Many to One Embeddings from Grids into Cylinders, Tori and Hypercubes *

John Ellis, Stirling Chow and Dennis Manke

Department of Computer Science, University of Victoria

P.O. Box 3055, Victoria, British Columbia, V8W 3P6, Canada

jellis@csr.uvic.ca

September 3, 2002

Abstract

We describe novel methods for embedding 2-dimensional grid graphs into cylinders (one way wrap-around grids), toruses and hypercubes where the guest grid G is larger than the host graph H, implying a many to one embedding.

We call $\lceil \mid G \mid / \mid H \mid \rceil$ the optimal load, denoted l. We consider optimal embeddings with respect to dilation (the stretching of guest edges) and load, i.e., edges are mapped to edges or onto one node and the number of grid nodes mapped onto any hypercube node is not greater than l.

We show, by construction, that for loads of at least four optimal embeddings into the hypercube always exist subject only to modest restrictions on the relative dimensions of guest and host. If the problem instances are grouped by grid height, the restrictions imply only that no more than some finite number of instances in each group may not be solvable by the given methods.

The essence of the method is a mapping from grids into cylinders of height at least one half of but not greater than the grid height, and so it can also be used to construct embeddings into cylinders and toruses.

Previous work has gone so far as to show that if the optimal load l is a power of 2, then dilation 1, load $l+1$, embeddings into the hypercube can be constructed. Optimal results for loads 2 and 3 are not known.

Keywords: graph embedding, many to one, grid, torus, hypercube.

*This work was supported by the Natural Sciences and Engineering Research Council of Canada

1 Introduction

The study of embeddings from one family of graphs to another has received substantial attention over many years. Besides the inherent mathematical interest of these problems, motivation for the study comes from the many concerns of computer science that can be effectively modelled by this graph theoretical abstraction. In this paper we describe some results on many to one embeddings from 2-dimensional grids into smaller cylinders, tori and hypercubes. Plausible applications for our results include the simulation of large grid-like networks of processors by smaller networks in the form of cylinders, tori or hypercubes, and the efficient implementation of a parallel algorithm conceived of as running on a grid of unlimited size, but executed on some fixed sized network in those same families. In these applications the embedding parameters *load*, *dilation* and *congestion*, defined in the next section, strongly influence the effectiveness of the simulation.

Several papers have discussed grid to grid, and grid to hypercube embeddings. [Ell91, Ell96, MH90, SHSHV96] describe one to one embeddings from 2-dimensional grids into other 2-dimensional grids of smaller aspect ratio, with dilation 2. Likewise [RS01] considers efficient embeddings from grids into grids.

One to one embeddings from 2-dimensional grids into hypercubes are considered in [RS98], from 3-dimensional grids into hypercubes in [CCCM96] and [RSU94].

It is well known, see for example [Lei92] and Lemma 3.1 in Section 3, that not all 2-dimensional grids are subgraphs of the minimum sized hypercube, sufficient to contain the grid. In [Cha91] it is shown that all 2-dimensional grids can be one to one embedded into the smallest possible hypercube with dilation two. This result can also be obtained by an entirely different route, namely by using the grid to grid embeddings just cited to reduce the width of the grid down to the next power of two, i.e., embed the guest grid into a grid that is a subgraph of the hypercube. This is the approach taken in [BMS92].

What happens when the guest grid is arbitrarily larger than the host graph, i.e., one considers many to one embeddings, is by no means completely understood. Do dilation one embeddings with an optimal load always exist, at least into the hypercube if not for the simpler families?

Many to one embeddings from 2-dimensional grids into smaller 2-dimensional grids is considered in [SS90]. There it is shown that dilation one with optimal load plus one is always obtainable, so long as the host is smaller than the guest in both dimensions. In [MS94] many to one embeddings from both 2 and 3-dimensional grids into hypercubes are considered. For 2-dimensional grids, the main result (Theorem 1) states (and we paraphrase): *If the optimal load is 2^k for some integer k, then an embedding exists with dilation one and load $2^k + 1$.*

Our main result shows that dilation one, optimal load solutions can always be constructed for embeddings from the 2-dimensional grid into the hypercube for all loads greater than three, with only a modest restriction on the relative dimensions of the guest and host. The restriction may be interpreted as saying that, in each group of problem instances of a given grid height, only a finite number might not be solved by our methods.

The technique is based on a mapping into the torus, so similar results also apply to the torus, under some restrictions on the relation between grid and torus dimensions.

For cylinders the absence of horizontal wrap-around edges seems to prevent the technique from achieving optimal load in those instances where the host is only just big enough to contain the guest.

We leave open the conjecture that instances of the grid to hypercube embedding problem always have optimal load and dilation one solutions for all loads ≥ 2 and irrespective of the relative dimensions of the guest and host.

2 Definitions

A *graph embedding* is a mapping from the nodes of the guest graph to the nodes of the host graph, together with a mapping from guest edges to paths in the host. In general the node mapping can be one to one or many to one, but here we are concerned only with many to one embeddings. The *dilation* of a guest edge is the length of the path which is the image of that edge in the host. Because, in a many to one mapping, nodes incident to an edge could be mapped to the same host node, it is possible that the dilation of an edge is zero. The dilation of an entire embedding is the largest dilation over all edges in the guest.

Here we consider only instances where the number of nodes in the guest graph, denoted $\mid G \mid$, is greater than the number of nodes in the host, $\mid H \mid$. For any many to one mapping, the maximum over all host nodes of the number of guest nodes mapped to the host node is called the *load* of the mapping. The ratio $\lceil \mid G \mid / \mid H \mid \rceil$, denoted l, we call the *optimal load.* Throughout the paper we assume that $l \geq 2$.

The *congestion* of an embedding is the maximum over all host edges of the number of guest edges mapped to a path that includes the host edge.

We are only interested in what we call *optimal* embeddings which are those with dilation one and for which no more than l guest nodes are mapped onto any host node. Restricting the dilation to one means that we consider only embeddings that map edges to edges or map both ends of a guest edge onto the same host node. Hence we have no need in this presentation to explicitly define the guest edge to host path mapping.

The family of guest graphs which we consider is the 2-dimensional grids. A 2-dimensional grid of height h and width w is the graph comprising the node set: $\{(x, y) \mid 0 \leq x < h, 0 \leq y < w\}$ and the edge set: $\{\{(u, v), (x, y)\} \mid \mid u - x \mid + \mid v - y \mid = 1\}$. We will refer to these guest graphs as *(h,w)-grids.*

The families of host graphs considered are:

- Cylinders, which we picture as (h, w)-grids with wrap-around edges in the vertical dimension, are the graphs comprising the node set: $\{(x, y) \mid 0 \leq x < h, 0 \leq y < w\}$ and the edge set:
  $\{\{(u, v), (x, y)\} \mid \mid u - x \mid + \mid v - y \mid = 1\} \bigcup \{(0, v), (h - 1, v)\}$. We will refer to these host graphs as *(h,w)-cylinders.*

- Tori, which we picture as (h, w)-grids with wrap-around edges in both dimensions, are the graphs comprising the node set: $\{(x, y) \mid 0 \leq x < h, 0 \leq y < w\}$ and the edge set: $\{\{(u, v), (x, y)\} \mid \mid u - x \mid + \mid v - y \mid = 1\} \bigcup \{(0, v), (h-1, v)\} \bigcup \{(u, 0), (u, w-1)\}$. We will refer to these host graphs as *(h,w)-tori.*

- Hypercubes of dimension d, denoted Q_d, which are the graphs with vertex set: $\{n \mid 0 \leq n \leq 2^n - 1\}$ and edge set: $\{\{x, y\} \mid$ the binary representations of x and y differ in exactly one bit position$\}$.

3 An Example

Figure 1 defines our notation for naming the nodes of the guest and host graphs. We begin with a small example by way of introducing both our notation and our method. In this example the host is a cylinder.

It is well known, see for example [Lei92], that not all 2-dimensional grids G are subgraphs of H, where H is the smallest hypercube such that $|G| \leq |H|$. The distinction between those that are and are not is defined in the following lemma.

Lemma 3.1 *The (h, w)-grid is a subgraph of Q_d iff $\lceil \log_2 h \rceil + \lceil \log_2 w \rceil \leq d$*

So, for example, the $(5, 6)$-grid is not a subgraph of Q_5. We also use another well known property of the hypercube Q_d, again see [Lei92].

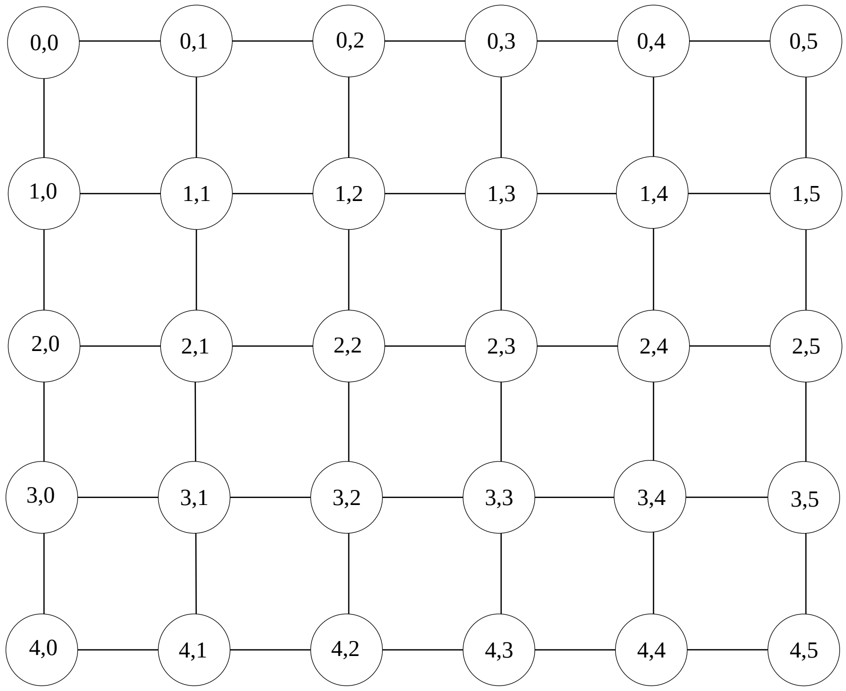
Lemma 3.2 *If $p + q \leq d$ then the $(2^p, 2^q)$-torus is a subgraph of Q_d.*

Consequently, the $(4, 8)$-cylinder is a subgraph of Q_5. Hence there is no dilation one, load one embedding from the $(5, 6)$-grid into the $(4, 8)$-cylinder because, if such an embedding did exist, it would imply the existence of a dilation one embedding from the (5,6)-grid into Q_5, contradicting Lemma 3.1.

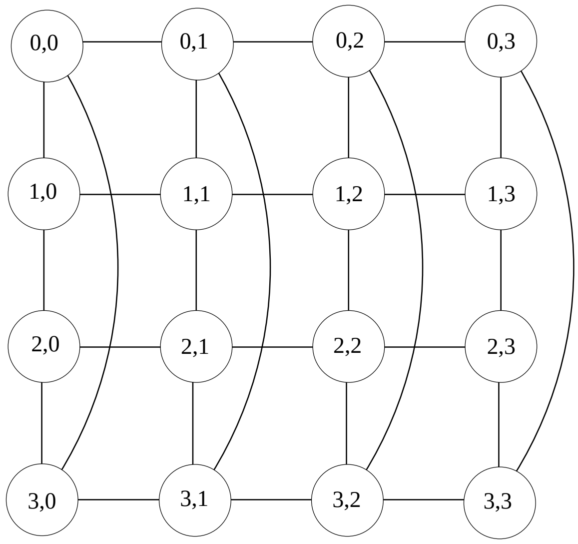
However, there is a load two, dilation one embedding into the $(4, 4)$-cylinder. A solution, illustrated in Figure 2, demonstrates the way we will illustrate all such embeddings. If the guest is an (h, w)-grid the diagram shows an $h \times w$ array of integers. Let the top leftmost element in the array shown in the figure be indexed (0,0). Then any element (i, j) in the array represents a node (i, j) in the guest grid, as illustrated in Figure 1. The value of array element (i, j) defines which host row is to be the image of grid node (i, j). The host column number is defined by the numbers beneath the bottom row, where adjacent host columns are distinguished by the presence or absence of shading. For example, Figure 2 tells us that node (4,3) of the guest is mapped to row 2, column 1 of the host.

Remembering that there exist wrap-around edges between rows 0 and 3 in the host cylinder, one can verify that the diagram defines a legitimate load two embedding in which edges are mapped to edges by noticing that:

- Each host node appears no more than twice



a) A (5,6)-grid



b) A (4,4)-cylinder

Figure 1: Node naming conventions

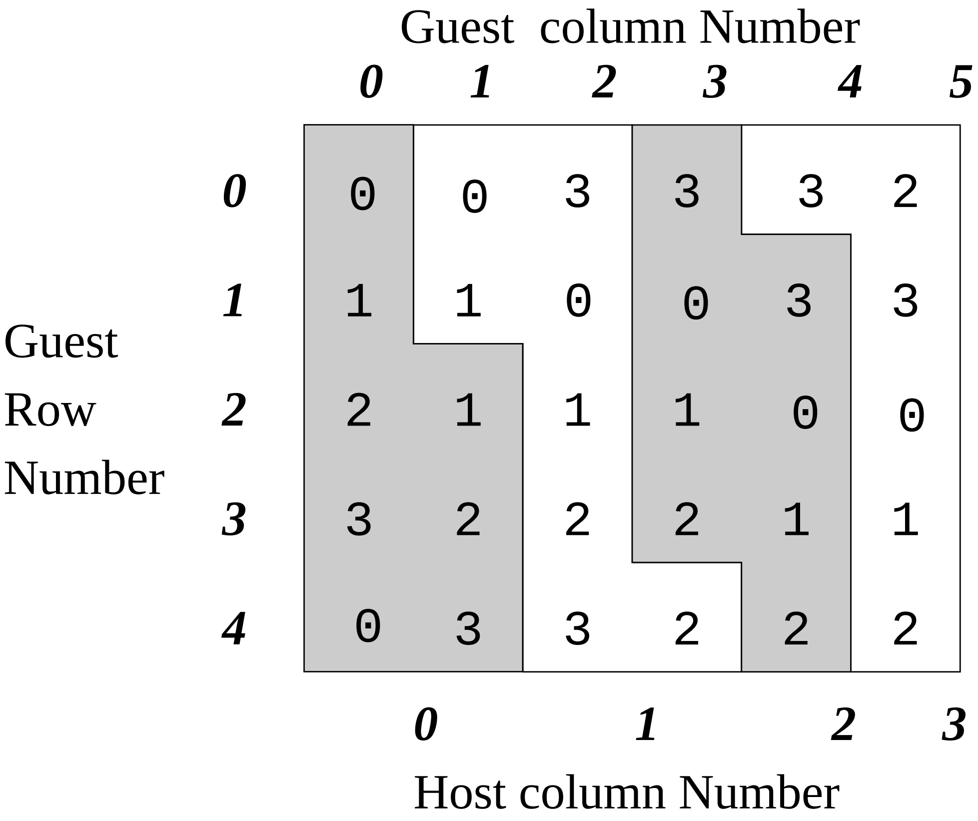


Figure 2: Embedding a (5,6)-grid into a (4,4)-cylinder with load 2

- Elements in the array that are vertically or horizontally adjacent, i.e., they represent adjacent guest nodes, are either designated adjacent row numbers in the same host column, or identical row numbers in adjacent host columns.

Our way of visualising embeddings is an alternative to that often used in previous work, where one draws the host and indicates for each host node, which guest nodes are mapped onto it. It is interesting to note that the solutions suggested by these different views are quite distinct and that the pattern underlying one type of solution may not be at all apparent in the alternative visualisation.

4 The Partial Mapping

4.1 Informal Description of the Partial Mapping

Our embedding is based on a single, simple, guest node to host node mapping. This mapping does not give us immediately the final embedding, because it is only a partial mapping from the guest (h, w)-grid. This partial mapping does not define an image for some guest nodes, namely those in the bottom left and top right corners in the diagrams. The final, total mapping is constructed from the partial mapping by methods to be described in later sections. In this section we describe and define the partial mapping and investigate some of its properties.

The partial mappings are illustrated in the same way as was done in the earlier example, Figure 2. Consider Figure 3 which defines a load 2, dilation one, mapping from the nodes of a sub-graph of a $(13, 20)$-grid to the nodes of an (8,13)-cylinder.

4.2 Formal Definition of the Partial Mapping

The partial mapping maps nodes from a grid of height h_g onto the nodes of a cylinder of height h_h, where $h_g/2 < h_h < h_g$, with load l. To completely define the mapping, we need a fourth parameter, which we call a *profile.* Let $\mathbf{p} = p_1 p_2 \cdots p_k$ denote a string of k integers.

Definition 4.1 *The string* $\mathbf{p}$ *is an* (k, n)*-profile if* $\sum_{i=1}^{k} p_i = n$ *and* $\forall i \; p_i \geq 2$.

Informally, the profile associated with a given mapping is the sequence of "step" heights on the left or right of a host column in a mapping diagram. We will refer to these step sequences as left and right column profiles when the context makes the meaning clear. To map a height h_g guest into a height h_h host we need an $((h_g - h_h, h_g))$-profile. Of course for any h_g, w_g pair there are many profiles satisfying the definition. For example, the (5,13)-profile used to generate Figure 3 and which is the left profile of host column 0 is 2 2 2 2 5.

For each $(h_g, h_h, l, \mathbf{p})$ 4-tuple we define a mapping, i.e., we define a function $F(h_g, h_h, l, \mathbf{p})$ whose value is the required node to node mapping.

For each column i of the host, $F(h_g, h_h, l, \mathbf{p})$ defines sets C_i of triples (r, c, v), where $0 \leq r < h_g$. The existence of the triple (r, c, v) in C_i indicates that node (r, c) of the guest is

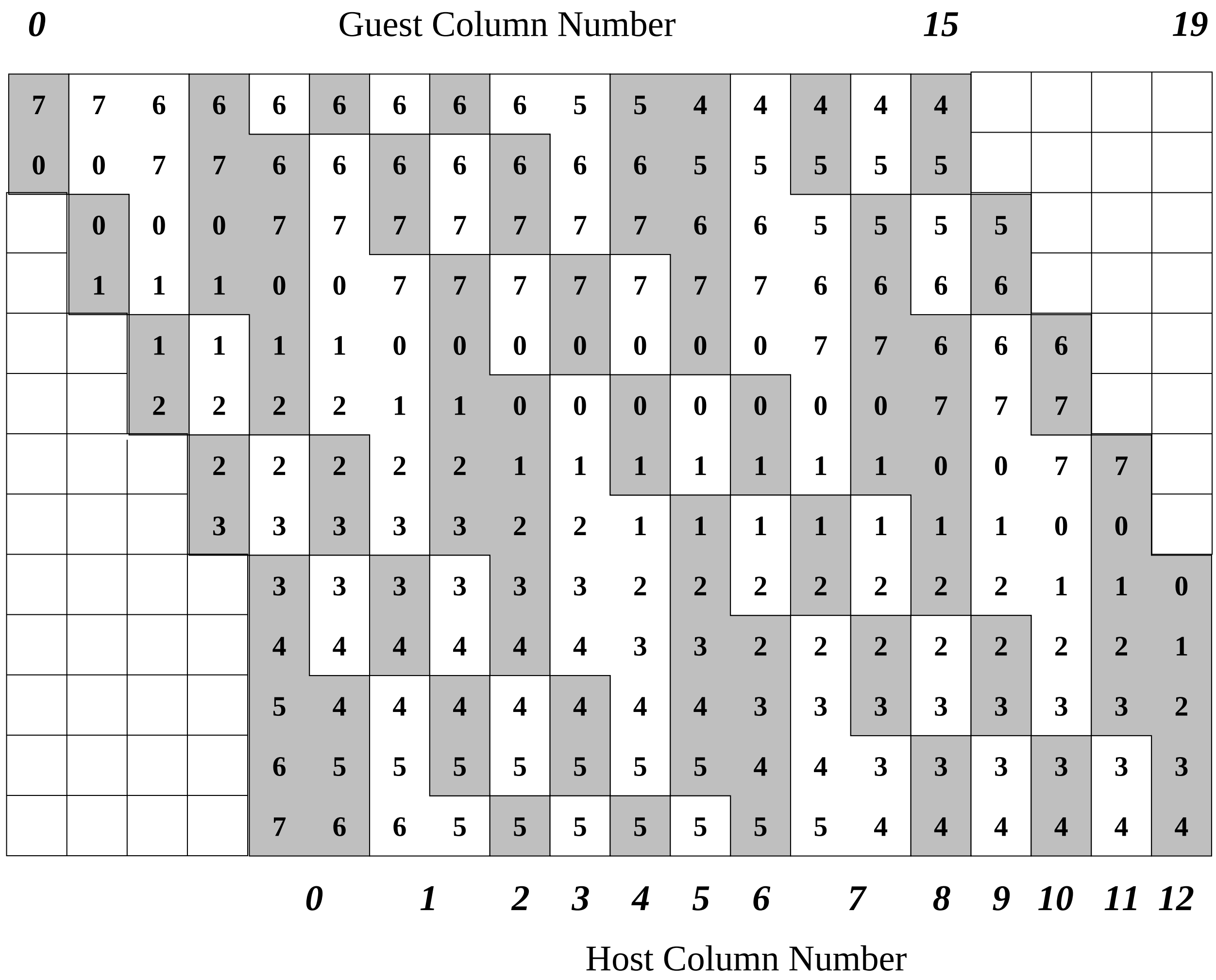


Figure 3: A height 13, load 2, partial mapping

to be mapped to node (v, i) of the host, i.e., the contents of C_i define completely which guest nodes are mapped onto which row node of column i in the host. Note that i is unbounded, meaning that we can apply the mapping to any width host of the given height.

Before presenting a formal definition of the mapping, which is used to prove claims about its properties and could be used as the basis for an algorithm to generate the mapping, we give an informal description. Figure 3 can be used as an example. Let $k = h_g - h_h$ so that we use a profile $\mathbf{p} = p_1 p_2 \cdots p_k$.

1. Starting at the top of guest column 0, mark off p_1 rows in that column. In column 1, mark off the p_2 rows in column 2 immediately following the marked rows in column 0. Continue in this fashion till the last p_k rows are marked in column $k-1$. By the definition of $\mathbf{p}$, h_g rows have been marked in the first k columns of the guest grid.

2. Leaving the top row in each marked column segment, write the numbers 0 through $p_1 - 2$ in the remaining $p_1 - 1$ rows in column 0, $p_1 - 1$ through $p_2 - 2$ in column 1, and so on to numbers p_{k-2} through $h_h - 1$ in column k. To this point we have computed C_0^0 as defined in item 1 (a) of Definition 4.2.

3. Repeat l - 1 times: run down the righthand edge of the current pattern and, for each of the most recent additions of the numbers 0 through $h_h - 1$, copy that number into the next location one column to the right and one row down from the current position. If moving down runs off the bottom of the pattern, place the number in the next vacant location in row 0. At this point we have completed the computation of C_0, i.e., the mapping into the first host column, as defined in item 1 (b) of Definition 4.2.

4. To create C_1 from C_0, i.e., host column 1 from host column 0, or in general C_{i+1} from C_i, all numbers are incremented by one and the pattern is cyclically rotated by l rows. In the cyclic rotation, items that run off the bottom are returned to the top rows, but the column number is decreased by k every time the new row number passes through a multiple of h_g. This process is defined in item 2 of Definition 4.2.

Here is the algebraic definition of the partial mapping.

Definition 4.2 *The Partial Mapping*
The C_i corresponding to the 4-tuple $(h_g, h_h, l, \mathbf{p})$, where $\mathbf{p}$ is an (k, n)-profile, are defined recursively as follows:

1. $C_0 = \bigcup_{j=0}^{l-1} C_0^j$ *where:*
   (a) *Let* $d_1 = 0$ *and* $\forall i, 2 \leq i \leq k,$ *let* $d_i = \sum_{q=1}^{i-1}(p_q - 1)$.
   then $\forall i, 1 \leq i \leq k, \forall z, d_i \leq z < d_i + p_i - 1, (z + i, i - 1, z) \in C_0^0$
   (b) $\forall j, 1 \leq j \leq l-1, (r, c, v) \in C_0^{j-1}$ *implies* $((r+1) \bmod h_g, c+1-\lfloor (r+1)/h_g \rfloor k, v) \in C_0^j$

2. $\forall i, 1 \leq i,$ $(r, c, v) \in C_{i-1}$ *implies*
   $((r + l) \bmod h_g, c + l - \lfloor (r + l)/h_g \rfloor k, (v + 1) \bmod h_h) \in C_i$

where "mod" denotes the remainder after integer division.

The reader will find that C_0, generated with $l = 2$, $h_g = 13$ and $h_h = 8$ and profile 2 2 2 2 5 yields host column 0 in Figure 3 and that C_1 defines host column 1 and so on. This definition can be used as the basis for an algorithm which computes the mapping using a number of arithmetic and other primitive operations linear in the size of the guest.

4.3 Properties of the Partial Mapping

The following properties of the mappings are evident from the diagrams and are straightforward to prove.

The Mapping Properties

1. Every node in the host is the image of exactly l guest nodes, i.e., the load is optimal (taking the size of the domain of the mapping to be the numerator).
2. If two nodes are adjacent in the guest and both have images defined by the mapping, then their images are adjacent in the host, i.e., the dilation of the mapping is one.
3. Each edge in a host column appears no more than $2l - 1$ times and each edge in a host row appears no more than l times, i.e., the congestion of the mapping is $\leq 2l - 1$.
4. The left and right column profiles are rotated through l rows from one host column to the next.

The first statement follows directly from rule 1(b) in the definition of the mapping. The second can be demonstrated by an inductive argument based on noting that within a column adjacent array elements are given adjacent row numbers and that across column boundaries adjacent elements are given identical row numbers. The third statement follows from the fact that adjacent consecutive row numbers within a host column correspond to column edges and there are no more than $2l - 1$ such adjacencies per edge. Host row edges correspond to adjacent numbers across host column boundaries. Along a boundary, each row number can appear no more than l times. The fourth statement follows directly from rule 2 in the definition of the mapping.

The partial mapping has two significant periodicity properties. For example, notice that in Figure 3 the left profile of host column 0 and the right profile of host column 12 (which is identical to the left profile of host column 13) are identical.

Definition 4.3 *The* structural periodicity *of the mapping is the smallest integer, p_s, such that $\exists \delta_s \forall i (r, c, v) \in C_i$ implies $(r, c + \delta_s, (v + p_s) \bmod h_h) \in C_{i+p_s}$.*

Note also that the values in host column 13 (the next column in the mapping but not shown in the diagram) would not be identical to those in column 0. However, C_0 would be identical, in both profiles and content, to C_{104} ($104 = 8 \times 13$), were the mapping to be continued that far.

Definition 4.4 *The* numerical periodicity *of the mapping, is the smallest integer, p_n, such that $\exists\delta_n \forall i(r, c, v) \in C_i$ implies $(r, c + \delta_n, v) \in C_{i+p_n}$.*

The following lemmas show how these periodicities are related to the parameters that define the mapping. We refer to the greatest common divisor and to the lowest common multiple of two positive integers i and j as $gcd(i, j)$ and $lcm(i, j)$ respectively and we use $i \mid j$ to mean i divides j.

Lemma 4.1 *The structural periodicity of the pattern is given by: $p_s = h_g/gcd(l, h_g)$ and the constant δ_s is given by: $\delta_s = lh_h/gcd(l, h_g)$.*

Proof
Suppose the operation defined in rule 2 of the initial mapping definition is repeated $p_s = h_g/gcd(l, h_g)$ times on a triple (r, c, v), producing the triple (r', c', v').
Then:

- $r' = (r + lp_s) \bmod h_g = (r + lh_g/gcd(l, h_g)) \bmod h_g = r$.
- Let $k = h_g - h_h$. The column number c will be incremented, by l, p_s times and decremented, by k, the number of times the value of r, incremented by l at each iteration, passes through a multiple of h_g, namely $\lfloor (r + lp_s)h_g \rfloor$ times. Noting that $\lfloor r/h_g + (l/h_g)(h_g/gcd(l, h_g)) \rfloor = l/gcd(l, h_g)$ we have:

$$\begin{aligned} c' &= c + lh_g/gcd(l, h_g) - lk/gcd(l, h_g) \\ &= c + l(h_g - k)/gcd(l, h_g) \\ &= c + lh_h/gcd(l, h_g) \end{aligned}$$

- $v' = (v + p_s) \bmod h_h$
- $h_g/gcd(l, h_g)$ is the smallest integer z with the required properties because, z must satisfy $r = (r + lz) \bmod h_g$. Then it must be that: $h_g \mid lz$, i.e., $l = ax$, $z = by$ where $xy = h_g$ for some integers a, b, x, y. Since z is minimal, it must be that $x = gcd(l, h_g)$. Hence $z = by = bh_g/x = bh_g/gcd(l, h_g) \geq h_g/gcd(l, h_g)$. □

Lemma 4.2 *The numerical periodicity of the pattern is given by:*

$$p_n = lcm(h_h, h_g/gcd(l, h_g)).$$

Proof
We require that $r' = r$ which implies that $p_n = ap_s$ for some integer a. Also we must have $v' = v$ and hence $ap_s \bmod h_h = 0$. Hence $ap_s = lcm(h_h, p_s) = lcm(h_h, h_g/gcd(l, h_g))$. □

Note that both periodicities are independent of the profile $\mathbf{p}$ used to initiate the mapping.

For some of our constructions we need a lower bound on what we will call the *column thickness* of a mapping.

Definition 4.5 *The* column thickness *of a mapping is the minimum over all rows of the number of grid columns in the row mapped to the same host column.*

Lemma 4.3 *The thickness of a partial mapping is* $\geq \lfloor l/2 \rfloor$.

Proof
The thickness of C_0 is $\geq \lfloor l/2 \rfloor$ because the $C_0^0 \cup C_0^1$ includes every row at least once. Hence every two repetitions of rule 1(b) of the mapping definition adds at least one to the thickness. The later columns are formed by rotation of C_0 which do not change its shape. □

5 Embedding into Cylinders

So far we have described only partial mappings from guest grid nodes to the host cylinder nodes, but our goal is to define a total mapping for (h, w)-grids. Obviously, a partial mapping can yield a total mapping for any (h, w)-grid all of whose nodes fall within the domain of the partial mapping. By a *partially mapped* guest column we mean one for which one or more but not all column nodes have an image defined by the mapping. For example in Figure 3 we can obtain a load 2, dilation 1 embedding of the $(13, 12)$-grid into an $(8, 13)$-cylinder by discarding the partially mapped columns.

In this section we improve on that by showing that we can "square up" at least one end of the mapping diagram. This squaring up method works for one particular family of profiles. Let $k = h_g - h_h$. The (k, h_g)-profile we need is a sequence of $k - 1$ twos followed by $h_g - 2(k - 1)$. Let us call these *cylindrical* profiles. Figure 3 was initiated by such a profile.

A guest node will be said to be *uncovered* if it is not in the domain of the partial mapping. In the mapping diagrams, the uncovered nodes that we consider now are those making up the bottom left corner. We will refer to this set of uncovered nodes as the *corner*.

Lemma 5.1 *There are* $(k - 1)h_h$ *nodes in the corner of a partial mapping initiated by a cylindrical profile.*

Proof
We deduce from the definition of the partial mapping, rule 1(a), that there are $k-1$ partially mapped columns, that the leftmost such column is of height $h_g - 2$ and that the rightmost is of height $h_g - 2k + 2$, decreasing by 2 at each column. A simple summation yields: number of uncovered nodes $= (k - 1)(h_g - k) = (k - 1)h_h$. □

The remainder of the paper is presented in terms of mapping diagrams. We do not attempt to specify the mappings algebraically. The following process creates a corner mapping diagram which maintains the necessary mapping properties, for any load l.

1. Take $(k-1)$, height h_h columns, numbered zero through $k-2$. Assign the first l columns to host column 0, the next l to host column 1, and so on until all are given some designation. The columns will now be divided into $\lceil (k-1)/l \rceil$ groups, with l columns per group, except perhaps for the last group which will contain $(k-1) \bmod l$ columns. Number the rows of column 0 from 0 downwards to $h_h - 1$, (or any rotation thereof). See Figure 4 a), where the an appropriate rotation is chosen to match Figure 3.

2. For columns 1 through $k-2$:

   - if the guest node and its left neighbour are assigned to the same host column, assign a row number one less (cyclically) than its left neighbour.
   - if the column and its left neighbour are assigned to different (adjacent) host columns, assign a row number identical to that of its left neighbour.

   See Figure 4 b).

3. Repeat the following *shift* process $k-2$ times:
   Consider a 45 degree diagonal drawn from above the top-leftmost element, down to the right edge of the current profile. Shift everything above this diagonal one place up and one place to the left.
   See Figure 4 c) through e).

We observe that:

- The diagram covers $h_h(k-1)$ nodes, i.e., the number originally uncovered, and distributes them with load $\leq l$ across $\lceil (k-1)/l \rceil$ host columns.
- After the steps 1 and 2, mapping properties 2 and 3 hold.
- Mapping properties 2 and 3 are preserved at each step of the shifting process. This can be seen by noting that:
  - Any diagonal shift involving elements originally related in the way defined by steps 1 and 2, preserves the properties. Consider two horizontally adjacent elements, *left* and *right*, on either side of the diagonal which defined the shift. Shifting puts *right* above *left* and the element which was below and to the right of *right* becomes the right neighbour of *left*. In each of the only two possible cases, i.e., *left* and *right* are either assigned to the same or to adjacent host columns, the row numbering defined in steps 1 and 2 ensures that mapping properties are preserved.
  - All diagonal shifts are on elements related as defined in steps 1 and 2, as the diagonal moves up through previously undisturbed elements.
- Height h_g - 2 is achieved.

- The right profile of the result is a cylindrical profile.

Finally we note that once a corner has been built in this way, we can continue the mapping in the usual way. It is necessary only to note whether those uncovered elements adjacent to elements covered by the corner building process should be assigned to the same host column or to the next one.

Case 1: $l \mid (k-1)$, i.e., the number of host columns used in the corner is a multiple of the load. In this case, the mapping that follows the corner begins with a new host column. Then the mapping definition 1(a) is initiated not by a profile but by the numbers already defined by the corner. Then definition 1(b) is applied without change.

Case 2: not $(l \mid (k-1))$, i.e., the available repetitions of the current host column, as determined by the load, have not been completely used. Again rule 1(a) is modified so that C_0^i, for some i is defined not by a profile but by the numbers defined by the corner construction. C_0^{i+1} through C_0^{l-1} are defined as usual by rule 1(b).

Figure 4 shows the construction of a corner for the problem instance: $h_h = 8$, $h_g = 13$ and load 2. Here we need 4 columns to fill the corner, i.e., 2 host columns, each appearing twice.

Figure 5 shows a similar example, the construction of a corner for the same problem instance, but with load 3. Now we need 2 host columns, one appearing 3 times and the other once. As the mapping continues, it will first use 2 copies of this latter host column.

In general we know how to square up the other end of the mapping only in certain special cases. We can however put an upper bound on how many extra host columns we may have to use to make up for the uncovered nodes in the top right corner of the mapping. We need to consider the *skew* of the mapping.

Definition 5.1 *Let the mapping be defined for host columns 0 through i. The* skew *of a mapping is the number of guest columns that are partially mapped and containing some element mapped to host column i.*

Lemma 5.2 *The skew of any partial mapping is $\leq h_g - h_h$.*

Proof
By definition of the mapping, rule 1(a), there are $h_g - h_h - 1$ partially mapped guest columns containing some element mapped to host column 0. As the mapping progresses across host columns the profile can be split by the wrap-around at only one place which can increase the skew by no more than one over the minimum. □

Let w_c denote the smallest integer such that $h_g w_g \leq l h_h w_c$, i.e., w_c is the smallest width of a host of height h_h that can accommodate the given guest at the given load. Let k denote $h_g - h_h$.

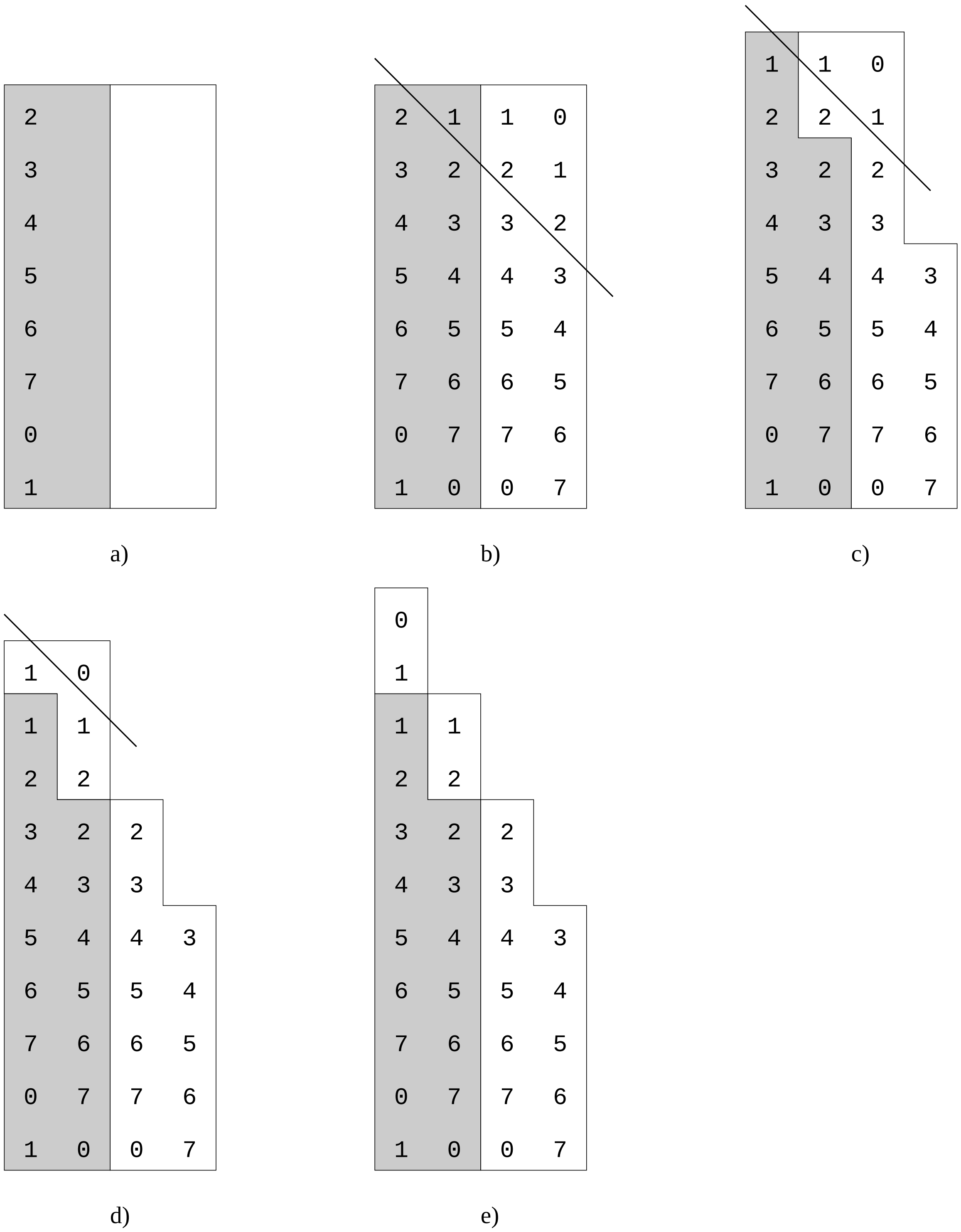


Figure 4: Creating a corner: $h_g = 13$, $h_h = 8$, load 2

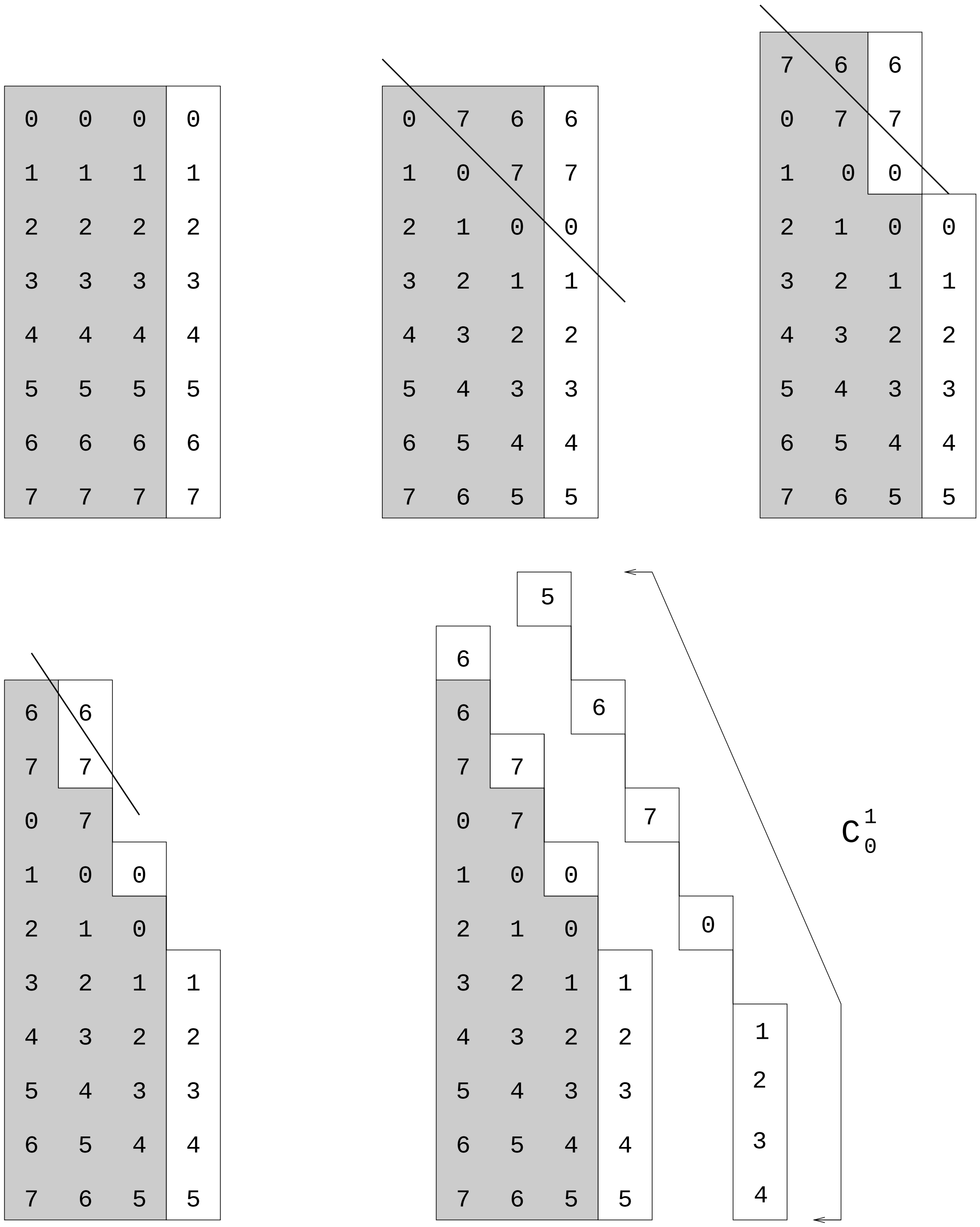


Figure 5: Creating a corner: $h_g = 13$, $h_h = 8$, load 3

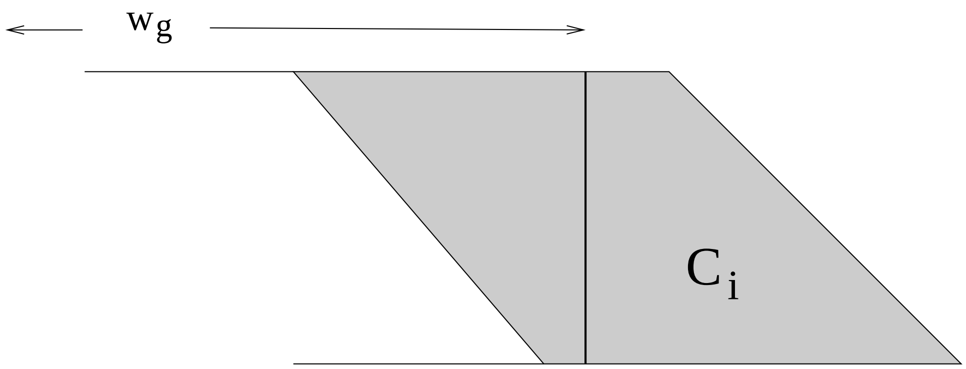


Figure 6: A high load can yield an optimal embedding

Theorem 5.1 *If $h_g/2 < h_h < h_g$ then the (h_g, w_g)-grid is embeddable in the (h_h, w_h)-cylinder, where $w_h = (w_c + \lceil k/\lfloor l/2 \rfloor \rceil)$ with dilation 1 and load l.*

Proof
We can square up the left end of the mapping as just demonstrated. The skew is an upper bound on the number of partially mapped columns at either end. Hence, by Lemma 5.2, the skew of the mapping at the right end is not greater than k. By Lemma 4.3, each host column covers at least $\lfloor l/2 \rfloor$ nodes from every row of the grid. Hence we need no more than $\lceil k/\lfloor l/2 \rfloor \rceil)$ extra host columns to make up for the skew.

□

We can also note that, if the load is sufficiently high, guest column w_g may be completely mapped, without the addition of extra host columns beyond column w_c. Such a situation is illustrated in Figure 6, where host column C_i is necessary, but not completely used.

Theorem 5.2 *If $lh_h w_h - h_g w_g \geq kh_h$ and $h_g/2 < h_h < h_g$ then the (h_g, w_g)-grid is embeddable in the (h_h, w_h)-cylinder with dilation 1 and load l.*

Proof
The skew is the number of partially mapped columns. The leftmost of these of height $\leq h_g - 1$, the next of height $\leq h - 3$ etc. for $\leq k$ columns. Summing this series shows that the number of guest nodes in the partially filled columns is $\leq k(h_g - k) = kh_h$. Hence we need to add no extra host columns so long as $lh_h w_c - h_g w_g \geq kh_h$. □

6 Embedding into Tori

6.1 The problem

We now consider the "hardest" instances of the problem of embedding an (h_g, w_g)-grid into a (h_h, w_h)-torus, namely those where w_g is maximal with respect to preserving $h_g w_g \leq lh_h w_h$

We will derive certain restrictions on the load and on the guest and host dimensions under which optimal solutions can be constructed.

To obtain these solutions we match up the ends of a partial mapping. Consider first those special cases where w_g is a multiple of the numerical periodicity (Definition 4.4 and Lemma 4.2). By definition, the host row numbers and profiles at the left and right ends of the partial mapping match. Hence we can cut the mapping vertically anywhere outside of the partially mapped columns, transpose the left and right portions of the mapping and so produce a total mapping for the guest grid.

For example, consider Figure 7. In the illustration, a continuation of the example shown in Figure 3, $h_g = 13$, $h_h = 8$ and $l = 2$. Hence, by Lemma 4.2, the numerical periodicity, $p_n = 104$. So the figure is illustrating an embedding of the (13,128)-grid into the (8,104)-cylinder with load 2.

Of course, except in these very special cases, neither the row numbers nor the profiles at the ends of the mapping will match. We now show how to solve both of these problems.

6.2 Toroidal Profiles

For embeddings into toruses we must initiate the partial mapping with a unique profile. We will refer to them here as *toroidal profiles*. In [ERS00], where their existence and properties are demonstrated, they are called *Euclidean strings* because they are generated by an extension to Euclid's greatest common divisor algorithm. In this section we show how to generate these special profiles and state their properties, without repeating the proofs.

Let $\mathbf{p} = p_1 p_2 \cdots p_k$ denote a string of k non-negative integers. Let $\rho(\mathbf{p})$ denote a left rotation of $\mathbf{p}$ by one position, i.e., $\rho(\mathbf{p}) = p_2 \cdots p_k p_1$ and let $\rho^d(\mathbf{p})$ denote a left rotation through d positions. Let $\sigma_i(\mathbf{p})$ be the string obtained from $(\mathbf{p})$ by replacing p_i by $p_i + 1$ and $p_{(i+1) \bmod k}$ by $p_{(i+1) \bmod k} - 1$.

Definition 6.1 *The string* $\mathbf{p}$ *is a* toroidal (k,n)-profile *if* $\sum_{i=1}^{k} p_i = n$ *and there exist integers* d *and* i *such that* $\sigma_i(\mathbf{p}) = \rho^d(\mathbf{p})$.

In [ERS00] the following lemma is established:

Lemma 6.1 *A toroidal (k,n)-profile exists iff gcd(k,n) = 1, where gcd denotes the greatest common divisor. If it exists then it is unique.*

Suppose 1^r denotes a string of r ones and suppose the function expand($\mathbf{p}$) replaces every element r in $\mathbf{p}$ by the string 01^r and the function increment($\mathbf{p}$) replaces every number r in $\mathbf{p}$ by $r+1$. Then Figure 8 describes, in pseudo-code, a procedure that computes a (k,n)-profile when k and n are relatively prime.

For the application of profiles to the computation of our embedding we also need to compute the following four parameters. We note that since the profile is unique, all these parameters are well defined.

- i and d – as in the definition of profile

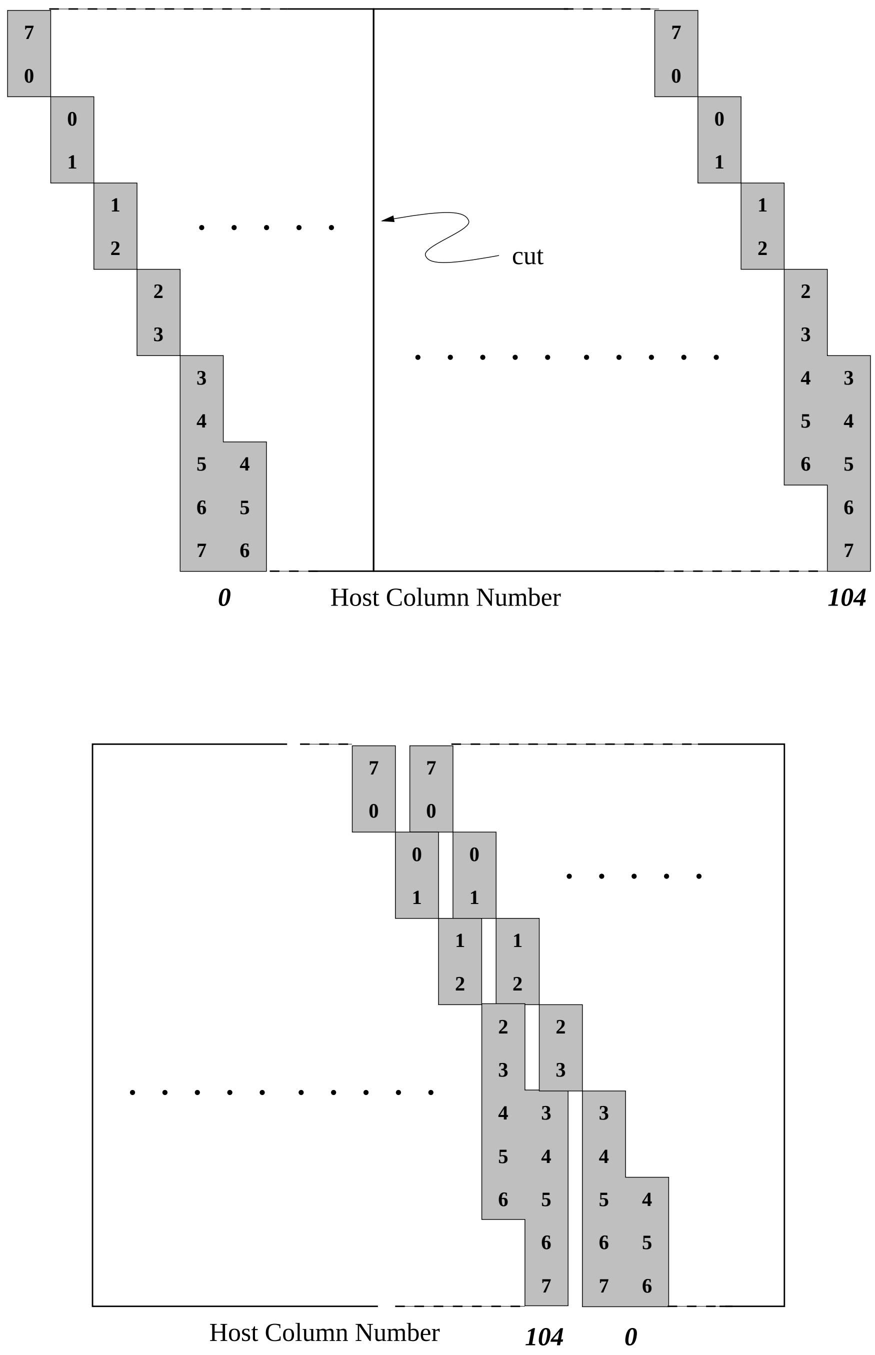


Figure 7: Matching up the ends of a partial mapping

function profile *(k, n : integer) : string*;
if $(k = 1)$ and $(n = 1)$**then** profile := 1
else if $n > k$ **then** profile := expand(profile$(k, n - k)$)
else profile := increment(profile$(k - n, n)$);

Figure 8: Computing a Profile

procedure parameters $(k, n : integer)$;
if $(k = 1)$ and $(n = 1)$ **then** $i := 0$; $j := 0$; $c := 0$; $d := 1$
else if $k < n$ **then** parameters $(k, n - k)$; $j := i + j$; $c := c + d$ **else** parameters $(k - n, n)$;
$i := i + j + 1$; $d := c + d$;

Figure 9: Computing the Profile Parameters

- j – the sum of elements p_1 through p_i
- c – the sum of the d elements to the left (cyclically) of and including p_i.

Figure 9 shows a pseudo-code procedure which computes these parameters, assuming that all the variables are global. Figure 10 shows the computation of a string and its associated parameters for particular values of k and n.

We will refer to the parameters c and d as the *cost* and *displacement* of a profile respectively and we let $c_{k,n}$ and $d_{k,n}$ denote the cost and the displacement respectively of a toroidal (k, n)-profile.

Lemma 6.2 *If $n + k \geq 2$ then $d_{k,n}n = c_{k,n}k + 1$*

Proof
We argue by induction on $n + k$. For the basis we note that if $n + k = 2$ then $(n = 1)$ and $(k = 1)$ and hence the initialisation defined by the algorithm satisfies the lemma.

We note that n and k are not equal because $gcd(n, k) = 1$. For the induction suppose first that $n > k$. By the inductive hypothesis: $d_{k,n-k}(n - k) = c_{k,n-k}k + 1$.
Hence $d_{k,n-k}n = d_{k,n-k}(n - k) + d_{k,n-k}k = (c_{k,n-k} + d_{k,n-k})k + 1$.
But, when $n > k$, the algorithm sets $d_{k,n-k} = d_{k,n}$ and $c_{k,n-k} + d_{k,n-k} = c_{k,n}$.
Hence $d_{k,n}n = c_{k,n}k + 1$.

Suppose $n < k$. By the inductive hypothesis: $d_{k-n,n}n = c_{k-n,n}(k - n) + 1$.
Hence $(d_{k-n,n} + c_{k-n,n})n = c_{k-n,n}k + 1$.
But, when $n < k$, the algorithm sets $d_{k-n,n} + c_{k-n,n} = d_{k,n}$ and $c_{k-n,n} = c_{k,n}$.
Hence $d_{k,n}n = c_{k,n}k + 1$. □

So $d_{k,n}$ and $-c_{k,n}$ are in fact the constants computed by the standard extended Euclidean algorithm. Lemma 6.2 yields the following corollary.

Corollary 6.1
$c_{k,n}$ is the multiplicative inverse of $(n - k)$ modulo k, i.e., $c_{k,n}(n - k) \equiv 1 \bmod n$.

k	n		p	i	j	d	c
14	33	I	22322322322323	11	25	3	7
14	19	I	11211211211212	11	14	3	4
14	5	E	00100100100101	11	3	3	1
9	5	E	010101011	7	3	2	1
4	5	I	1112	3	3	1	1
4	1	E	0001	3	0	1	0
3	1	E	001	2	0	1	1
2	1	E	01	1	0	1	0
1	1		1	0	0	1	0

^ -- indicates the exchange point

i -- the ith element is exchanged with the (i+1)th

d -- the left rotation which has the same effect as the exchange

c -- the sum of the d numbers to the left of the exchange point

I -- an incrementation operation

E -- an expansion operation

Figure 10: An Example of Computing a Profile and its Parameters

Proof
By Lemma 6.2, $d_{k,n}n = c_{k,n}k + 1$, implying $-c_{k,n}k = -d_{k,n}n + 1$.
Hence $c_{k,n}(n-k) = c_{k,n}n - c_{k,n}k = c_{k,n}n - d_{k,n}n + 1 = (c_{k,n} - d_{k,n})n + 1$
Hence $c_{k,n}(n-k) \equiv 1 \bmod n$. □

It is also shown in [ERS00] that all the elements in a profile are of the form i or $i+1$ for some integer i. We will be using $(h_g - h_h, h_g)$-profiles and so, since $h_h > h_g/2$, all the profile elements will be ≥ 2.

6.3 Matching profiles

We are considering those problem instances where $0 \leq l|H| - |G| < h_g$, the case where the difference is zero being trivial. So we need a way of dropping between 1 and $h_g - 1$ host nodes from the mapping while simultaneously guaranteeing that the left profile of host column 0 is identical to the right profile of the last column.

Consider the running example where $h_g = 13$ and $h_h = 8$. The application of the toroidal profile generator, Figures 8 and 9, yields the profile: 2 3 2-3 3, where swapping the hyphenated pair yields a rotation of the original string. In our application the profile elements define the height of the steps in the left or right profiles of the host columns.

Suppose our mapping is based on a toroidal profile and suppose we remove element $(i+1)$ from the mapping in the right profile of some host column, where the i is the profile parameter in Definition 6.1. Removing an element decreases the height of the step containing that element and increases the height of the step above. But, because the profile is toroidal, the resulting profile is a rotation of the original. Because the profile is preserved, except for a rotation, the operation can be repeated as many times as necessary. This is illustrated in Figure 11.

We deduce that we can drop $(l|H| - |G|) < h_g$ elements and be assured that the right profile of the rightmost host column is a rotation of the left profile of column 0. It remains only to show that not only are these two profiles rotations of each other, but they are identical. For the moment, let us assume that we apply the drop process to distinct host columns.

Lemma 6.3 *If the drop operation is applied to $l|H| - |G|$ distinct host columns the right profile of the rightmost host column is identical to the left profile of host columns 0.*

Proof
Every host column rotates the profile l rows in what we will call the forward direction. Therefore the profile shift s in the forward direction relative to its initial position is given by:

$$s = (l(w_h \bmod p_s)) \bmod h_g \tag{1}$$

where p_s denotes the structural periodicity. By Lemma 4.1, $p_s = h_g/gcd(l, h_g)$ and so we derive from Equation 1:

$$s = (lw_h) \bmod h_g \tag{2}$$

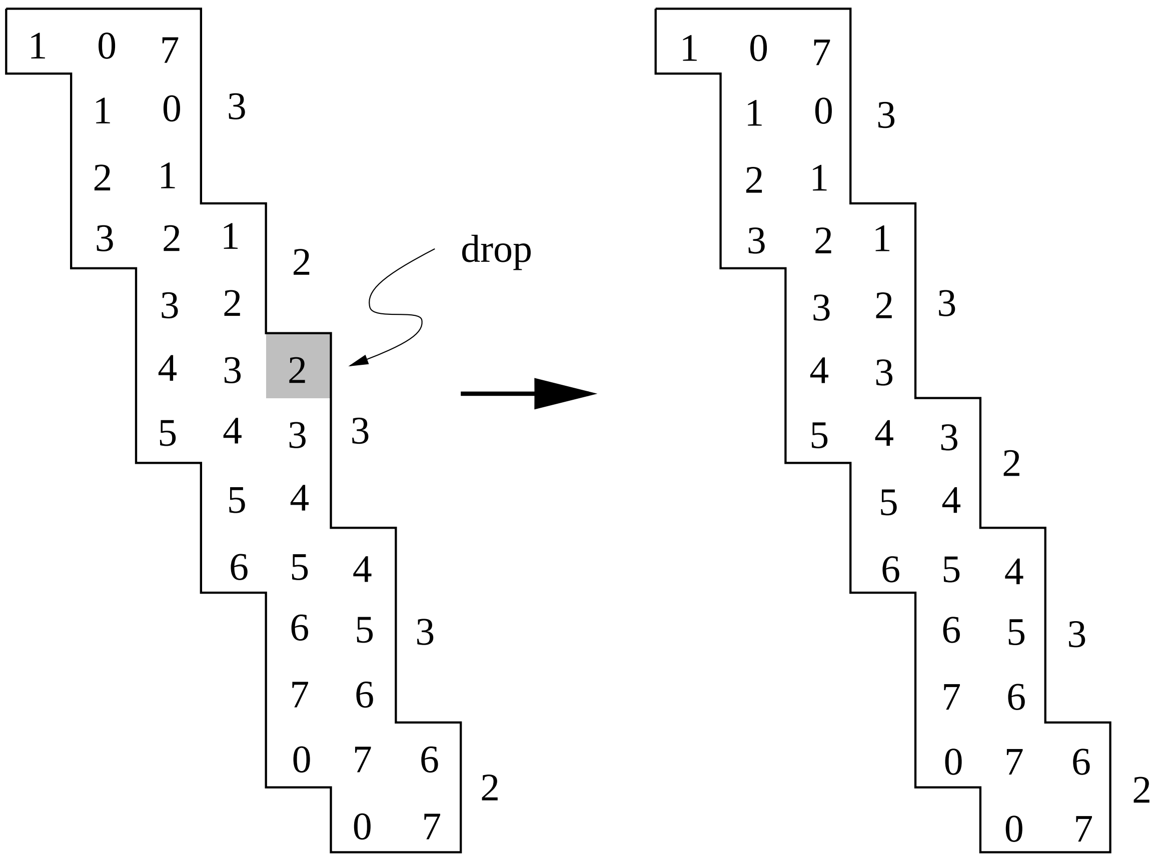


Figure 11: Dropping an element while preserving the profile

Let $\delta = lw_hh_h - w_gh_g$. Suppose we apply the drop process δ times. Each application shifts the profile c rows backwards, where c is the cost of the profile. So the repeated application shifts the profile $s' = \delta c \bmod h_g$ rows. But:

$$\delta c \bmod h_g = c(lw_hh_h - w_gh_g) \bmod h_g \tag{3}$$

and so:

$$s' = lw_h \bmod h_g = s \tag{4}$$

because $ch_h \bmod h_g = 1$ by Corollary 6.1.

Hence the profile shift caused by the excess host columns is exactly compensated for by the repeated application of the drop process. Finally we note that all the drop operations can be applied to one host column, without removing all the elements from that column from any one row.

Lemma 6.4 *If* $(l|H| - |G|) < h_g$ *drop operations are applied to one host column and the load* $l \geq 4$ *the resulting width of the column is at least one.*

Proof

By Corollary 6.1 $ch_h \bmod h_g = 1$. It follows that c and h_g are relatively prime. Hence, since $\delta = (l|H| - |G|) < h_g$, if up to δ elements are dropped, they are all taken from different rows, so the width of the column is decreased by no more than one. By Lemma 4.3, the width of a column in the partial mapping is at least $\lfloor l/2 \rfloor$, hence if $l \geq 4$, the resulting width is at least $\lfloor l/2 \rfloor - 1 = 1$. □

6.4 Matching row numbers

We describe a perturbation of the partial mapping which has the effect of incrementing all the host row numbers by one or two in all the columns following the perturbation. The perturbation can be applied to any instance of the problem for which the load is at least 4.

Let us suppose we are perturbing host column i. C_i is composed of C_i^0 through C_i^{l-1}. The perturbation simply adds one or two to all the host row values in C_i^2 and to those in C_i^3. If $l > 4$ then the mapping follows on in the usual way after C_i^3. It is clear that the perturbation is still yielding dilation 1. In fact, at the junction where the perturbation occurs, adjacent guest nodes are mapped to the same host node, i.e., for those edges the dilation is zero. It is also clear that, since only the values in the mapping are changed, the profiles are not changed.

Figures 12 and 13 illustrate the process with a load 4 example, following from our running example. In these figures the shading distinguishes C_i^0 and C_i^1 from C_i^2 and C_i^3.

Suppose we need to increment the row numbers by more than one. The process just described can be repeated on C_i^4 and C_i^5 etc. within a host column. Each column therefore can be used to obtain an increment of up to $2\lfloor (l-2)/2 \rfloor$. We observe that the row values never need to be incremented by more than $h_h - 1$ and that we can adjust either by incrementing by one or two. We have then justified the following lemma.

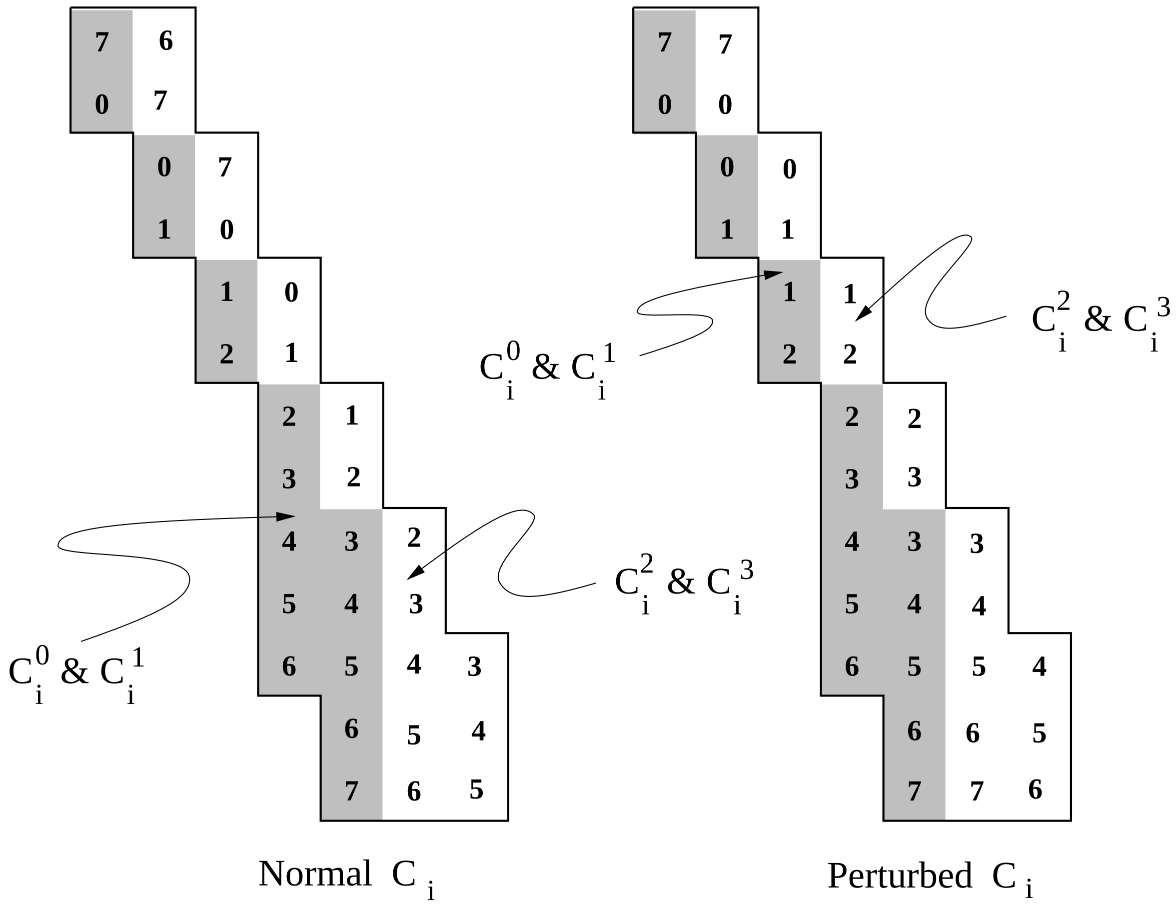


Figure 12: The perturbation that increments row numbers by one

Lemma 6.5 *If $\lceil (h_h - 1)/2 \rceil \leq w_h \lfloor (l-2)/2 \rfloor$ and the right profile of the last column in the mapping is identical to the left profile of column 0 then we can adjust row values so that they are equal in these two profiles.*

6.5 The solution

The computation of the total mapping can then proceed as follows:

1. Compute a toroidal $(h_g - h_h, h_g)$-profile together with the associated parameters using Figures 8 and 9.

2. Use this profile to initiate the computation of a partial mapping using Definition 4.1.

3. Compute δ, the difference between available host nodes and the number of guest nodes and apply the drop process δ times to (say) the right profile of the rightmost host column.

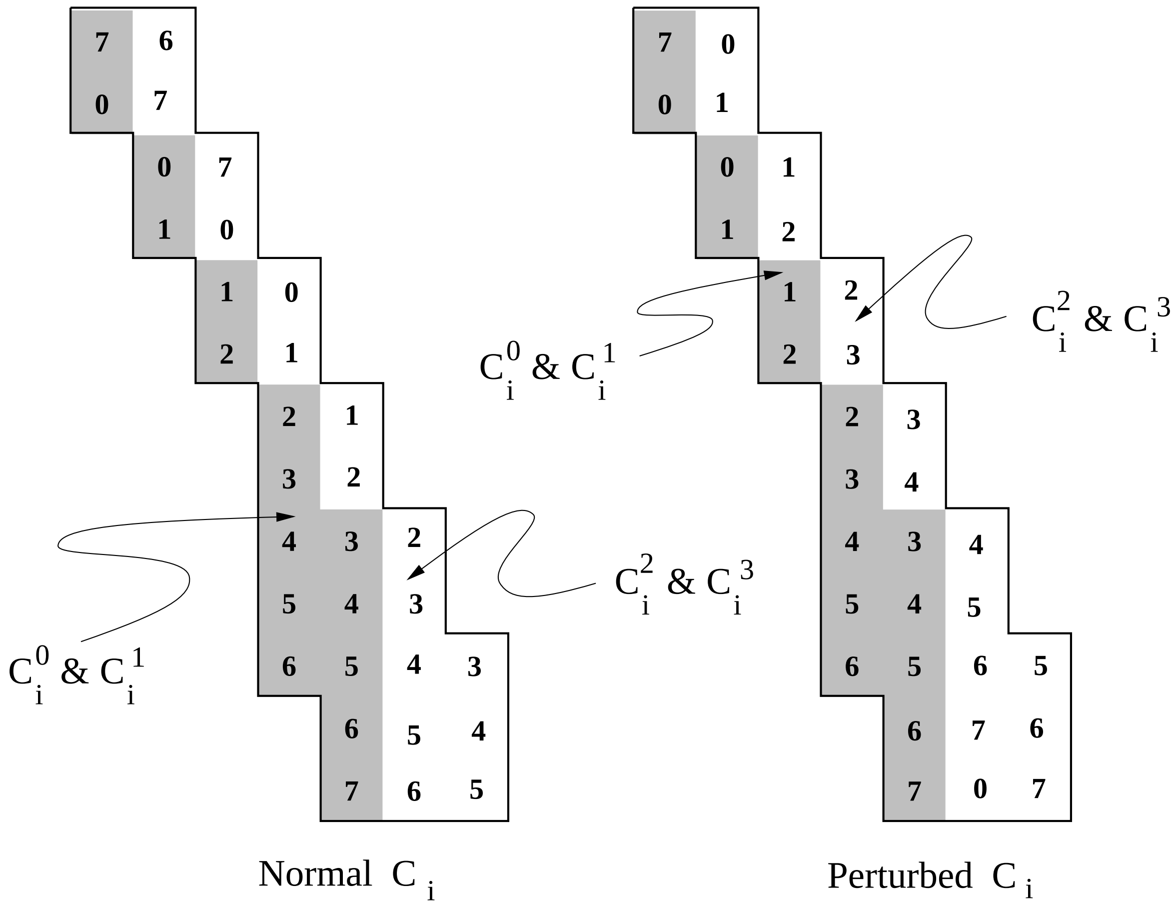


Figure 13: The perturbation that increments row numbers by two

4. Compute the row number incrementation required and apply the incrementation perturbation.

5. Cut the modified partial mapping vertically at any point outside of the partially mapped columns and transpose the left and right portions produced by the cut.

The conditions under which this process is effective are summarised in the following theorem.

Theorem 6.2
If

1. $h_g/2 < h_h < h_g$ *and*

2. h_g *and* $(h_g - h_h)$ *are relatively prime and*

3. $w_h(\lfloor (l-2)/2 \rfloor) \geq \lceil (h_h - 1)/2 \rceil + 1$ *and*

4. *the optimal load is at least 4*

then the (h_g, w_g)*-grid can be embedded in the* (h_h, w_h)*-torus with optimal load and dilation one.*

Proof
The first item is a necessary condition for the construction of the partial mapping. The second is sufficient for a suitable profile to exist for the initiation of the mapping. The restriction on the load is sufficient for the increment operation and allows the drop operation to be repeated on one column.

The last item ensures that there is sufficient width within which to carry out both the row incrementation and the drop operations. Each host column i is composed of $\lfloor (l-2)/2 \rfloor$ pairs C_i^k, C_i^{k+1} where $k \geq 2$. We may not assume that we can apply the increment operation on a pair that has already been the subject of the drop operation. As many as $\lfloor (h_h - 1)/2 \rfloor$ increment operations may be necessary. Consequently, we have sufficient host width if: $w_h(\lfloor (l-2)/2 \rfloor) \geq \lfloor (h_h - 1)/2 \rfloor + 1$. □

7 Embedding into Hypercubes

The previous section has shown that we can construct optimal embeddings into the torus under some restrictions on the load and dimensions of the guest and host. Because we can always find a torus of suitable height within a hypercube host, we obtain a much simpler result.

Consider the restrictions listed in Theorem 6.2 when applied to the embedding of the (h_g, w_g)-grid into the hypercube Q_d of dimension d.

1. By Lemma 3.2 there is a toroidal subgraph of the hypercube with height h_h in the range $h_g/2$ to h_g, namely $h_h = 2^{\lfloor \log_2 h_g \rfloor}$ which is the next power of 2 less than h_g. Then also w_h is a power of two, namely $2^{d-\lfloor \log_2 h_g \rfloor}$.

2. If h_g is an odd number, then h_g and h_h are relatively prime, since h_h is now a power of two.

3. We retain the condition on the optimal load, namely $l = \lceil h_g w_g / 2^d \rceil \geq 4$.

4. Nor can the last restriction be dispensed with, but see the corollaries below. It can be simplified slightly by noting that h_h is a power of 2, yielding:

$$w_h(\lfloor (l-2)/2 \rfloor) \geq h_h/2 + 1$$

   where h_h and w_h are now as defined in the first item.

Let h_h and w_h be the dimensions of the host torus as defined above as functions of the guest grid and host hypercube dimensions, h_g, h_h and d. Then the following theorems follow from Theorem 6.2.

Theorem 7.1
If

1. h_g *is odd and*

2. *the optimal load* $l \geq 4$ *and*

3. $w_h(\lfloor (l-2)/2 \rfloor) \geq h_h/2 + 1$

then the (h_g, w_g)*-grid can be embedded into* Q_d *with optimal load and dilation one.*

We can not guarantee that all instances of the problem present sufficient width to permit the necessary incrementation and drop operations. For example, the problem instance:

$$h_g = 11, w_g = 14, h_h = 8, w_h = 4, l = 5$$

does not satisfy the third constraint of Lemma 7.1. Some slightly stronger but simpler forms of that constraint show more clearly that it is excluding only a small portion of problem instances.

Corollary 7.2
If h_g *is odd and the optimal load* $l \geq 4$ *and* $w_h \geq h_h$ *then the* (h_g, w_g)*-grid can be embedded into* Q_d *with optimal load and dilation one.*

Proof
If $w_h \geq h_h$, then the third constraint of Theorem 7.1 is satisfied. $\square$

Corollary 7.3
If h_g is odd and either $((l = 4$ or $l \geq 6)$ and $w_g \geq 2h_g)$ or $(l = 5$ and $w_g \geq 5h_g/2)$ then the (h_g, w_g)-grid can be embedded into Q_d with optimal load and dilation one.

Proof
Since, by definition of the problem, $lh_h w_h \geq h_g w_g$ and $w_g \geq h_g > h_h$, if $w_g \geq 2h_g$, we deduce that $lw_h > 2h_h \geq 2h_h + 1$. But $\lfloor l/2 \rfloor - 1 \geq l/4$ for $l = 4$ and $l \geq 6$, thus satisfying the third constraint. For load 5 the same algebra shows that $w_g \geq 5h_g/2$ suffices. $\square$

The theorem implies that, if we group the problem instances by h_g, there are no more than a finite number of instances of the problem in each group that may not be solved by our method. Finally we note that instances where h_g is even are either trivial if h_g is a power of two, or can be reduced to instances where h_g is odd.

Theorem 7.4
If h_g is of the form $a2^k$ for some odd integer a and positive integer k and either $k \leq d$ and there is an optimal embedding from the (a, w_g)-grid into the hypercube of degree $d - k$ or $k > d$ then the (h_g, w_g)-grid can be embedded into the hypercube of degree d with optimal load and dilation one.

Proof
If $k \leq d$ we note that it must be that $2^k | (l2^d - a2^k w_g)$. Hence we can solve the sub-problem of embedding the (a, w_g)-grid into Q_{d-k} with load l and then construct a solution to the original problem by simple duplication of the host.

If $k > d$ then $|Q_d| < h_g$ and $|Q_d| \mid h_g$. Hence an embedding can be constructed by simple duplication of the host. $\square$

Theorem 7.5
If h_g is a power of two, then an optimal embedding always exists into any hypercube.

Proof
If $h_g = 2^k$, and $d \leq k$, then the host is not bigger than one column of the guest. Hence a simple duplication technique can cover the guest with copies of the host.

If $h_g = 2^k$, and $d > k$, then we consider the $(2^k, 2^{d-k})$-grid which is a subgraph of the host. A simple duplication technique can then cover the guest with copies of the host. $\square$

Theorems 7.1, 7.4 and 7.5, cover all possibilities for the form of h_g and constitute the main result of this paper. Corollaries 7.2, 7.3 slightly simplify the third condition.

8 Conclusions

We have described a novel grid to hypercube embedding construction which is simple and optimal with respect to load and dilation for all except perhaps finitely many instances for each grid height when the load is at least four.

The method also yields optimal embeddings into the torus, under constraints on the relative heights of the torus and grid. For embeddings into the cylinder the same constraints are necessary and we may have to extend the cylinder somewhat beyond the minimum necessary to accommodate the grid.

Open problems include the removal of the restrictions on load four and above hypercube embeddings and the question as to whether optimal load, dilation one, hypercube embeddings exist for loads two and three. It may also be possible to push the results we have obtained here further for cylindrical and toroidal hosts.

Acknowledgment

We are indebted to the referees for thorough readings and for their many valuable suggestions and corrections.

References

[BMS92] Said Bettayeb, Zevi Miller, and I. Hal Sudborough. Embedding grids into hypercubes. *Journal of Computer and System Sciences*, 45(3):340–366, December 1992.

[CCCM96] M. Y. Chan, F. Chin, C. N. Chu, and W. K. Mak. Dilation-5 embedding of 3-dimensional grids into hypercubes. *Journal of Parallel and Distributed Computing*, 33(1):98–106, 25 February 1996.

[Cha91] Mee Yee Chan. Embedding of grids into optimal hypercubes. *SIAM Journal on Computing*, 20(5):834–864, October 1991.

[Ell91] J. A. Ellis. Embedding rectangular grids into square grids. *IEEE Transactions on Computers*, 40(1):46–52, January 1991.

[Ell96] J. A. Ellis. Embedding grids into grids: Techniques for large compression ratios. *Networks*, 27:1–17, 1996.

[ERS00] J. Ellis, F. Ruskey, and J. Sawada. Euclidean strings. In *Proceedings of the Eleventh Australasian Workshop on Combinatorial Algorithms*, pages 87–92. University of Newcastle, Australia, 2000.

[Lei92] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufman, San Mateo, 1992.

[MH90] R. G. Melhem and G. Y. Hwang. Embedding rectangular grids into square grids with dilation two. *IEEE Transactions on Computers*, 39(12):1446–1455, December 1990.

[MS94] Z. Miller and I. H. Sudborough. Compressing grids into small hypercubes. *Networks*, 24:327–358, 1994.

[RS98] Markus Rottger and Ulf-Peter Schroeder. Embedding 2-dimensional grids into optimal hypercubes with edge-congestion 1 or 2. *Parallel Processing Letters*, 8(2):231–242, June 1998.

[RS01] Rottger and Schroeder. Efficient embeddings of grids into grids. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 108, 2001.

[RSU94] M. Roettger, U.-P. Schroeder, and W. Unger. Embedding 3-dimensional grids into optimal hypercubes. *Lecture Notes in Computer Science*, 805:81–94, 1994.

[SHSHV96] Hongfei Liu Shou-Hsuan S. Huang and Rakesh M. Verma. A new combinatorial approach to optimal embeddings of rectangles. *Algorithmica*, 16(2):161–180, Aug 1996.

[SS90] F. C. Sang and I. H. Sudborough. Embedding large meshes into small ones. In *IEEE International Conference on Circuits and Systems*, pages 323–326, 1990.