve for

$$
\begin{aligned}
w(i) &= 2^{n+1} m(i) - (2^n + 1) i \\
&= 2^{n+1} m(i) - 2^n i - i \\
&= 2^{n+1} m(i) - 2^n (\lceil i/2 \rceil + \lfloor i/2 \rfloor) - i
\end{aligned}
$$

{let H_i be H grouped into width $2^n + 1$ pieces}

Let H_i denote "cols$(H, (i-1)2^n + i, i(2^n + 1))$", for $1 \leq i \leq m$

$$\text{Set } S_i \leftarrow \begin{cases} \text{EXPAND}(H_i) & i \text{ odd} \\ \text{reverse}(\text{EXPAND}(\text{reverse}(H_i))) & i \text{ even} \end{cases}$$

$$\text{Set } S \leftarrow S_1 \,||\, S_2 \,||\, \ldots \,||\, S_m$$

Figure 7.8: Height $2^n + 1$ Embedding Algorithm (B)

and by substituting the unknowns, for $1 \leq i \leq 2^n$ we get

$$\begin{aligned} w(i) &= 2^{n+1}(1 + \lfloor i/2 \rfloor) - 2^n(\lceil i/2 \rceil + \lfloor i/2 \rfloor) - i \\ &= 2^{n+1} - i - 2^n(\lceil i/2 \rceil - \lfloor i/2 \rfloor) \\ &= 2^{n+1} - i - \begin{cases} 0 & i \text{ even} \\ 2^n & i \text{ odd} \end{cases} \end{aligned}$$

and for $2^n < i \leq 2^{n+1}$ we get

$$\begin{aligned} w(i) &= 2^{n+1}(1 + \lceil i/2 \rceil) - 2^n(\lceil i/2 \rceil + \lfloor i/2 \rfloor) - i \\ &= 2^{n+1} - i + 2^n(\lceil i/2 \rceil - \lfloor i/2 \rfloor) \\ &= 2^{n+1} - i + \begin{cases} 0 & i \text{ even} \\ 2^n & i \text{ odd} \end{cases} \end{aligned}$$

Therefore $w(i)$ satisfies Eqn 7.15. □

Corollary 3 *$G(2^n + 1, 2^{n+1}m)$, for any integer $m \geq 1$ can be embedded into the next smaller Hypercube using any $(2^n + 1)m$ consecutive columns from a height 2^n*

Hypercube map, by applying Algorithm 7.7 in a reflective way (alternating the shifts) to successive groupings of $2^n + 1$ columns from the height 2^n Hypercube map.

Proof: We claim that Algorithm 7.8 solves the problem. To verify, we must show that

$$\mathrm{RT}(S_i) \quad \text{adj} \quad \mathrm{LF}(S_{i+1}), \text{ for } 1 \leq i < m \tag{7.16}$$

By Eqn 7.10 and Eqn 7.11, for some γ and δ, we have

$$L = \mathrm{vshift}(G_1, 2^{n-1}) \,||\, \mathrm{vshift}(G_2, 2^{n-1} - 1) \,||\, \gamma \tag{7.17}$$

$$R = \delta \,||\, \mathrm{vshift}(G_{2^n+1}, 1 - 2^{n-1}) \,||\, \mathrm{vshift}(G_{2^n+2}, -2^{n-1}) \tag{7.18}$$

By Eqn 7.1 and Eqn 7.12

$$\begin{aligned} G_1 &= \mathrm{LF}(G_1) = \mathrm{RT}(G_1) = \mathrm{LF}(G_2) \\ G_{2^n+2} &= \mathrm{RT}(G_{2^n+2}) = \mathrm{LF}(G_{2^n+2}) = \mathrm{RT}(G_{2^n+1}) \end{aligned}$$

Recall by Alg 7.7

$$\begin{aligned} \mathrm{LF}(S) &= \mathrm{LF}(L') = \mathrm{LF}((\nearrow)L) \\ \mathrm{RT}(S) &= \mathrm{RT}(R') = \mathrm{RT}((\swarrow)R) \end{aligned}$$

Therefore by Eqn 7.17 and Eqn 7.18

$$\mathrm{LF}(S) = \begin{bmatrix} \mathrm{vshift}(G_1, 2^{n-1}) \\ \mathrm{vshift}(G_1, 2^{n-1} - 1)[2^n] \end{bmatrix} = \begin{bmatrix} \mathrm{vshift}(G_1, 2^{n-1}) \\ G_1[1 + 2^{n-1}] \end{bmatrix}$$

$$\mathrm{RT}(S) = \begin{bmatrix} \mathrm{vshift}(G_{2^n+2}, 1 - 2^{n-1})[1] \\ \mathrm{vshift}(G_{2^n+2}, -2^{n-1}) \end{bmatrix} = \begin{bmatrix} G_{2^n+2}[2^{n-1}] \\ \mathrm{vshift}(G_{2^n+2}, 2^{n-1}) \end{bmatrix}$$

Note that in the following we superscript the variables of Alg 7.7 with the value of i used when Alg 7.8 calls Alg 7.7.

Therefore

$$\mathrm{LF}(S^i) = \begin{bmatrix} \mathrm{vshift}(G_1^i, 2^{n-1}) \\ G_1^i[1 + 2^{n-1}] \end{bmatrix}$$

$$\mathrm{RT}(S^i) = \begin{bmatrix} G_{2^n+2}^i[2^{n-1}] \\ \mathrm{vshift}(G_{2^n+2}^i, 2^{n-1}) \end{bmatrix}$$

By definition of reverse, we have

$$\begin{aligned} \mathrm{LF}(S) &= \mathrm{RT}(\mathrm{reverse}(S)) \\ \mathrm{RT}(S) &= \mathrm{LF}(\mathrm{reverse}(S)) \end{aligned}$$

Therefore, by Alg 7.8

$$G_1^i = \begin{cases} \mathrm{LF}(H_i) & i \text{ odd} \\ \mathrm{LF}(\mathrm{reverse}(H_i)) = \mathrm{RT}(H_i) & i \text{ even} \end{cases}$$

$$G_{2^n+2}^i = \begin{cases} \mathrm{RT}(H_i) & i \text{ odd} \\ \mathrm{RT}(\mathrm{reverse}(H_i)) = \mathrm{LF}(H_i) & i \text{ even} \end{cases}$$

Therefore, for i odd

$$\mathrm{LF}(S_i) = \mathrm{LF}(S^i) = \begin{bmatrix} \mathrm{vshift}(\mathrm{LF}(H_i), 2^{n-1}) \\ \mathrm{LF}(H_i)[1 + 2^{n-1}] \end{bmatrix}$$

$$\mathrm{RT}(S_i) = \mathrm{RT}(S^i) = \begin{bmatrix} \mathrm{RT}(H_i)[2^{n-1}] \\ \mathrm{vshift}(\mathrm{RT}(H_i), 2^{n-1}) \end{bmatrix}$$

and, for i even

$$\mathrm{LF}(S_i) = \mathrm{RT}(S^i) = \begin{bmatrix} \mathrm{LF}(H_i)[2^{n-1}] \\ \mathrm{vshift}(\mathrm{LF}(H_i), 2^{n-1}) \end{bmatrix}$$

$$\mathrm{RT}(S_i) = \mathrm{LF}(S^i) = \begin{bmatrix} \mathrm{vshift}(\mathrm{RT}(H_i), 2^{n-1}) \\ \mathrm{RT}(H_i)[1 + 2^{n-1}] \end{bmatrix}$$

Also, by Alg 7.8

$$\mathrm{RT}(H_i) \quad \text{adj} \quad \mathrm{LF}(H_{i+1}), \text{ for } 1 \leq i < m$$

Therefore, for i odd

$$\mathrm{RT}(S_i) = \begin{bmatrix} \mathrm{RT}(H_i)[2^{n-1}] \\ \mathrm{vshift}(\mathrm{RT}(H_i), 2^{n-1}) \end{bmatrix} \quad \text{adj} \quad \begin{bmatrix} \mathrm{LF}(H_{i+1})[2^{n-1}] \\ \mathrm{vshift}(\mathrm{LF}(H_{i+1}), 2^{n-1}) \end{bmatrix} = \mathrm{LF}(S_{i+1})$$

and, for i even

$$\mathrm{RT}(S_i) = \begin{bmatrix} \mathrm{vshift}(\mathrm{RT}(H_i), 2^{n-1}) \\ \mathrm{RT}(H_i)[1 + 2^{n-1}] \end{bmatrix} \quad \text{adj} \quad \begin{bmatrix} \mathrm{vshift}(\mathrm{LF}(H_{i+1}), 2^{n-1}) \\ \mathrm{LF}(H_{i+1})[1 + 2^{n-1}] \end{bmatrix} = \mathrm{LF}(S_{i+1})$$

Therefore, Eqn 7.16 is satisfied. □

Theorem 8 *All grids of height* $(2^n + 1)$ *can be embedded into the next smaller Hypercube.*

Proof: Let the grid be $G(2^n + 1, w)$ and pick any m such that $w \leq m2^{n+1}$. By Corollary 3 we can create an embedding for grids of width $m2^{n+1}$ using Algorithm 7.8. By Lemma 10 we know the Hypercube map columns are used greedily in the solution, so we only need to use the first w columns of the solution to solve for a width w grid. □

Chapter 8

Summary

8.1 Conclusions

We have focused on two-to-one embeddings of Grids into the next smaller Hypercube (the smallest Hypercube under load factor 2) and have derived dilation 1 embedding techniques for the following classes of Grids:

$$G(h,w),\ \text{where}\ \begin{cases} h = w & \text{or} \\ h = 2^j - 2^k, w > 2^{j-k} & \text{for } 0 \le k < j, \text{ or} \\ h = b - a, w = b + a & \text{for } 0 \le a < \sqrt{b^2 - 2^{\lceil 2\log_2 b \rceil - 1}}, \text{ or} \\ h = 2^n - a, w = 2^n + a & \text{for } 0 \le a < 2^n/\sqrt{2} \\ h = 2^n + 1 & \text{for any width } w \end{cases}$$

We can also solve for any Grid smaller than (height and width less than or equal to) a Grid in the classification above, but whose next smaller Hypercube is the same dimension.

8.2 Unresolved Questions and Conjectures

The following are unresolved:

- We had observed that even when embedding Grids into Grids, solutions usually existed when the embedding did not require use of most of the Host (i.e. when $2|G| << |H|$). We are aware of no characterizations of when this is or is not possible, other than the simple case where the Guest is a subgraph of the Host. Under what conditions will there exist a dilation 1, two-to-one embedding of a Grid into a Grid?

- We found that with small (tractable) problems, computer searching would always find solutions for two-to-one, dilation 1 embeddings. Are all Grids dilation 1, two-to-one embeddable into the next smaller Hypercube?

- Although we have not investigated many-to-one embeddings using the techniques we applied on two-to-one embeddings, it seems likely that these techniques could be used to improve upon the dilation of current many-to-one embedding results. Under load factor n, is a dilation 1 embedding into the smallest Hypercube always possible?

From our computer searching results and our results from manipulating vertex mappings to solve for embeddings, we conjecture:

- All Grids are dilation 1, two-to-one embeddable into the next smaller Hypercube.

- All Grids are dilation 1, many-to-one embeddable into the smallest Hypercube.

Bibliography

[1] R. Aleliunas and A. L. Rosenberg. On embedding rectangular grids in square grids. *IEEE Transactions on Computers*, C–31(9):907–913, 1982.

[2] Said Bettayeb, Zevi Miller, and I. Hal Sudborough. Embedding grids into hypercubes. *Journal of Computer and System Sciences*, (45):340–366, 1992.

[3] Sandeep Bhatt, Fan Chung, Jia-Wei Hong, Thompson Leighton, and Arnold Rosenberg. Optimal simulations by butterfly networks (preliminary version). In *Proceedings of the* 20^{th} *Annual ACM Symposium on Theory of Computing*, number 20, pages 192–204, May 1988.

[4] Dan Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory*, Series B 49:103–136, 1990.

[5] M. Y. Chan. Dilation-2 embeddings of grids into hypercubes. Technical Report UTDCS 1–88, 1988.

[6] M. Y. Chan. Embedding of 3-dimensional grids into optimal hypercubes. Technical Report UTDCS 13–88, 1988.

[7] M. Y. Chan. Embedding of d-dimensional grids into optimal hypercubes. Technical Report UTDCS 21–88, 1988.

[8] M. Y. Chan and F. Y. L. Chin. On embedding rectangular grids in hypercubes. *IEEE Transactions on Computers*, 37(10), October 1988.

[9] John Ellis. Embedding grids into grids : Techniques for large compression ratios. Technical Report DCS–154–IR, University of Victoria, 1991.

[10] John Ellis. Embedding rectangular grids into square grids. *IEEE Transactions on Computers*, 40(1), January 1991.

[11] Michael Fellows. *Encoding Graphs in Graphs.* Ph.d. dissertation, University of California at San Diego, 1985.

[12] David S. Greenberg, Lenwood S. Heath, and Arnold L. Rosenberg. Optimal embeddings of butterfly-like graphs in the hypercube. *Math. Systems Theory*, (23):61–77, 1990.

[13] Jukka Helen. Performance analysis of the CM-2, a massively parallel SIMD computer. *CONCURRENCY:PRACTICE AND EXPERIENCE*, 5(1):71–85, 1993.

[14] D.S. Scott J. E. Brandengurg. Embeddings of communication trees and grids into hypercubes. Intel Scientif. Comput. Scientif. Rep. 280182-001, Intel, 1985.

[15] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures:ARRAYS-TREES-HYPERCUBES.* Morgan Kaufmann Publishers, 1992.

[16] Tom Leighton, Bruce Maggs, and Arnold Rosenberg. Diamacs workshop proposal: Graph embeddings and parallel architecture. Letter, January 1992.

[17] Y. E. Ma and D. G. Shea. The embedding kernel on the IBM Victor Multiprocessor for program mapping. 2^{nd} *IEEE Symposium on Parallel and Distributed Processing*, (2), 1990.

[18] Rami G. Melhem and Ghil-Young Hwang. Embedding rectangular grids into square grids with dilation two. *IEEE Transactions on Computers*, 39(12), 1990.

[19] Z. Miller and H. Sudborough. Compressing grids into small hypercubes. Submitted for publication.

[20] B. Monien and H. Sudborough. Comparing interconnection networks. *Reihe Informatik*, (55), October 1988.

[21] Christos H. Papadimitriou. The np-completeness of the bandwidth minimization problem. *Computing*, (16):263–270, 1976.

[22] Arnold L. Rosenberg. Issues in the study of graph embeddings. Number 100 in Lecture Notes in Computer Science, pages 150–176. Springer Verlag, 1981.

[23] Arnold L. Rosenberg. Graph embeddings 1988 : Recent breakthroughs, new directions. COINS Technical Report 88–28, 1988.

VITA

Surname: **Manke** Given Names: **Dennis Lorne**
Place of Birth: **Chemainus, B.C., Canada** Date of Birth: **June, 30th 1963**

Educational Institutions Attended:

University of Victoria	1982 to 1989
University of Victoria	1990 to 1993

Degrees Awarded:

B.Sc.	1989	University of Victoria, Victoria, B.C., Canada

Partial Copyright License

Title of Thesis:

2 to 1 Embeddings of Grids into Hypercubes

Author: [redacted]
Dennis L. Manke
September 23, 1993