1985

Optimal Simulations Between Mesh-Connected Arrays of Processors

S. Rao Kosaraju

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Report Number:
85-561

TR 561

OPTIMAL SIMULATIONS BETWEEN MESH-CONNECTED ARRAYS OF PROCESSORS[1]

S. Rao Kosaraju[2]
Dept. of Computer Science
Johns Hopkins University
Baltimore, MD 21218

Mikhail J. Atallah[3]
Dept. of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract. Let G and H be two mesh-connected arrays of processors, where $G=g_1\times g_2\times\cdots\times g_t$, $H=h_1\times h_2\times\cdots\times h_d$, and $g_1\cdots g_t\le h_1\cdots h_d$. We consider the problem of simulating G by H, and we characterize in terms of the g_i's and h_i's the best possible simulation by giving such a simulation and proving its optimality in the worst-case sense. We also establish the same bound on the average cost of encoding the edges of G as distinct paths in H.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems; G.2.2 [Graph Theory]: Graph Embedding Problems

Genaral Terms: Algorithms, Theory

Additional Key Words and Phrases: Mesh-Connected Processor Arrays, Parallel Computation, Parallel Simulation, Graph Encodings

---

1 A preliminary and less general version the results of this paper appeared in the Proc. of the Eighteenth Annual ACM Symposium on Theory of Computing (May 1986). The preliminary version restricted the simulating mesh to have the same number of processors as the one being simulated and was therefore less general than the present paper.

2 This research was supported by the National Science Foundation under Grant DCR-856361.

3 This research was supported by the Office of Naval Research under Grants N00014-84-K-0502 and N00014-86-K-0689, and the National Science Foundation under Grant DCR-8451393, with matching funds from AT&T.

1. Introduction

A network of processors H is said to *m-simulate* another network G if and only if every step of G can be simulated with $O(m)$ steps of H. Note that if H can m-simulate G then any problem that G solves in time T can be solved by H in time $O(m \cdot T)$. Establishing simulation results is an important research topic, since it enables the transport of algorithms written for one network to another [3,5,10]. This paper deals with the problem of simulating a mesh $G=g_1\times\cdots\times g_t$ (called the *guest*) by another mesh $H=h_1\times\cdots\times h_d$ (called the *host*), where $|G|=g_1\cdots g_t\leq h_1\cdots h_d=|H|$. Since any mesh $x_1\times\cdots\times x_s$ is the same as the mesh $x_{\pi(1)}\times\cdots\times x_{\pi(s)}$ for any permutation π of $\{1,\cdots,s\}$, for the rest of this paper we assume, without loss of generality, that $h_1\geq\cdots\geq h_d$ and $g_1\geq\cdots\geq g_t$. We also assume, without loss of generality, that both G and H are d-dimensional (i.e. $t=d$). This is justified because we can always introduce additional dimensions of length one, e.g. if $t<d$ then we consider G to be the mesh $g_1\times\cdots\times g_d$ where $g_{t+1}=\cdots=g_d=1$.

The model of a d-dimensional mesh is introduced in Section 2. Sections 3 and 4 establish that mesh H can α-simulate mesh G where $\alpha=\max_{1\leq i\leq d}(g_{i+1}\cdots g_d\,/\,h_{i+1}\cdots h_d)^{1/i}$, and that this bound is optimal to within a constant factor. Throughout, we adopt the notational convention that, for $i=d$, $g_{i+1}\cdots g_d/h_{i+1}\cdots h_d=1$, which simplifies many definitions. In previous work, [1] had considered a special case of this problem, that in which $|H|=|G|$ and either H is a cube (i.e. $h_1=\cdots=h_d$) or G is a cube. It was shown in [1] that if G is a cube then H can h_1/g_1-simulate G, and that if H is a cube then H can 1-simulate G. Both of these results are special cases of our general simulation result, which constitutes a nontrivial generalization of the results of [1].

The result in Section 5 is relevant to the extensive work that has been done in the area of minimum-cost encodings of one graph in another graph [2,4,6-9]. The lower bound part of the simulation result of sections 3 and 4 implies that the *worst-case* cost of

encoding G in H is $\Omega(\alpha)$, where α is as defined above. It is then natural to ask whether there is an encoding of G in H whose *average* cost is $o(\alpha)$. We settle this issue by proving that any encoding of G in H must have average cost $\Omega(\alpha)$. Only special cases of this result were previously known: In [4], for the case of $G=\sqrt{n}\times\sqrt{n}$ and $H=n\times 1$, an optimal $\Omega(\sqrt{n})$ bound was established. In [9] this was generalized to the case $G=n^{1/d}\times\cdots\times n^{1/d}$ and $H=n\times 1\times\cdots\times 1$, and an optimal $\Omega(n^{1-1/d})$ bound established. Both of these results follow from our general result.

2. The d-Dimensional Mesh of Processors

In a d-dimensional mesh of processors, the processors operate synchronously and are positioned on an $h_1\times\cdots\times h_d$ grid, one processor per grid point. A processor is denoted by its position in the grid, a typical one being denoted by $(i_1,\cdots,i_d)$ where $1\le i_k\le h_k$ for every $k\in\{1,\cdots,d\}$. Processors $(i_1,\cdots,i_d)$ and $(j_1,\cdots,j_d)$ are *neighbors* if and only if $|i_1-j_1|+|i_2-j_2|+\cdots+|i_d-j_d|=1$. Processors $(i_1,\cdots,i_d)$ and $(j_1,\cdots,j_d)$ are *neighbors along dimension k* if and only if they are neighbors and $|i_k-j_k|=1$. Note that a processor cannot have more than $2d$ neighbors (processors at the boundary have fewer). A *step* of such a mesh consists either of each processor communicating with a neighbor by sending/receiving the contents of a register (a *data movement step*), or of each processor performing a computation within its own registers (a *computation step*). A *data movement* is a sequence of data movement steps. A processor has a fixed (i.e. $O(1)$) number of storage registers. Some researchers assumed that a register can store up to $\log n$ bits, while others limited the size of a register to $O(1)$ bits: our results hold for either model.

Two meshes A and B are *equivalent* if A can 1-simulate B and B can 1-simulate A. Note that "stretching" the dimensions of a mesh by constant factors results in an equivalent mesh, e.g. an $h_1\times h_2$ mesh and a $(c_1h_1)\times(c_2h_2)$ mesh can 1-simulate each other

if c_1 and c_2 are constants.

Throughout this paper d, the number of dimensions, is assumed to be a constant, i.e. $d=O(1)$. Since the case $d=1$ is trivial, we also assume that $d\geq 2$.

If every processor is viewed as a vertex of a graph and every communication line between two neighboring processors is viewed as an undirected edge, then a mesh can alternatively be viewed as an undirected graph.

In order to establish the simulation upper bounds, we will use the idea of *embedding* the guest G into the host H, i.e. assigning every processor of G to a processor of H which will mimic its behavior (i.e. simulate it) during the complete computation. A processor of H might be simulating more than one processor of G in this way, but (because of the storage limitation) it cannot simulate more than a constant number of processors of G. Although this type of embedding-based simulation will enable us to establish the desired upper bounds, our lower bound proof holds for any type of simulation, including simulations where the embedding changes dynamically during the computation, and simulations where more than one processor of H can simulate the same processor of G.

In this paper we have not considered the case $|H|=o(|G|)$ since in such a case, H cannot even store as much information as G because each processor of H has $O(1)$ storage registers. Relaxing the standard assumption of limited storage per processor gives rise to interesting questions that are not within the scope of this paper.

3. A Simulation Lemma

Here we establish a lemma which is crucial to the simulations of Section 4.

Lemma 3.1. Let $G=g_1\times\cdots\times g_d$ and $H=h_1\times\cdots\times h_d$, where $g_1\geq\cdots\geq g_d$, $h_1\geq\cdots\geq h_d$, and every g_i and h_i is a power of two. Suppose that G and H satisfy the following:

(i) $g_1g_2\cdots g_i\geq h_1h_2\cdots h_i$ for every $i\in\{1,\cdots,d-1\}$, and

(ii) $g_1 g_2 \cdots g_d = h_1 h_2 \cdots h_d$.

Then H can 1-simulate G.

Proof. The proof is by induction on d:

Basis. $d=2$. From (i) and (ii), we have $g_1 \geq h_1 \geq h_2 \geq g_2$. Embed G in H as follows. Partition H into h_2/g_2 ($=g_1/h_1 \geq 1$) rectangular slabs of dimensions $h_1 \times g_2$ each. Now, "snake" G through these slabs in the manner depicted in Fig. 1 (this is similar to the "folding" used in [2]). Because of the way G moves from one slab of H to the next, some of the processors of H are idle (those in the empty triangular regions in Fig. 1b), while other processors of H are each simulating two of G's processors. It is obvious that this embedding enables H to simulate one step of G with $O(1)$ of its own steps.

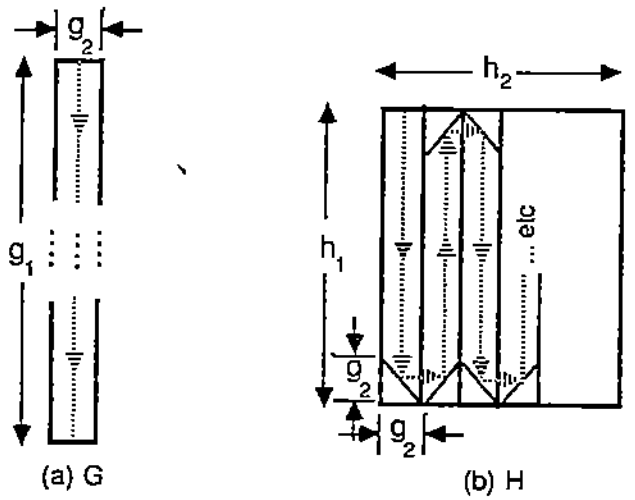

Figure 1. Illustrating the basis of induction.

Inductive Step. $d>2$. View $g_2 \times \cdots \times g_d$ as being the *base* of G, and view g_1 as being its *depth*. Let $\theta = g_2 \cdots g_d$, and note that $\theta = h_1 \cdots h_d / g_1 \leq h_2 \cdots h_d$ (since $g_1 \geq h_1$). The main idea is to partition $h_2 \times \cdots \times h_d$ into $h_2 \cdots h_d / \theta$ rectangular slabs of dimensionality $d-1$ and volume θ each; each slab will be a $\hat{h_2} \times \cdots \times \hat{h_d}$ mesh. Later on we shall precisely state what each $\hat{h_i}$ is, however for the time being we collect as we go along the conditions that we would like $\hat{h_2}, \cdots, \hat{h_d}$ to satisfy. Initially we require conditions (a)-(c) to hold:

(a) $\hat{h_2} \geq \cdots \geq \hat{h_d}$,

(b) $\hat{h_i} \leq h_i$ and $\hat{h_i}$ is a power of two for every $i \in \{2,3,\cdots,d\}$, and

(c) $\hat{h_2} \cdots \hat{h_d} = g_2 \cdots g_d = \theta$.

Conditions (b) and (c) guarantee that we can indeed partition $h_2 \times \cdots \times h_d$ into $(h_2/\hat{h_2}) \cdots (h_d/\hat{h_d})$ pieces each of which is $\hat{h_2} \times \cdots \times \hat{h_d}$ and has same volume ($=\theta$) as the base of G. (Condition (a) will be useful later, when we eventually require that a piece can 1-simulate the base of G.) This partition of $h_2 \times \cdots \times h_d$ into pieces induces a partition of H into *host slabs* each of which is $h_1 \times \hat{h_2} \times \cdots \times \hat{h_d}$, where we view h_1 as being the host slab's depth and $\hat{h_2} \times \cdots \times \hat{h_d}$ as being its base.

We would like the hosts's slab base to be capable of 1-simulating the base of G, i.e. $\hat{h_2} \times \cdots \times \hat{h_d}$ must be capable of 1-simulating $g_2 \times \cdots \times g_d$. The above conditions, along with (d) below, will enable us to use the induction hypothesis and embed the base of G in the base of the host slab such that the latter 1-simulates the former.

(d) $g_2 \cdots g_i \geq \hat{h_2} \cdots \hat{h_i}$ for every $i \in \{2,\cdots,d\}$.

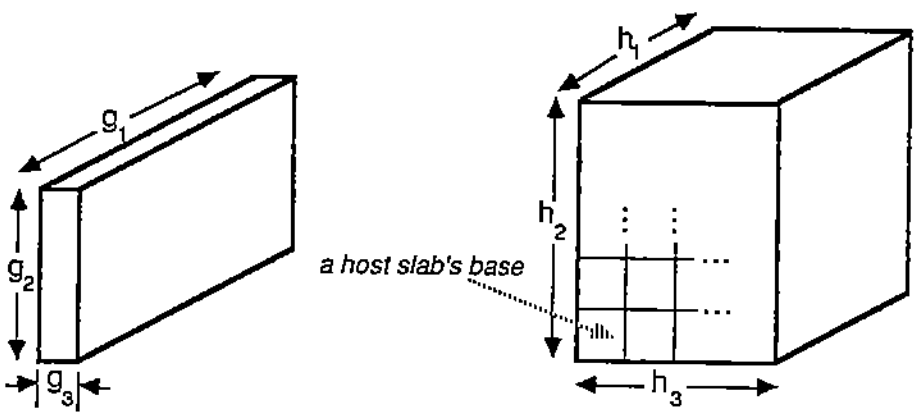

Figure 2. Illustrating the inductive step.

We now begin describing the embedding of G in H which enables the latter to 1-simulate the former. First embed the base of G in the base of a "corner" slab (e.g. in the lowest-leftmost slab base in Fig. 2b). Then, with its own base embedded in that of a slab, we snake G back and forth through the slabs until it is completely embedded in H. This

"snaking" is an obvious generalization of that done in Fig. 1b, and it requires that the depth of a host slab ($=h_1$) be no smaller than the longest dimension of its base ($=\hat{h}_2$) in order for G to shift smoothly from one slab to another (this condition was satisfied in Fig. 1b since we had $g_2 \leq h_1$). Therefore we need:

(e) $\hat{h}_2 \leq h_1$.

If we could find $\hat{h}_2, \cdots, \hat{h}_d$ to satisfy conditions (a)-(e) above, then the above embedding would clearly enable H to 1-simulate G. A data movement step in G along its first dimension can obviously be simulated in $O(1)$ steps in H, while a data movement step along any of the remaining $d-1$ dimensions of G can also be simulated in $O(1)$ steps in H because, by the induction hypothesis, the base of a slab in H can 1-simulate the base of G.

We are left with the problem of finding $\hat{h}_2, \cdots, \hat{h}_d$ such that conditions (a)-(e) above are satisfied. We choose $\hat{h}_2, \cdots, \hat{h}_d$ as follows:

$$\hat{h}_2 = \min(h_2 , g_2)$$

$$\hat{h}_3 = \min(h_3 , g_2 g_3 / \hat{h}_2)$$

$$\hat{h}_4 = \min(h_4 , g_2 g_3 g_4 / \hat{h}_2 \hat{h}_3)$$

$$\cdots$$

$$\hat{h}_d = \min(h_d , g_2 g_3 \cdots g_d / \hat{h}_2 \hat{h}_3 \cdots \hat{h}_{d-1}).$$

Now we prove that the above choice for $\hat{h}_2, \cdots, \hat{h}_d$ satisfies all the conditions (a)-(e), which completes the proof of Lemma 3.1. That every $\hat{h}_i$ is a power of two can easily be established by induction on i, since every g_i and h_i is also a power of two. Since $\hat{h}_i = \min(h_i, g_2 g_3 \cdots g_i / \hat{h}_2 \hat{h}_3 \cdots \hat{h}_{i-1})$, we have $\hat{h}_i \leq h_i$, establishing (b), and $\hat{h}_i \leq g_2 g_3 \cdots g_i / \hat{h}_2 \hat{h}_3 \cdots \hat{h}_{i-1}$, establishing (d). Condition (e) holds since $\hat{h}_2 \leq h_2$ and $h_2 \leq h_1$.

Proof of (a). We prove that $\hat{h}_i \geq \hat{h}_{i+1}$ by a case analysis:

Case 1. $\hat{h}_i = h_i$. This, together with $h_i \geq h_{i+1}$ (given) and $h_{i+1} \geq \hat{h}_{i+1}$ (property (b)), implies

$\hat{h_i} \geq \hat{h_{i+1}}$.

Case 2. $\hat{h_i} < h_i$. Hence

$\hat{h_i} = g_2 \cdots g_i / \hat{h_2} \cdots \hat{h_{i-1}}$ (by definition of $\hat{h_i}$)

$\geq g_i$ (by (d), $g_2 \cdots g_{i-1} \geq \hat{h_2} \cdots \hat{h_{i-1}}$)

$\geq g_{i+1}$ (by hypothesis)

$= g_2 \cdots g_{i+1} / \hat{h_2} \cdots \hat{h_i}$ (since $g_2 \cdots g_i = \hat{h_2} \cdots \hat{h_i}$)

$\geq \hat{h_2} \cdots \hat{h_{i+1}} / \hat{h_2} \cdots \hat{h_i}$ (by (d))

$= \hat{h_{i+1}}$.

Proof of (c). We must show that $\hat{h_2} \cdots \hat{h_d} = g_2 \cdots g_d$. Let $q = h_1 / g_1$, and note that Lemma 3.1's hypotheses imply the following:

$g_2 \cdots g_i \geq q h_2 \cdots h_i$ for $i \in \{2, \cdots, d-1\}$, and $g_2 \cdots g_d = q h_2 \cdots h_d$. (3.1)

If we can establish that $\hat{h_2} \cdots \hat{h_d} \geq q h_2 \cdots h_d$ then this fact, together with (3.1) above and with property (d), would imply the following: $q h_2 \cdots h_d \leq \hat{h_2} \cdots \hat{h_d} \leq g_2 \cdots g_d = q h_2 \cdots h_d$. This would imply that $\hat{h_2} \cdots \hat{h_d} = g_2 \cdots g_d$. Therefore it suffices to show that $\hat{h_2} \cdots \hat{h_d} \geq q h_2 \cdots h_d$. Actually we show, by induction on i, that $\hat{h_2} \cdots \hat{h_i} \geq q h_2 \cdots h_i$ for every $i \in \{2, \cdots, d\}$.

Basis. $i=2$. Recall that $\hat{h_2} = \min(h_2, g_2)$. If $\hat{h_2} = h_2$ then surely $\hat{h_2} \geq q h_2$ (since $q \leq 1$). If $\hat{h_2} = g_2$ then (3.1), for $i=2$, implies that $\hat{h_2} \geq q h_2$.

Inductive Step. $\hat{h_{i+1}} = \min(h_{i+1}, g_2 \cdots g_{i+1} / \hat{h_2} \cdots \hat{h_i})$.

Case 1. $\hat{h_{i+1}} = h_{i+1}$. Then we have:

$\hat{h_2} \cdots \hat{h_{i+1}} = \hat{h_2} \cdots \hat{h_i} h_{i+1} \geq q h_2 \cdots h_i h_{i+1}$, where we used the induction hypothesis.

Case 2. $\hat{h_{i+1}} = g_2 \cdots g_{i+1} / \hat{h_2} \cdots \hat{h_i}$. Then

$\hat{h_2} \cdots \hat{h_{i+1}} = g_2 \cdots g_{i+1} \geq q h_2 \cdots h_{i+1}$, where we used (3.1).

Thus property (c) holds, which completes the proof of Lemma 3.1. □

4. The Main Simulation Result

In this section we exploit the above simulation lemma to prove the following theorem.

Theorem 1. Let $H=h_1\times\cdots\times h_d$ and $G=g_1\times\cdots\times g_d$, with $h_1\geq\cdots\geq h_d$, $g_1\geq\cdots\geq g_d$, and $h_1\cdots h_d\geq g_1\cdots g_d$. Then mesh H can α-simulate mesh G, where $\alpha=\max_i (g_{i+1}\cdots g_d / h_{i+1}\cdots h_d)^{1/i}$. (Recall our notational convention that, for $i=d$, $g_{i+1}\cdots g_d/h_{i+1}\cdots h_d=1$.) This bound is optimal to within a constant factor.

Subsection 4.1 proves a special case of the upper bound part of Theorem 1. Subsection 4.2 uses the result of subsection 4.1 to prove the upper bound part of Theorem 1. Subsection 4.3 proves the lower bound part of Theorem 1.

4.1. A Special Case

This subsection considers the special case when $|H|=|G|=n$.

Lemma 4.1. Let $H=h_1\times\cdots\times h_d$ and $G=g_1\times\cdots\times g_d$, with $h_1\geq\cdots\geq h_d$, $g_1\geq\cdots\geq g_d$, and $h_1\cdots h_d=g_1\cdots g_d=n$. Then mesh H can α-simulate mesh G, where $\alpha=\max_i (g_{i+1}\cdots g_d / h_{i+1}\cdots h_d)^{1/i}$.

The rest of this subsection proves the above lemma.

For the time being, assume that every g_i and h_i is a power of two (later in this subsection, we remove this restriction). Let $\hat{\alpha}$ be the smallest power of two which is greater than or equal to α. It clearly suffices to show that H can $\hat{\alpha}$-simulate G. We define an intermediate mesh I:

(1) $I=n\times 1\times\cdots\times 1$ if $\hat{\alpha}g_1>n$,

(2) $I=(\hat{\alpha}g_1)\times\cdots\times(\hat{\alpha}g_m)\times\beta\times 1\times\cdots\times 1$ if $\hat{\alpha}g_1\leq n$, where $\beta\leq\hat{\alpha}g_{m+1}$ and $(\hat{\alpha}g_1)\cdots(\hat{\alpha}g_m)\beta=n$.

That H can $\hat{\alpha}$-simulate G would clearly follow if we could establish the following two claims:

Claim 4.1. *H* can 1-simulate *I*.

Claim 4.2. *I* can $\hat{\alpha}$-simulate *G*.

Proof of Claim 4.1: It is easy to verify that *I* and *H* satisfy the hypothesis of Lemma 3.1, with *I* playing the role of *G*. □

Proof of Claim 4.2: The proof relies on $\hat{\alpha}$ being a power of two, and is by induction on *d*.

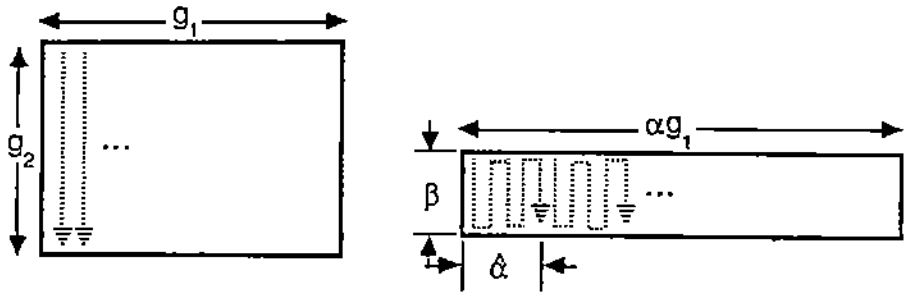

Figure 3. Illustrating how the columns of *G* are embedded into *I*.

Basis. d=2. We begin with the case when $\hat{\alpha}g_1 \le n$, i.e. $I=(\hat{\alpha}g_1)\times\beta$. *G* consists of g_1 adjacent columns of length g_2 each (see Fig. 3a). Now, snake these columns one after the other in *I*, as depicted in Fig. 3b. Note that each snaked column occupies a horizontal width of $g_2/\beta=\hat{\alpha}$ columns in *I*. A data movement step between adjacent processors in the same column of *G* can clearly be simulated with $O(1)$ steps of *I*. It is trivial to design a data movement taking $O(\hat{\alpha})$ steps on *I* and which simulates a data movement step between adjacent processors on the same row of *G*.

If $\hat{\alpha}g_1 > n$, then $I=(g_1g_2)\times 1$ and we can go through the same simulation as for the first case, except that g_2 and 1 now play the roles that (respectively) $\hat{\alpha}$ and β played in the previous case. Hence *I* can g_2-simulate *G*. Since $\hat{\alpha}g_1 > g_1g_2$, we have $\hat{\alpha} > g_2$ and hence *I* can $\hat{\alpha}$-simulate *G*.

Inductive Step. We begin with the case (2), when $\hat{\alpha}g_1 \le n$. That is, we want to show that $I=(\hat{\alpha}g_1)\times\cdots\times(\hat{\alpha}g_m)\times\beta\times 1\times\cdots\times 1$ can $\hat{\alpha}$-simulate $G=g_1\times\cdots\times g_d$, where $g_1\ge\cdots\ge g_d$, $\beta\le\hat{\alpha}g_{m+1}$ and $(\hat{\alpha}g_1)\cdots(\hat{\alpha}g_m)\beta=g_1\cdots g_d$. Divide *I* along its first dimension (i.e. using

hyperplanes orthogonal to first dimension) into g_1 consecutive submeshes (which we call *I-chunks*) each of which is an $\hat{\alpha}\times(\hat{\alpha}g_2)\times\cdots\times(\hat{\alpha}g_m)\times\beta\times 1\times\cdots\times 1$ mesh. Similarly divide G along its first dimension into g_1 consecutive submeshes (which we call *G-chunks*) each of which is a $1\times g_2\times\cdots\times g_d$ mesh. We shall use each I-chunk to $\hat{\alpha}$-simulate a G-chunk (more on how this is done will be said later). Of course, for this simulation, the G-chunks are assigned to the I-chunks in consecutive order. First observe that two processors of G that are neighbors along G's first dimension are simulated by two processors of I that are in two consecutive I-chunks, and that one data movement step in G between such processors can be simulated in I by a data movement in time proportional to the width of an I-chunk along its first dimension, i.e. $O(\hat{\alpha})$ (we omit the detailed specification of this easy data movement). We still need to show that a data movement step of G along its second, or third, ..., or d-th dimension, can also be simulated by $O(\hat{\alpha})$ steps of I. Since each such data movement is between processors in the same G-chunk, it suffices to show that an I-chunk can $\hat{\alpha}$-simulate a G-chunk. To prove this, we introduce a new d-dimensional mesh J, where:

$$J = (\hat{\alpha}g_2)\times\cdots\times(\hat{\alpha}g_m)\times(\hat{\alpha}\beta)\times 1\times\cdots\times 1 \qquad \text{if } \beta\le g_{m+1}, \text{ and}$$

$$J = (\hat{\alpha}g_2)\times\cdots\times(\hat{\alpha}g_m)\times(\hat{\alpha}g_{m+1})\times(\beta/g_{m+1})\times 1\times\cdots\times 1 \qquad \text{if } \beta>g_{m+1}.$$

Note that if $m=d-1$ then surely $\beta\le g_{m+1}$ holds, by the definition of β in mesh I. Otherwise note that $\beta/g_{m+1}\le\hat{\alpha}g_{m+2}$, since $\beta\le\hat{\alpha}g_{m+1}$. By the induction hypothesis for Claim 4.2, J can $\hat{\alpha}$-simulate a G-chunk ($=g_2\times\cdots\times g_d$). Therefore it suffices to show that an I-chunk can 1-simulate J, and this follows from Lemma 3.1.

We now consider the case (1), $\hat{\alpha}g_1>n$. Partition G and I into chunks as in the first case. An I-chunk is now an $(n/g_1)\times 1\times\cdots\times 1$ mesh and hence it can n/g_1-simulate a G-chunk (because a 1-dimensional array of x processors can always x-simulate any other x-processor mesh by circularly rotating the data in $O(x)$ time for every step of the other mesh). Since the width of an I-chunk along its first dimension is also n/g_1, it follows that

I can n/g_1-simulate G. Since $\hat{\alpha} > n/g_1$, I can $\hat{\alpha}$-simulate G. □

This proves Claim 4.2. Thus H can α-simulate G when $|H|=|G|$ and every g_i and h_i is a power of two.

Suppose not all the g_i's and h_i's are powers of two. It is easy to find a mesh $A=a_1\times\cdots\times a_d$ such that (i) every a_i is a power of two, (ii) $2^{-1}h_i\le a_i\le 2h_i$ for every $i\in\{1,\cdots,d\}$, and (iii) $2^{-1}h_1\cdots h_i\le a_1\cdots a_i\le 2h_1\cdots h_i$ for every $i\in\{1,\cdots,d\}$. (To obtain such a mesh A, for $i=1,\cdots,d$ let a_i be either the smallest power of two $\ge h_i$, or the largest power of two $\le h_i$, choosing the alternative that preserves property (iii).) Observe that (ii) implies that A and H are equivalent (i.e. can 1-simulate each other). Let $B=b_1\times\cdots\times b_d$ be to G what A is to H. Since A and B are equivalent to H and G, respectively, it suffices to prove that A can α-simulate B. First observe that the number of processors of A and B are equal to within a multiplicative constant of (at most) four, so that one of them can be "scaled up" to have the same number of processors as the other (by multiplying by 1,2, or 4 its largest dimension). Suppose this has already been done (i.e. $|A|=|B|$). Then A and B are still equivalent to H and G, respectively, and moreover every a_i and b_j is a power of two. Consequently, A can μ-simulate B where $\mu=\max_i (b_{i+1}\cdots b_d/a_{i+1}\cdots a_d)^{1/i}$. Since $\mu=\Theta(\alpha)$, it follows that A can α-simulate B, which completes the proof of Lemma 4.1.

It should be clear, by the above proof, that Lemma 4.1 is still true when we have $|H|=\Theta(|G|)$ instead of $|H|=|G|$.

4.2. Upper Bound Proof for General Case

This subsection generalizes the result of Lemma 4.1 to the case when $|H|\ge|G|$, thus establishing the upper bound part of Theorem 1. First, observe that there exists an index k and an integer $\hat{h}$ such that

(i) $h_k\ge\hat{h}\ge h_{k+1}$ if $k<d$, and $h_d\ge\hat{h}$ if $k=d$, and

(ii) $2^{-k}|G| \le \hat{h}^k h_{k+1} \cdots h_d \le |G|$ if $k<d$, and $2^{-d}|G| \le \hat{h}^d \le |G|$ if $k=d$.

(To find such a k and $\hat{h}$, try to find such an $\hat{h}$ for k=1, then for k=2, ...etc until you succeed.)

Let $\hat{H}=\hat{h}\times\cdots\times\hat{h}\times h_{k+1}\times\cdots\times h_d$ if $k<d$, and $\hat{H}=\hat{h}\times\cdots\times\hat{h}$ if $k=d$. Notice that, in either case, (ii) implies that $|\hat{H}|=\Theta(|G|)$. Note that H can 1-simulate $\hat{H}$, since $\hat{H}$ is a submesh of H. Hence to prove that H can α-simulate G, it suffices to prove that $\hat{H}$ can α-simulate G. We prove it for $k<d$, $k=d$ being very similar. Since $|\hat{H}|=\Theta(|G|)$, Lemma 4.1 implies that $\hat{H}$ can γ-simulate G, where

$$\gamma = \max\left(\max_{1\le i<k} (g_{i+1}\cdots g_d/\hat{h}^{k-i}h_{k+1}\cdots h_d)^{1/i} , \max_{k\le i\le d} (g_{i+1}\cdots g_d/h_{i+1}\cdots h_d)^{1/i} \right). \qquad (4.1)$$

It suffices to show that $\gamma = O(\alpha)$. We distinguish two cases, depending on the outcome of the "max" on the right-hand side of equation (4.1).

Case 1. For some j, $1\le j<k$, we have

$$\begin{aligned}
\gamma &= (g_{j+1}\cdots g_d/\hat{h}^{k-j}h_{k+1}\cdots h_d)^{1/j} \\
&= \Theta\left((|\hat{H}|\, g_{j+1}\cdots g_d/|G|\,\hat{h}^{k-j}h_{k+1}\cdots h_d)^{1/j} \right) \qquad (\text{since } |\hat{H}|=\Theta(|G|)) \\
&= \Theta\left(\hat{h}/(g_1\cdots g_j)^{1/j} \right)
\end{aligned}$$

Substituting for $\hat{h}$ in the above gives

$$\begin{aligned}
\gamma &= O\left((|\hat{H}|/h_{k+1}\cdots h_d)^{1/k}(g_1\cdots g_j)^{-1/j} \right) \\
&= O\left((|\hat{H}|/h_{k+1}\cdots h_d)^{1/k}(g_1\cdots g_k)^{-1/k} \right) \qquad (\text{since } (g_1\cdots g_j)^{-1/j}\le(g_1\cdots g_k)^{-1/k}) \\
&= O\left((|G|/h_{k+1}\cdots h_d)^{1/k}(g_1\cdots g_k)^{-1/k} \right) \qquad (\text{since } |\hat{H}|=\Theta(|G|)) \\
&= O(\alpha) \qquad (\text{by the definition of } \alpha)
\end{aligned}$$

Case 2. For some j, $k\le j\le d$, we have

$$\begin{aligned}
\gamma &= (g_{j+1}\cdots g_d/h_{j+1}\cdots h_d)^{1/j} \\
&\le \max_i (g_{i+1}\cdots g_d/h_{i+1}\cdots h_d)^{1/i} = \alpha
\end{aligned}$$

This completes the proof of the upper bound part of Theorem 1.

4.3. Lower Bound Proof

Now we establish the lower bound part of Theorem 1 by proving that α is, to within a constant factor, the best bound achievable by any simulation of G by H. Note that the simulations we gave so far have the property that a processor q of G is simulated by exactly one processor p of H, and that p simulates q throughout the computation. Our lower bound proof holds not only for such "embedding-like" simulations, but also for any simulation where every processor p (there may be many such p's) that is simulating q's condition at a certain instant of time, must store within its registers a complete description of q's condition at that instant of time. Thus a processor of G can be "simulated" by many processors of H at any particular instant of time, and this assignment of processors can change dynamically for the class of simulations that the lower bound holds for.

Let there exist a β-simulation of G by H, and focus on the situation in G at some instant of time t during the simulation (any t will do). If the condition of a processor q in G at time t depends on the condition, at time $t-g_i$, of a set Q of processors in G, then $|Q| \geq (g_i)^i g_{i+1} \cdots g_d$ (because every processor located at a distance less than g_i can influence q). Since H β-simulates G, the condition of a processor p in H which simulates q's condition at time t, depends on the condition of a set P of processors in H, where $|P| = O((\beta g_i)^i h_{i+1} \cdots h_d)$ (because any processor located at a distance greater than this number of steps cannot influence p). In addition, for H to properly simulate G, we must have $|P| = \Omega(|Q|)$. Thus

$(\beta g_i)^i h_{i+1} \cdots h_d = \Omega((g_i)^i g_{i+1} \cdots g_d)$. Consequently

$\beta = \Omega((g_{i+1} \cdots g_d \,/\, h_{i+1} \cdots h_d)^{1/i})$, for every $1 \leq i \leq d$. Thus

$$\beta = \Omega(\max_i (g_{i+1} \cdots g_d \,/\, h_{i+1} \cdots h_d)^{1/i}) = \Omega(\alpha),$$

which establishes the lower bound. □

5. An Encoding Lower Bound

For any graph H, let $Paths(H)$ denote the set of all paths in H. An *encoding* [9] of a guest graph $G=(V_G,E_G)$ in a host graph $H=(V_H,E_H)$ is a one-to-one mapping

$$\varepsilon: E_G \rightarrow Paths(H)$$

such that the mapping ε induces another one-to-one mapping

$$\mu: V_G \rightarrow V_H;$$

in other words edges e_1 and e_2 of G share an endpoint if and only if paths $\varepsilon(e_1)$ and $\varepsilon(e_2)$ share an endpoint in H.

The *worst-case* cost of the encoding ε is

$$WCOST(\varepsilon) = \max_{e \in E_G} length(\varepsilon(e))$$

where $length(\varepsilon(e))$ denotes the length of the path $\varepsilon(e)$.

The *average-case* cost of the encoding ε is

$$ACOST(\varepsilon) = |E_G|^{-1} \sum_{e \in E_G} length(\varepsilon(e))$$

Here we are interested in the case where G is a $g_1 \times \cdots \times g_d$ mesh and H is a $h_1 \times \cdots \times h_d$ mesh, with $h_1 \geq \cdots \geq h_d$, $g_1 \geq \cdots \geq g_d$, and

$$|H| = |V_H| = h_1 \cdots h_d \geq g_1 \cdots g_d = |V_G| = |G|.$$

The embeddings of G in H that we used in the simulation results of sections 3 and 4 are not encodings in the above sense, since we allowed them to map $O(1)$ vertices (edges) of G into a single vertex (path) of H (whereas in an encoding the mapping is one-to-one). Of course, the embeddings of sections 3 and 4 are quite appropriate for simulation purposes, because they capture the facts that (i) the dimensions of the mesh can be stretched/shrunk by a constant factor without any change in its computational power, and that (ii) for simulation purposes it is perfectly acceptable for one host processor to simulate more than one guest processors. Moreover, the lower bound part of the

proof of Theorem 1 can easily be seen to hold for encodings and it leads to a proof of the fact that for any encoding ε of G in H, we have

$$WCOST(\varepsilon) = \Omega(\alpha)$$

where α is as in Theorem 1. In addition, the proof of the simulation result of sections 3 and 4 can be viewed as a proof that $g_1 \times \cdots \times g_d$ can be *encoded* in $(bh_1) \times \cdots \times (bh_d)$ at a *WCOST* equal to $c\alpha$, where b and c are constants. To prove this, let e be the largest number of processors of G that are simulated by a single processor of H; then "expand" each vertex of H into a $b \times \cdots \times b$ mesh where $b^d \geq e$, ...etc). It is obvious that there is a tradeoff between the constants b and c, and to study this tradeoff is an interesting research issue, but one which is beyond the scope of this paper. Reference [2] investigates this tradeoff for the problem of encoding a 2-dimensional rectangle in a 2-dimensional square.

The following theorem, which establishes a tight lower bound on the average cost, is the main result of this section.

Theorem 2. Let H and G be mesh graphs as in Theorem 1. Then any encoding of G in H must have average cost $\Omega(\alpha)$, where $\alpha = \max_i (g_{i+1} \cdots g_d \,/\, h_{i+1} \cdots h_d)^{1/i}$.

Proof. Let k be such that $\alpha = \max_i (g_{i+1} \cdots g_d \,/\, h_{i+1} \cdots h_d)^{1/i} = (g_{k+1} \cdots g_d \,/\, h_{k+1} \cdots h_d)^{1/k}$.

If $k=d$ then $\alpha=1$ and hence the theorem holds trivially, so from now on assume that $k<d$. We prove that, for every encoding ε of G in H, we have

$$ACOST(\varepsilon) = |E_G|^{-1} \sum_{e \in E_G} length(\varepsilon(e)) = \Omega(\alpha) \qquad (5.1)$$

Let $\varepsilon(E_G) \subseteq Paths(H)$ be defined as follows:

$$\varepsilon(E_G) = \bigcup_{e \in E_G} \{\varepsilon(e)\}.$$

Note that $|\varepsilon(E_G)| = |E_G|$, since $\varepsilon(E_G)$ is a set of paths, not edges. Now, for every edge $e \in E_H$, let $count(e)$ be the number of paths in $\varepsilon(E_G)$ which contain e. Observe that

$$\sum_{e \in E_G} length(\varepsilon(e)) = \sum_{e \in E_H} count(e).$$

From the above equality and the fact that $|E_G| = \Theta(|G|)$, it follows that to prove (5.1) it suffices to show that

$$\sum_{e \in E_H} count(e) = \Omega(\alpha |G|).$$

Now we introduce some additional terminology. If $G=(V,E)$ is a graph and $A \subseteq V$, let $\Gamma(A)$ denote the edges of G with one endpoint in A and the other endpoint in $V-A$. The function Γ captures the "exposure" of A in G [9]. Recall that $\mu: V_G \rightarrow V_H$ is the one-to-one mapping induced by $\varepsilon: E_G \rightarrow Paths(H)$. For every $v \in V_H$, $\mu^{-1}(v)=u$ if and only if $\mu(u)=v$ for $u \in V_G$ (if no such u exists then $\mu^{-1}(v)$ is undefined). If $A \subseteq V_G$, then $\mu(A)=\{\mu(v) : v \in A\}$. For $B \subseteq V_H$, $\mu^{-1}(B)=\{\mu^{-1}(v) : v \in B\}$.

We now prove that $\sum_{e \in E_H} count(e) = \Omega(\alpha |G|)$.

Let $x \leq h_k/2$, and let A denote an $x \times \cdots \times x \times h_{k+1} \times \cdots \times h_d$ mesh graph. (Recall that k is such that $\alpha = (g_{k+1} \cdots g_d / h_{k+1} \cdots h_d)^{1/k}$.) Let B_H denote the subset of V_H consisting of the vertices whose coordinates in H are of the form $(i_1, \cdots, i_k, 1, \cdots, 1)$.

Now, imagine keeping H in a fixed position in d-dimensional space, while moving A so that its dimensions remain parallel to those of H; for every $v \in B_H$, let A_v be obtained by positioning A with respect to H such that

(i) for every $i \in \{1, \cdots, d\}$ the i-th dimension of A is parallel to the i-th dimension of H, and

(ii) vertex v of H coincides with the vertex of A whose coordinates in A are $(x/2, \cdots, x/2, 1, \cdots, 1)$.

Note that the position of some of the A_v's may be such that some of their vertices are "outside" of H (e.g. when v is at the boundary of H). The union of all of the A_v's defines a mesh $H' = (h_1+x) \times \cdots \times (h_k+x) \times h_{k+1} \times \cdots \times h_d$ (for each of the first k dimensions of H, an

$x/2$ increase in the positive direction and also in the negative direction). In what follows, we view H and the A_v's as particular subgraphs of H', positioned within H' as is implied by the previous discussion. Of course μ^{-1} remains such that, if vertex v is in H' but not in H, then $\mu^{-1}(v)$ is undefined. In addition, if edge e is in H' but not in H, then obviously $count(e)=0$.

For simplicity of notation, in what follows we use A_v to denote both the subgraph and its vertex set (the context will resolve the ambiguity). In particular, $\Gamma(A_v)$ denotes the set of edges of H' that have exactly one endpoint in A_v.

Proposition 5.1. For every edge e in H' we have

$$|\{v : e \in \Gamma(A_v)\}| = O(x^{k-1}).$$

Proof. The number of such vertices v is no larger than the surface area of an $x \times \cdots \times x$ (k-dimensional) cube, which is $O(x^{k-1})$. □

This proposition implies that

$$\sum_{e \in H'} count(e) = \Omega(x^{-k+1} \sum_{v \in B_H} \sum_{e \in \Gamma(A_v)} count(e)) \qquad (5.2)$$

Also, observe that

$$\sum_{e \in \Gamma(A_v)} count(e) \geq |\Gamma(\mu^{-1}(A_v))| \qquad (5.3)$$

because if (say) y edges leave $\mu^{-1}(A_v)$ in G, then surely at least y paths of $\varepsilon(E_G)$ leave A_v in H'.

Substitution of (5.3) in (5.2) results in

$$\sum_{e \in H'} count(e) = \Omega(x^{-k+1} \sum_{v \in B_H} |\Gamma(\mu^{-1}(A_v))| \qquad (5.4)$$

Now, observe that the definitions of α and k imply that

$(g_{k+1} \cdots g_d \,/\, h_{k+1} \cdots h_d)^{1/k} \geq (g_k \cdots g_d \,/\, h_k \cdots h_d)^{1/(k-1)}$, which implies

$(h_k)^k \geq (g_{k+1} \cdots g_d \,/\, h_{k+1} \cdots h_d)(g_k)^k$.

Applying the definition of α to this inequality results in $\alpha g_k \le h_k$, and hence $\alpha g_k/2 \le h_k/2$. Therefore we can choose $x=\alpha g_k/2$. Doing so results in $|A_v| = (\alpha g_k/2)^k h_{k+1} \cdots h_d = 2^{-k}(g_k)^k g_{k+1} \cdots g_d$. Together with the fact that $|\mu^{-1}(A_v)| \le |A_v|$, this implies that we can use Lemma 5.2 (given below) with $l=k$, $0<c\le 2^{-k}$, and $S=\mu^{-1}(A_v)$. Therefore (5.4) becomes

$$\sum_{e \in H'} count(e) = \Omega\Big(x^{-k+1} \sum_{v \in B_H} |\mu^{-1}(A_v)|/g_k \Big)$$

For every $u \in V_G$, $\mu(u)$ is contained in $(2x)^k$ of the A_v's, because A_v contains $\mu(u)$ whenever one of the first k coordinates of $\mu(u)$ differs from the same coordinate of v by an amount of x or less. This implies that $\sum_{v \in B_H} \mu^{-1}(A_v) = (2x)^k |G|$. Therefore

$$\sum_{e \in H'} count(e) = \Omega\big(x^{-k+1} x^k |G|/g_k \big) = \Omega\big(x |G|/g_k \big).$$

Since $x=\alpha g_k/2$, Theorem 2 follows. □

Lemma 5.2. Let S be any subset of V_G such that $|S| = c(g_l)^l g_{l+1} \cdots g_d$, for some $0<c\le 1$ and $1 \le l \le d$. Then $|\Gamma(S)| \ge (1-c)|S|/g_l$.

Proof. We begin by describing a transformation which, when applied to S, may modify its shape but does not cause any increase in $|\Gamma(S)|$. We call this transformation *COMPRESS*.

The *COMPRESS* transformation:

> Perform *COMPRESS*(1), followed by *COMPRESS*(2), $\cdots$, followed by *COMPRESS*(d), where *COMPRESS*(j) consists of "compressing" S along the j-th dimension, towards the lower end of that dimension (i.e. in the direction of lower values of the j-th coordinate). In other words, for every segment of G parallel to the j-th dimension (there are $g_1 \cdots g_{j-1} g_{j+1} \cdots g_d$ such segments and each of them has length g_j), "slide" the points of S on that segment so that they occupy adjacent positions at the beginning of that segment. To see that *COMPRESS*(j) does not

increase $|\Gamma(S)|$, simply note that (i) after compression along one segment has occurred, only one of the compressed points on that segment has a neighbour along the j-th dimension that is not in S, whereas there were ≥ 1 such points before the compression along that segment, and (ii) if H_1 and H_2 are adjacent segments containing n_1 and (respectively) n_2 points of S, then after the *COMPRESS*(j) there are exactly $|n_2-n_1|$ edges between H_1 and H_2 joining a point in S to one not in S, whereas there were $\geq |n_2-n_1|$ such edges before *COMPRESS*(j). Also observe that once *COMPRESS*(j) has been done, S remains "compressed" along the j-th dimension after we perform *COMPRESS*$(j+1), \cdots$, *COMPRESS*(d).

For the rest of this proof we assume that S has already been *COMPRESS*ed. Now, partition G into g_d slices each of which is $g_1 \times \cdots \times g_{d-1}$, and let G_i denote the i-th slice. That is, G_i contains all the vertices of G whose coordinates are of the form $(j_1, \cdots, j_{d-1}, i)$. Let S_i be the projection of $S \cap G_i$ onto hyperplane G_1, i.e. $S_i = \{(i_1, \cdots, i_{d-1}, 1) : (i_1, \cdots, i_{d-1}, i) \in S\}$.

Fact 5.3. $S_{g_d} \subseteq \cdots \subseteq S_1$.

Proof of Fact 5.3. We show that $S_j - S_i = \emptyset$ for any j, i such that $g_d \geq j > i \geq 1$. Suppose to the contrary that vertex p is in S_j but not in S_i. Then consider the segment of G that is parallel to the d-th dimension (and hence has length g_d) and that contains p: The j-th vertex on that segment is p and belongs to S, while the i-th vertex does not belong to S. Now $i < j$ contradicts the fact that S is *COMPRESS*ed along the d-th dimension. This proves Fact 5.3.

Consequently, $|S_1| \geq \cdots \geq |S_{g_d}|$. Hence

$$|S_1| \geq |S|/g_d \geq |S_{g_d}| \qquad (5.5)$$

For every subset X of V_G, we view $\Gamma(X)$ as consisting of two components: $\Gamma_1(X)$ and $\Gamma_2(X)$. $\Gamma_1(X)$ contains the edges of $\Gamma(X)$ that join a vertex in X to a vertex in $V_G - X$ along the d-th dimension, i.e. an edge of $\Gamma(X)$ which joins vertex $(i_1, \cdots, i_d)$ to vertex

$(j_1, \cdots, j_d)$ is in $\Gamma_1(X)$ if and only if $|i_d - j_d| = 1$. $\Gamma_2(X)$ contains the edges of $\Gamma(X)$ that join a vertex in X to one in $V_G - X$ along one of the other $d-1$ dimensions, i.e. $\Gamma_2(X) = \Gamma(X) - \Gamma_1(X)$. Observe that

$$|\Gamma_1(S)| = |S_1| - |S_{g_d}| \qquad (5.6)$$

The proof of Lemma 5.2 proceeds by induction on d, the number of dimensions. The basis ($d=1$) is trivial. For the inductive step ($d>1$), first observe that if $|S_{g_d}| = 0$ then (5.5) and (5.6) imply the following:

$$|\Gamma_1(S)| = |S_1| \geq |S|/g_d \geq |S|/g_l \geq (1-c)|S|/g_l$$

and hence the lemma holds. Therefore we henceforth assume that $|S_{g_d}| > 0$.

We now distinguish two cases.

Case I. $|S_1| > (g_l)^l g_{l+1} \cdots g_{d-1}$, in which case (5.5) and (5.6) imply that

$$|\Gamma_1(S)| = |S_1| - |S_{g_d}| > (g_l)^l g_{l+1} \cdots g_{d-1} - |S|/g_d = |S|/(cg_d) - |S|/g_d$$

$$= c^{-1}(1-c)|S|/g_d \geq (1-c)|S|/g_d \geq (1-c)|S|/g_l$$

and hence the lemma holds.

Case II. $|S_1| \leq (g_l)^l g_{l+1} \cdots g_{d-1}$, in which case for every S_i there exists a c_i, $0 < c_i \leq 1$, such that

$|S_i| = c_i (g_l)^l g_{l+1} \cdots g_{d-1}$. Hence we can use the induction hypothesis and obtain

$$|\Gamma_2(S_i)| \geq (1-c_i)|S_i|/g_l$$

We therefore have:

$$|\Gamma_2(S)| = |\Gamma_2(\bigcup_i G_i \cap S)| = |\bigcup_i \Gamma_2(G_i \cap S)| = \sum_{i=1}^{g_d} |\Gamma_2(S_i)| \geq (g_l)^{-1} \sum_{i=1}^{g_d} (1-c_i)|S_i|.$$

This implies that

$$|\Gamma(S)| \geq |\Gamma_1(S)| + (g_l)^{-1} \sum_{i=1}^{g_d} (1-c_i)|S_i| = |S_1| - |S_{g_d}| + (g_l)^{-1} \sum_{i=1}^{g_d} (1-c_i)|S_i|$$

Acknowledgement. The authors are extremely grateful to the referees for their careful reading and many insightful comments which significantly improved the presentation.

References

1. Atallah, M. J., Simulations Between Mesh-Connected Processor Arrays, *Proc. 23nd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, October 1985. (Also Purdue Univ. Computer Science Tech. Rept. #528, July 1985.)

2. Aleliunas, R., and Rosenberg, A. L., On embedding rectangular grids in square grids, *IEEE Trans. on Computers*, C-31 (1982), pp. 907-913.

3. Bhatt, S., Chung, F., Leighton F. T., and Rosenberg, A. L., Optimal Simulations of Tree Machines, *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986, pp. 273-282.

4. DeMillo, R. A., Eisenstat, S. C., and Lipton, R. J., Preserving average proximity in arrays, *Comm. ACM*, 21 (1978), pp. 228-231.

5. Leiserson, C. E., "Fat-trees: universal networks for hardware-efficient supercomputing," *Proc. of the 1985 IEEE International Conf. on Parallel Processing*, 1985, pp. 393-402.

6. Rosenberg, A. L., Preserving proximity in arrays, *SIAM J. Comput.*, 4 (1975), pp. 443-460.

7. Rosenberg, A. L., Data encodings and their costs, *Acta Inform.*, 9 (1978), pp. 273-292.

8. Rosenberg, A. L., Encoding data structures in trees, *J. Assoc. Comput. Mach.*, 26 (1979), pp. 668-689.

9. Rosenberg, A. L., and Snyder, L., Bounds on the costs of data encodings, *Math. Syst. Theory*, 12 (1978), pp. 9-39.

10. Siegel, H. J., A model of SIMD machines and a comparison of various interconnection networks, *IEEE Trans. on Computers*, C-28 (1979), pp. 907-917.