INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Optimal Trade-Offs Between Size and Slowdown for Universal Parallel Networks*

Friedhelm Meyer auf der Heide[†‡]

Martin Storch[†] **Rolf Wanka**[§]

TR-96-052

December 1996

Abstract

A parallel processor network is called n-universal with slowdown s, if it can simulate each computation of each constant-degree processor network with n processors with slowdown s. We prove the following lower bound trade-off: For each constant-degree n-universal network of size m with slowdown s, $m \cdot s = \Omega(n \log m)$ holds. Our trade-off holds for a very general model of simulations. It covers all previously considered models and all known techniques for simulations among networks. For $m \geq n$, this improves a previous lower bound by a factor of $\log \log n$, proved for a weaker simulation model. For $m < n$, this is the first non-trivial lower bound for this problem. In this case, this lower bound is asymptotically tight.

* Supported by DFG-Sonderforschungsbereich 376 "Massive Parallelität," by DFG Leibniz Grant Me 872/6-1, and by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT). A preliminary version of this paper appeared in: *Proc. 7th ACM Symposium on Parallel and Algorithms and Architectures (SPAA), pp. 119-128; 1995.* The full version will appear in: *Mathematical Systems Theory (MST).*

[†]Heinz Nixdorf Institute and Dept. of Mathematics and Computer Science, Paderborn University, 33095 Paderborn, Germany

[‡]email: fmadh@uni-paderborn.de.

[§]International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704-1198, USA, email: wanka@icsi.berkeley.edu. On leave from Heinz Nixdorf Institute and Dept. of Mathematics and Computer Science, Paderborn University, 33095 Paderborn, Germany, email: wanka@uni-paderborn.de.

1 Introduction

The Problem. In the past few years, the model of *parallel processor networks* has received much attention for solving time-critical tasks [11]. In this model of computation, the *processors* $P_1, \ldots, P_n$ are interconnected via a fixed *communication graph* $M = (\{P_1, \ldots, P_n\}, E)$. The edges of the graph represent *communication links*. Processors may communicate messages along these links. The number of links that a processor P has is called the *degree* of P. The degree of M is defined as the largest degree of its processors. For technological reasons, families of networks are required where the degree is bounded by a constant.

Usually, for special problems, special networks have been introduced. Famous constant-degree networks are the meshes, the butterfly network, the cube-connected cycles network, the shuffle-exchange network, and expander networks, to list only a few of them.

In order to compare the computation power of different networks, many simulations of a network M_1, the *guest*, on another network M_2, the *host*, are proposed, for many interesting networks M_1, M_2. We refer to these as *special simulations*.

As it is not feasible to tailor topologies of networks to specific process graphs or communication patterns of parallel programs, Galil and Paul [6] have asked for constant-degree *universal* networks that are able to simulate all constant-degree networks of size n without slowing down the computation time substantially. More specifically, let $\mathcal{U}$ be the class of all constant-degree networks with n processors. A constant-degree network M is called *n-universal with slowdown s*, if for each $G \in \mathcal{U}$, M can simulate T steps of G in at most $s \cdot T$ steps.

Previous Work. One approach for special simulations is the concept of *embeddings*. The processors of the guest are statically mapped to the processors of the host. For this concept, many results are known. See [16] for an overview.

With respect to universal parallel networks, Galil and Paul [6] show that each network M of size m that can sort n numbers in time $sort(n, m)$ is n-universal with slowdown $O(sort(n, m))$. E.g., this means that constant-degree networks like the shuffle-exchange network and the cube-connected cycles network, each of size n, are n-universal with slowdown $O(\log n \cdot (\log \log n)^2)$ using the algorithm presented in [5].

A more general approach is the concept of *dynamic simulations* (also sometimes called *emulations with redundancy*). Such simulations allow that single guest processors are simulated at several host processors, and that the set of simulating processors may vary during the simulation. A model of this type was first considered by Meyer auf der Heide in [13].

For special simulations, there are several results indicating the strength of these simulations, see [9, 18, 7, 8, 1]. E.g., the $\sqrt{n} \times \sqrt{n}$-mesh can be simulated on an n-processor butterfly network with constant slowdown [9], whereas an embedding has slowdown $\Omega(\log n)$, as shown in [4, 3].

A more drastic difference is known for universal networks: In [14], an $n^{1+\varepsilon}$-processor universal network is presented which has constant slowdown only. The same result holds when oblivious computations of the complete network of size n are simulated, instead of bounded degree networks, as also shown in [14]. On the other hand, if only embeddings are allowed, universal networks with constant slowdown have exponential size (upper and lower bound), as shown in [13]. In [13], a butterfly-like network of size $n^{1+\varepsilon}$ is presented that is n-universal

with slowdown $O(\log\log n)$. A similar result is obtained in [8]. In [8], a network of size n is constructed that can simulate each planar network of size n with slowdown $O(\log\log n)$. For restricted classes of bounded degree networks (those with polynomial spreading function, i. e., networks where the size of the t-neighborhood of each node is bounded by a polynomial in t), constant slowdown simulations even only need $O(n \cdot \text{polylog } n)$ size universal networks [15].

Lower bounds for dynamic simulations between special networks are shown in [9]: If, e. g., an m-processor butterfly network simulates an expander network of size n, it has a slowdown of $\Omega(\frac{n}{m}\frac{\log m}{\log\log m})$. Further results show that meshes of size m are not able to simulate a variety of networks of size n with the load-induced slowdown of $\frac{n}{m}$ only. For proving these properties, congestion- and diameter-based arguments are used. Rappoport shows in [17] that a multi-butterfly network of size n cannot be simulated efficiently by a butterfly network of size $O(n^\varepsilon)$, $\varepsilon > 0$.

Further lower bounds are shown in [10], where the communication bandwidth of guest and host are investigated as criteria to exceed the load-induced bound on the slowdown.

All these techniques are not strong enough to prove lower bounds for universal networks. E. g., no non-trivial lower bound for simulating an explicit network is known in case of an expander host (which might be a good choice for a universal network).

In [13], a lower bound on the slowdown of a universal network of size $m \geq n$ is shown using a somewhat weaker model of simulation (which still captures all known simulations for the case that the host is not smaller than the guest). It shows that slowdown s can only be achieved if

$$m \cdot s = \Omega\Big(n \cdot \frac{\log n}{\log\log n}\Big) \ .$$

In case of simulating non-oblivious computations of the complete network of size n, $s = \Omega(\log n)$ holds, independent of m [14].

With the techniques from [14], it is possible to show the following upper bound trade-off: For each $\ell \geq 1$, there is a universal network of size $n \cdot \ell$, whose slowdown fulfills: $s \cdot \log \ell = O(\log n)$. Closing the gap between the lower bound and the upper bound trade-off is still an open problem.

New Results. We present a lower bound on the trade-off between host size and slowdown of universal networks. It is shown for any n-universal network M of size m with slowdown s that

$$m \cdot s = \Omega(n \log m) \ .$$

Our lower bound proof even holds if only computations of length $\lceil 2\sqrt{\log m} \rceil$ have to be simulated. Note that looking at too short computations is not sufficient, because the techniques from [14] easily imply that a constant-degree network of size $2^{O(t)} \cdot n$ (consisting of n constant-degree trees of depth t) suffices to simulate all length t computations of all networks from $\mathcal{U}$ with constant slowdown.

For $m \leq n$, our result is easily shown to be asymptotically tight (Section 2). For our lower bound presented in Section 3, the underlying simulation model is some kind of pebble game [9], the assumably most general simulation model currently known. The proof is a counting argument inspired by the lower bounds in [13]. It improves the previous results in the following ways:

1. The simulation model (see Section 3.1) is generalized: load > 1 is allowed, and steps may be simulated asynchronously.

2. It extends the lower bound to small hosts ($m \leq n$).

3. The factor $1/\log\log n$ is removed from the lower bound in the more general type of simulation, making the bound optimal for $m \leq n$.

The result can be interpreted in the following ways: Improving the lower bound trade-off of [13], we get for $m \geq n$, $m \cdot s = \Omega(n \log n)$. Thus, the size of a universal network with slowdown $O(1)$ is $m = \Omega(n \log n)$.

For small universal networks ($m \leq n$), we obtain a slowdown $s = \Omega(\frac{n}{m} \log m)$ that exceeds the load-induced slowdown of $\frac{n}{m}$ by a factor of $\log m$, the minimum diameter of any constant-degree network of size m. Also, as the matching upper bound in Section 2 is done by a static embedding, it shows that dynamic embeddings do not yield an increase in efficiency for universal networks if $m \leq n$, in contrast to the fact that they do increase the efficiency, if $m > n$ (see [14]).

2 Upper Bound

Before we focus on the lower bound, we show that networks that have good routing capabilities are also good small universal networks. We describe an easy, intuitive simulation. Note that each processor is allowed to communicate with at most one of its neighboring processors during a single time step.

Let G be an arbitrary network, and let each processor of G hold at most h packets each with a desired destination address. These packets have to be routed to their destinations. Let each processor be the destination of at most h packets. This routing problem is called an *h-h routing problem*. Let $route_G(h)$ denote the time to solve each h-h routing problem on G.

Theorem 2.1 *Let $m \leq n$, and M be an arbitrary constant-degree network of size m. M is an n-universal network with slowdown at most $O(route_M(\frac{n}{m}))$.*

Proof: Let $G \in \mathcal{U}$ be a network that has to be simulated by M. Let f be a mapping of the nodes of G to the nodes of M such that each node Q of M gets at most $\lceil \frac{n}{m} \rceil$ of the nodes of G. The simulation is step by step. Q simulates the internal computations of its guests sequentially. If a processor P of G wants to communicate with its neighbor P', the processor Q of M with $f(P) = Q$ generates a packet with destination $f(P')$. Obviously, the desired communication of the guest processors forms an $\lceil \frac{n}{m} \rceil$-$\lceil \frac{n}{m} \rceil$ routing problem. □

By Theorem 2.1, we can conclude in the following way that for $m \leq n$, the butterfly network of size m is n-universal with slowdown $O(\frac{n}{m} \log m)$. Because the guest has constant degree, the $\lceil \frac{n}{m} \rceil$-$\lceil \frac{n}{m} \rceil$ routing problem that may arise can be solved by routing $O(\frac{n}{m})$ permutations that depend on G only, and, therefore, are known in advance. For the necessary techniques, see [11, Section 3.2.2]. The off-line routing problem can be solved in time $O(\log m)$ ([19]). Note that because of the lower bound shown in Section 3, this slowdown is asymptotically optimal.

Theorem 2.1 is also true if the complete network is simulated. In contrast to the above construction, we now need an *online* routing algorithm for the $\lceil \frac{n}{m} \rceil$-$\lceil \frac{n}{m} \rceil$ relations, because they are no longer known in advance. There are several randomized routing algorithms known, e.g., for the butterfly network (see [11]). Efficient deterministic algorithms for the h-h routing problem are known, e.g., for the constant-degree network one obtains by applying Leighton's Columnsort approach [12] to the AKS sorting circuit [2] and using parallel sorting as routing mechanism.

3 Lower Bound

3.1 The Simulation Model

The simulation of T steps of a guest network $G \in \mathcal{U}'$ by T' steps on a host network M of size m is modeled by a pebble game. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of nodes of G, and let $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ be the set of processors of M. For every node P_i of G and every guest time step t, there are pebbles of type (P_i, t). A pebble of type (P_i, t) stands for the configuration of guest processor P_i at time t.

A simulation is performed as follows: At the beginning, each processor of M contains all the initial pebbles $(P_1, 0), (P_2, 0), \ldots, (P_n, 0)$. In every host time step, every processor Q of M can perform one of the following operations:

- Generate a pebble of type (P_i, t). This is allowed only if pebbles of the types $(P_i, t-1), (P_{i_1}, t-1), (P_{i_2}, t-1), \ldots, (P_{i_{c'}}, t-1)$ are contained in Q, where $\{P_{i_1}, P_{i_2}, \ldots, P_{i_{c'}}\}$ are the neighbors of P_i in G. These pebbles are called *predecessors* of (P_i, t).

- Send a copy of one arbitrary pebble contained in Q to a neighboring processor. Note that after this step both processors contain a pebble of that type. Pebbles do not get lost.

- Receive a pebble from a neighbor. Only one pebble can be received at one time step.

After the T' host time steps, pebbles of types $(P_1, T), \ldots, (P_n, T)$ must be generated on at least one processor, each. They are called *final pebbles*.

As both the host and the guest are constant-degree networks, it is not a restriction that in one step of the guest the next configuration of a P_i depends on all its neighbors, whereas the host processors are allowed to communicate with one neighbor only.

A *protocol S of a simulation* is a listing of the operations performed on every processor at every step.

For ease of explanation, let us make a couple of definitions for a given simulation protocol S. Remember that a processor keeps every pebble it generated or received for the entire simulation. Let $[n] := \{1, \ldots, n\}$, for all $n \in I\!N$. For $i \in [n]$, $t \in [T]$, let

- $\mathcal{Q}_S(i,t) := \{j \in [m] \mid Q_j$ contains a pebble of type (P_i, t) at the end of $S\}$.

- $\mathcal{Q}'_S(i,t) := \{j \in \mathcal{Q}_S(i,t) \mid Q_j$ generates a pebble of type $(P_i, t+1)$ in some step of $S\}$.

$\mathcal{Q}_S(i,t)$ is the *set of representatives of P_i at guest time t in simulation protocol S*, and $\mathcal{Q}'_S(i,t)$ is the *set of generators of P_i at guest time $t+1$ in simulation protocol S*.

Let d be the degree of the universal network M. For a guest graph $G \in \mathcal{U}'$, consider a simulation of T steps of G by T' steps of M. Then $s := T'/T$ is the *slowdown* and

$$k := s \cdot \frac{m}{n} = \frac{T' \cdot m}{T \cdot n}$$

is the *inefficiency* of the simulation. Note that even in the case of $m \geq n$, the slowdown $s = \frac{n}{m} \cdot k$ is at least 1.

A simulation protocol, or simply a simulation, is called *k-inefficient*, if it simulates a graph $G \in \mathcal{U}'$ on our fixed network M with inefficiency at most k.

Note that this computation model is very powerful because here it takes unit time only to transmit a full configuration of a processor along a link to a neighboring processor, and because the computation of the next configuration is also assumed to consume one step only. Lower bounds holding for this model clearly also hold for models where the bandwidth of the links is bounded.

3.2 The Lower Bound Proof

In this section, we will prove our main result:

Theorem 3.1 *If M is n-universal with slowdown s, then $m \cdot s = \Omega(n \cdot \log m)$.*

For our proof, we only consider computations of length $T \geq \lceil 2\sqrt{\log m} \rceil$ steps executed on regular networks with n processors having degree $c = 16$. Let this class of networks be denoted by $\mathcal{U}'$. Consider an m-processor network M with degree d that simulates all T-step computations of all networks from $\mathcal{U}'$ in at most $T' = T \cdot \frac{n}{m} \cdot k$ steps.

We have to show that k, the inefficiency, is $\Omega(\log m)$. We proceed as follows: We count the number $|\mathcal{U}'|$ of networks with n vertices of constant-degree c, and the number $D(k)$ of networks from $\mathcal{U}'$ that have a k-inefficient simulation by M. We have to determine the minimum k with $D(k) = |\mathcal{U}'|$. Counting $|\mathcal{U}'|$ is simple. It has already been done in [13]. Bounding the number $D(k)$ is a difficult task and will be done in the rest of this section.

Our approach to computing $D(k)$ is as follows: Consider a computation of length T on some network $G \in \mathcal{U}'$, and its k-inefficient simulation on M. Fix some $t_0 \leq T$. The only information about S we are going to consider is, for each $i \in \{1, \ldots, n\}$

- $\mathcal{Q}_S(i, t_0)$, the set of representatives of P_i at guest time step t_0, i. e., the set of processors that know the configuration of P_i after t_0 steps of the computation of M,
- one $b_i \in \mathcal{Q}_S(i, t_0)$ such that Q_{b_i} simulates the $(t_0 + 1)$st step of P_i, i. e., $b_i \in \mathcal{Q}'_S(i, t_0)$.

This information will be called a *fragment consistent with S*. More formally, a fragment is defined as follows:

Definition 3.2 *A* fragment *is a triple* $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ *with*

- $\mathcal{B} = (B_1, \ldots, B_n)$, $B_i \subseteq [m]$,
- $\mathcal{B}' = (b_1, \ldots, b_n)$, $b_i \in B_i$,
- $\mathcal{D} = (D_1, \ldots, D_n)$, $D_i = \{i' \in [n] \mid b_i \in B_{i'}\}$.

Given a simulation protocol S *of* T *steps of a graph* $G \in \mathcal{U}'$ *by* T' *steps of* M*, we say that a fragment* $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ *is* consistent with S*, if there is a* $t_0 \in [T]$ *such that* $\mathcal{B} = (\mathcal{Q}_S(1, t_0), \ldots, \mathcal{Q}_S(n, t_0))$ *and* $b_i \in \mathcal{Q}'_S(i, t_0)$ *for all* $i \in [n]$*.* t_0 *is called a* critical time step *of that fragment.*

Note that adding $\mathcal{D}$ to the fragment is redundant, because it is uniquely defined by $\mathcal{B}$ and $\mathcal{B}'$. We add it for convenience only.

Clearly, a fragment can be consistent with simulations of more than one network from $\mathcal{U}'$. The following lemma shows that it is consistent with not too many $G \in \mathcal{U}'$. For a fixed fragment $(\mathcal{B}, \mathcal{B}', \mathcal{D})$, let X denote the number of different graphs from $\mathcal{U}'$ that have a simulation on M consistent with $(\mathcal{B}, \mathcal{B}', \mathcal{D})$. X is called the *multiplicity* of $(\mathcal{B}, \mathcal{B}', \mathcal{D})$.

Lemma 3.3 *For a fixed fragment* $(\mathcal{B}, \mathcal{B}', \mathcal{D})$*,*

$$X \leq \prod_{i=1}^{n} \binom{|D_i|}{\frac{1}{2}c}$$

Proof: Assume that the graphs $G \in \mathcal{U}'$ are represented as directed graphs, where each node has $\frac{1}{2}c$ incoming and $\frac{1}{2}c$ outgoing edges. Note that such an orientation of the edges exists because c is even and G is c-regular (it can be obtained by walking along a Eulerian Tour). G is completely described by listing, for all P_i of G, the $\frac{1}{2}c$ edges leaving P_i.

Let $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ be a fragment of a simulation of G by M. Because for some $t_0 \in [T]$, processor Q_{b_i} of M generates a pebble of type $(P_i, t_0{+}1)$, Q_{b_i} has to contain, for all neighbors $P_{i'}$ of P_i, pebbles of type $(P_{i'}, t_0)$. Therefore, $i' \in D_i$ for all these $P_{i'}$. Thus there are only $\binom{|D_i|}{c/2}$ possibilities to choose the edges leaving P_i. This proves the lemma. □

Our next goal is to find a set of fragments with the following properties:

- The set is small.
- For (almost) each $G \in \mathcal{U}'$, it contains a fragment $(\mathcal{B}, \mathcal{B}', \mathcal{D})$.
- This fragment has small multiplicity, i. e., the D_i's are small.

The following Main Lemma states the properties of such a set of fragments. In order to keep the set small, we only consider simulations of networks $G \in \mathcal{U}'$ that contain a certain subgraph G_0. This property guarantees a certain minimum spreading of information of G (along the edges of G_0). Therefore, it also imposes some structure on the simulation, which is of central importance for the proof of the Main Lemma.

Let $\mathcal{U}[G_0] := \{G \in \mathcal{U}' \mid G_0 \text{ is a subgraph of } G\}$. For a set A, $I\!P(A)$ denotes its power set.

Lemma 3.4 (Main Lemma) *There is a graph G_0 with n nodes of degree 12 such that the following holds: There are constants $q, r \in \mathbb{N}$, $0 < \gamma < 1$, a set $\mathcal{A} \subseteq \mathbb{P}([m])^n$ with $|\mathcal{A}| \leq 2^{r \cdot n \cdot k}$ such that every k-inefficient simulation protocol of a graph $G \in \mathcal{U}[G_0]$ is consistent with a fragment $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ with*

1. $\mathcal{B} = (B_1, B_2, \ldots, B_n) \in \mathcal{A}$
2. $\sum_{i=1}^{n} |B_i| \leq q \cdot n \cdot k$
3. $|D_i| \leq \dfrac{n}{\sqrt{m}}$ *holds for at least* γn *many* $i \in [n]$.

The elements of $\mathcal{A}$ describe parts of all possible k-inefficient simulation protocols. As $|\mathcal{A}|$ is not too large, this helps us to keep the number of graphs small that can be simulated k-inefficiently. The construction of the appropriate subgraph G_0 and the proof of the Main Lemma is the most involved part of the lower bound proof. This will be done in Subsection 3.3.

Assume Lemma 3.4 to be true for a moment. Let $\mathcal{G}(k) \subseteq \mathcal{U}[G_0]$ denote the set of graphs, for which a k-inefficient simulation exists. Then we conclude:

Lemma 3.5

$$|\mathcal{G}(k)| \leq |\mathcal{A}| \cdot (q \cdot k)^n \cdot \frac{n^{\frac{c-12}{2} \cdot n}}{m^{\frac{1}{2}\gamma \cdot \frac{c-12}{2} \cdot n}}$$

Proof: Consider the fragments consistent with k-inefficient simulations for graphs $G \in \mathcal{G}(k)$, and chosen according to Lemma 3.4. Let Y be the number of such fragments. Let the multiplicity of any such fragment be bounded by X. By Lemma 3.4, we know that every graph $G \in \mathcal{G}(k)$ has at least one such fragment. Thus we conclude $|\mathcal{G}(k)| \leq X \cdot Y$. The proof is completed by showing the Proposition 3.6.

Proposition 3.6

(a) $$Y \leq |\mathcal{A}| \cdot (q \cdot k)^n$$

(b) $$X \leq \frac{n^{\frac{c-12}{2} \cdot n}}{m^{\frac{1}{2}\gamma \cdot \frac{c-12}{2} \cdot n}}$$

Proof:
(a) Consider the fragments $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ chosen according to Lemma 3.4. Because $\mathcal{B} \in \mathcal{A}$, there are only $|\mathcal{A}|$ possible choices for $\mathcal{B} = (B_1, B_2, \ldots, B_n)$. Now let $\mathcal{B}$ be fixed. As $b_i \in B_i$ for all $i \in [n]$, we have only

$$\prod_{i=1}^{n} |B_i| \leq \Big(\frac{1}{n} \cdot \sum_{i=1}^{n} |B_i|\Big)^n \leq \Big(\frac{q \cdot n \cdot k}{n}\Big)^n = (q \cdot k)^n$$

possible choices for the $(b_1, b_2, \ldots, b_n) = \mathcal{B}'$. $\mathcal{B}$ and $\mathcal{B}'$ already define $\mathcal{D}$.

(b) Let $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ be a fixed fragment chosen according to Lemma 3.4. We want to bound X, the multiplicity of $(\mathcal{B}, \mathcal{B}', \mathcal{D})$, i.e., count the graphs that have k-inefficient simulations consistent with $(\mathcal{B}, \mathcal{B}', \mathcal{D})$.

Every graph $G \in \mathcal{G}(k)$ contains the fixed regular subgraph G_0 with degree 12. Thus, G can be uniquely determined by the *residual graph* $G' := G \setminus G_0$. G' is regular with even degree $c - 12$. Combining Lemma 3.3 and the Main Lemma, part (3), we get:

$$\begin{aligned} X &\leq \prod_{i=1}^{n} \binom{|D_i|}{\frac{c_i-12}{2}} \\ &\leq \binom{\frac{n}{\sqrt{m}}}{\frac{c_i-12}{2}}^{\gamma \cdot n} \cdot \binom{n}{\frac{c_i-12}{2}}^{(1-\gamma)\cdot n} \\ &\leq \left(\frac{n}{\sqrt{m}}\right)^{\frac{c_i-12}{2}\cdot\gamma\cdot n} \cdot n^{\frac{c_i-12}{2}\cdot(1-\gamma)\cdot n} \\ &= \frac{n^{\frac{c-12}{2}\cdot n}}{m^{\frac{1}{2}\cdot\frac{c-12}{2}\cdot\gamma\cdot n}} \end{aligned}$$

□

This finishes the proof of Lemma 3.5. ■

Proof: (of Theorem 3.1):
In [13], it is shown that there is a $\delta > 0$ with

$$|\mathcal{U}[G_0]| \geq n^{\frac{c-12}{2}n} \cdot 2^{-\delta \cdot n} \quad .$$

In Lemma 3.5, we have upper bounded the number of graphs in $\mathcal{U}[G_0]$, for which a k-inefficient simulation exists. If our network M is universal for $\mathcal{U}'$ with slowdown $\frac{n}{m} \cdot k$, it is also universal for $\mathcal{U}[G_0]$. Thus, $|\mathcal{G}(k)| = |\mathcal{U}[G_0]|$ must hold. We get:

$$n^{\frac{c-12}{2}n} \cdot 2^{-\delta\cdot n} \leq 2^{r\cdot n\cdot k} \cdot (q \cdot k)^n \cdot \frac{n^{\frac{c-12}{2}\cdot n}}{m^{\frac{1}{2}\gamma\cdot\frac{c-12}{2}\cdot n}}$$

Let $r' \in I\!N$ such that

$$(q \cdot k)^n \cdot 2^{\delta\cdot n} \cdot 2^{r\cdot n\cdot k} \leq 2^{r'\cdot n\cdot k} \quad .$$

It follows

$$m^{\frac{1}{2}\gamma\cdot\frac{c-12}{2}\cdot n} \leq 2^{r'\cdot n\cdot k}$$

and consequently

$$k \geq \frac{1}{2r'} \cdot \gamma \cdot \frac{c-12}{2} \cdot \log m = \Omega(\log m).$$

■

3.3 Proof of the Main Lemma

We describe a T-step computation of a network G by the following graph:

Definition 3.7 *Let $G = (\mathcal{P}, E)$ be a graph. The* dependency graph *of T steps of G is the graph $\Gamma_G := (V_{\Gamma_G}, E_{\Gamma_G})$ with vertices $V_{\Gamma_G} = \mathcal{P} \times \{0, \ldots, T\}$ and directed edges $((P,t),(P',t+1))$ for all $t \in \{0, \ldots, T-1\}$, if $P = P'$ or $\{P, P'\} \in E$. Then (P,t) is a* predecessor *of $(P', t+1)$, denoted with $(P,t) \longrightarrow (P', t+1)$.*

(P,t) is an i-th predecessor *of $(P', t+i)$, if there is a directed path from (P,t) to $(P', t+i)$ in Γ_G. We denote this with $(P,t) \stackrel{i}{\longrightarrow} (P', t+i)$.*

Note that for any subgraph $G_0 \subseteq G$, Γ_{G_0} is a subgraph of Γ_G.

We have to define a small class $\mathcal{A}$ of sets of representatives such that for every k-inefficient simulation protocol S of a graph $G \in \mathcal{U}[G_0]$, we can find a guest time step t_0 for which the tuple of sets of representatives for guest time step t_0, $(\mathcal{Q}_S(1,t_0), \mathcal{Q}_S(2,t_0), \ldots, \mathcal{Q}_S(n,t_0))$, belongs to $\mathcal{A}$. Clearly, choosing the $\mathcal{Q}_S(i,t_0)$ arbitrarily from the m processors would yield a too large set $\mathcal{A}$:

$$\prod_{i=1}^{n} \binom{m}{|\mathcal{Q}_S(i,t_0)|} \leq \prod_{i=1}^{n} m^{|\mathcal{Q}_S(i,t_0)|} = m^{\Theta(n \cdot k)}$$

We will reduce the number of choices by defining a small set R of nodes, and, for some x, embed their $(t_0 - x)$-pebbles randomly into M, i. e., choose the sets $\mathcal{Q}_S(i, t_0 - x)$ for each $i \in R$. R must be selected such that for each $i \in [n]$ there is $i_0 \in R$ such that $(P_{i_0}, t_0 - x) \stackrel{x}{\longrightarrow} (P_i, t_0)$. This implies that the (P_i, t_0)-pebbles cannot travel "too far" from their respective x-th predecessor of type $(P_{i_0}, t_0 - x)$, which makes counting significantly easier.

A pebble dependency $(P_u, t) \longrightarrow (P_v, t+1)$ always arises from an edge $\{P_u, P_v\} \in E_G$, unless $u = v$. As the only edges we know for sure are the edges of the fixed subgraph G_0, we manufacture the required paths of length x in Γ_G from the well known edges of its subgraph Γ_{G_0}.

The Definition of G_0

In this part of the proof, we will define the graph G_0. We will show some graph theoretic properties for its dependency graph Γ_{G_0} that we will make use of in the next subsection.

Definition 3.8 *Assume that n is a square of an integer $N = \sqrt{n}$. The graph $H = ([N] \times [N], E)$ with $E = \{\{(x,y),(x',y')\} \mid |x - x'| + |y - y'| = 1, \text{ and } x, x', y, y' \in [N]\}$ is called an* n-mesh.

If E also contains the edges $\{\{(x,1),(x,N)\}, \{(1,x),(N,x)\} \mid x \in [N]\}$, then H is called an n-torus.

For $a \leq N$, we define an (a,n)-multitorus *to be an n-torus in which each $a \times a$-submesh is extended by edges to form an $a \times a$-torus.*

For $0 < \alpha < 1$ and $\beta > 1$, a graph $G = (V, E)$ is said to be an (α, β)-expander, *if for all $A \subseteq V$ with $|A| \leq \alpha \cdot |V|$, the set of neighbors of A contains at least $\beta \cdot |A|$ nodes.*

Definition 3.9 *Let $a := \sqrt{\log m}$. W. l. o. g., we assume $n \geq 4a^2$ to be a square of an integer. We define our fixed subgraph as $G_0 := (V_0, E_0)$ with $V_0 = \mathcal{P}$ and edges $E_0 := E_1 \cup E_2$ where E_1 contains the edges of a $(2a, n)$-multitorus, and E_2 contains the edges of a 4-regular (α, β)-expander graph of size n, and $0 < \alpha < 1$, and $\beta > 1$. Every node of G_0 has degree 12.*

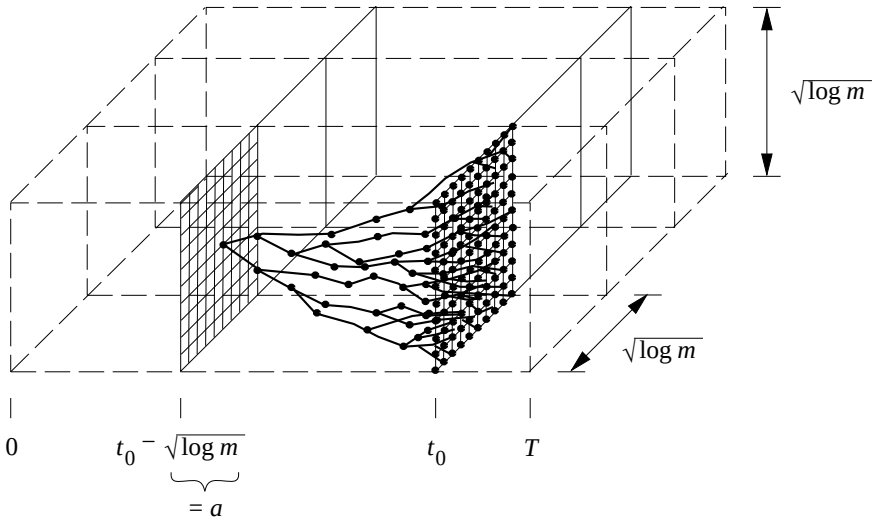

Figure 1: A dependency tree in Γ_{G_0}.

The expander properties will be used in Lemma 3.15 to prove part (3) of the Main Lemma.

W.l.o.g., we assume that n is a multiple of $4a^2$. We partition G_0 into $(4a^2)$-tori $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_h \subseteq G_0$ such that every node of G_0 is contained in one of these tori. Thus, $h \leq \frac{n}{4a^2}$. The diameter of each torus is a.

For the rest of this paragraph, it is convenient to identify a graph $G = (V_G, E_G)$ with its nodes. We simply write $v \in G$ instead of $v \in V_G$ and $|G| := |V_G|$.

Lemma 3.10 *For each $t > a$, for each $j \in [h]$, and each $P_i \in \mathcal{T}_j$, Γ_{G_0} contains a binary tree of size at most $48a^2$, rooted at $(P_i, t-a)$ with leaves $\mathcal{T}_j \times \{t\}$. It is called $T_{i,t}$, the* dependency tree rooted at $(P_i, t-a)$.

Proof: It can be shown that for any copy $\mathcal{T}$ of a $(4a^2)$-torus and a fixed vertex $P_i \in \mathcal{T}$, there is a dependency tree T rooted at $(P_i, 0)$ in $\Gamma_{\mathcal{T}}$ with the above properties. This is done by recursively partitioning the torus into 4 submeshes. Connect the center P_i of $\mathcal{T}$ to the centers of the 4 submeshes by paths in Γ_{G_0}. At the lowest level of the recursion, every node of $\mathcal{T}$ will be reached. Although this is not hard to prove, we will not give the details of the proof in this paper.

Because the torus $\mathcal{T}$ is symmetric, any $P_i \in \mathcal{T}$ can be chosen to be the root $(P_i, t-a)$ of the dependency tree. □

For example, we have illustrated a dependency tree in Figure 1.

Properties of Simulations of G_0

Definition 3.11 *For a given k-inefficient simulation protocol S of a graph $G \in \mathcal{U}[G_0]$, and for any $(P_i, t) \in \Gamma_G$, call $q_{i,t} := |\mathcal{Q}_S(i,t)|$ the* weight *of node (P_i, t). Let*

$$w_{i,t} := \sum_{(P_{i'}, t') \in T_{i,t}} q_{i',t'}$$

be the weight *of dependency tree $T_{i,t}$.*

As we mentioned before, we have to find a guest time step t_0 for which there are not too many choices for the sets of representatives $\mathcal{Q}_S(1,t_0), \mathcal{Q}_S(2,t_0), \ldots, \mathcal{Q}_S(n,t_0)$.

For some suitable $t_0 \in [T]$, we select nodes $r_1, \ldots, r_h \in [n]$ with $P_{r_j} \in \mathcal{T}_j$ for each $j \in [h]$. They form the previously mentioned set $R = \{r_1, \ldots, r_h\}$. The nodes in $\mathcal{R} := \{(P_i, t_0 - a) \mid i \in R\}$ are the roots of the dependency trees $T_{r_1,t_0}, \ldots, T_{r_h,t_0} \subseteq \Gamma_G$.

As every node of G is contained in one torus, the leaves of these h trees cover the entire set $\mathcal{P} \times \{t_0\}$. Thus for each $i \in [n]$, there is $j \in [h]$ such that $(P_{r_j}, t_0 - a) \xrightarrow{a} (P_i, t_0)$.

The number of pebbles used in the simulation of a graph G is at most the number of operations performed by the host M, namely $T' \cdot m = T \cdot n \cdot k$. Thus only k pebbles on average of any type (P_i, t) come up during the simulation. By averaging, we will prove the existence of suitable $t_0, r_1, \ldots, r_h$ such that the average weight of the respective dependency trees is $O(k \cdot |T_{r_j,t_0}|)$, and the average weight of any root of these trees is $O(k)$.

Lemma 3.12 *Consider a k-inefficient simulation protocol S of a graph $G \in \mathcal{U}[G_0]$. There is a set Z_S of guest time steps with $|Z_S| \geq T/4$, and, for each $t_0 \in Z_S$, there are $r_1, r_2, \ldots, r_h \in [n]$ with $P_{r_1} \in \mathcal{T}_1$, $P_{r_2} \in \mathcal{T}_2$, $\ldots$, $P_{r_h} \in \mathcal{T}_h$ such that the following inequalities hold (recall that $a = \sqrt{\log m}$):*

$$\sum_{j=1}^{h} q_{r_j,t_0-a} \leq 8 \cdot \frac{n}{a^2} \cdot k \tag{1}$$

$$\sum_{j=1}^{h} w_{r_j,t_0} \leq 384 \cdot n \cdot k \tag{2}$$

Proof: Assume $T \geq 2a$.

$$\begin{aligned}
&\sum_{t=a+1}^{T} \sum_{j=1}^{h} \sum_{P_i \in \mathcal{T}_j} w_{i,t} \\
&= \sum_{t=a+1}^{T} \sum_{P_i \in G} q_{i,t} \cdot |\{(v',t') \in \Gamma_G \mid (P_i,t) \in T_{v',t'}\}| \\
&\leq \sum_{t=a+1}^{T} \sum_{P_i \in G} q_{i,t} \cdot 48a^2 \leq 48a^2 \cdot \sum_{t=a+1}^{T} \sum_{P_i \in G} q_{i,t} \\
&\leq 48a^2 \cdot n \cdot k \cdot T \ ,
\end{aligned}$$

because $\sum_{t=1}^{T} \sum_{P_i \in G} q_{i,t} \leq m \cdot T' = n \cdot k \cdot T$.

Let

$$Z' := \{t \in \{a+1, \ldots, T\} \mid \sum_{j=1}^{h} \sum_{P_i \in \mathcal{T}_j} w_{i,t} \leq 48a^2 \cdot n \cdot k \cdot T \cdot \tfrac{4}{T-a}\} \ ,$$

and

$$Z'' := \{t \in \{a+1, \ldots, T\} \mid \sum_{P_i \in G} q_{i,t-a} \leq n \cdot k \cdot T \cdot \tfrac{4}{T-a}\} \ .$$

Then $|Z'| \geq \frac{3}{4}(T-a)$, and $|Z''| \geq \frac{3}{4}(T-a)$. Set $Z_S := Z' \cap Z''$. Verify that $|Z_S| \geq \frac{1}{2}(T-a) \geq T/4$. Let $t_0 \in Z_S$ be fixed. Then:

$$\begin{aligned} \sum_{j=1}^{h} \sum_{P_i \in \mathcal{T}_j} w_{i,t_0} &\leq 384a^2 \cdot n \cdot k \\ \sum_{P_i \in G} q_{i,t_0-a} &\leq 8n \cdot k \end{aligned}$$

For each $j \in [h]$, let

$$\begin{aligned} V_j' &:= \{P_i \in \mathcal{T}_j \mid \text{ there are } a^2 \text{ nodes } P_{i'} \in \mathcal{T}_j \text{ with } w_{i,t_0} \leq w_{i',t_0}\}, \\ V_j'' &:= \{P_i \in \mathcal{T}_j \mid \text{ there are } a^2 \text{ nodes } P_{i'} \in \mathcal{T}_j \text{ with } q_{i,t_0-a} \leq q_{i',t_0-a}\}. \end{aligned}$$

It follows $|V_j'| \geq \frac{3}{4}|\mathcal{T}_j| \geq 3a^2$ and $|V_j''| \geq 3a^2$. As $|\mathcal{T}_j| \geq 4a^2$ and $V_j', V_j'' \subset \mathcal{T}_j$, this implies that $|V_j' \cap V_j''| \geq 2a^2$. Choose $P_{r_j} \in V_j' \cap V_j''$. To complete the proof, we conclude:

$$\begin{aligned} \sum_{j=1}^{h} q_{r_j,t_0-a} \cdot a^2 \leq \sum_{P_i \in G} q_{i,t_0-a} \leq 8n \cdot k \\ \sum_{j=1}^{h} w_{r_j,t_0} \cdot a^2 \leq \sum_{j=1}^{h} \sum_{P_i \in \mathcal{T}_j} w_{i,t_0} \leq 384a^2 \cdot n \cdot k \end{aligned}$$

□

Properties (1) and (2) of the Main Lemma

With the help of Lemma 3.12, we can define the set $\mathcal{A}$ from the Main Lemma. Let $\mathcal{A} := \{(\mathcal{Q}_S(1,t_0), \ldots, \mathcal{Q}_S(n,t_0)) \mid G \in \mathcal{U}[G_0],\ S$ is a k-inefficient simulation protocol for G, $t_0 \in Z_S\}$. Remember that d is the (constant) degree of M; $q_{i,t}$, $w_{i,t}$ are as in Definition 3.11.

Lemma 3.13 *Consider any k-inefficient simulation protocol S of a graph $G \in \mathcal{U}[G_0]$. Then for every $t_0 \in Z_S$, and for every fragment $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ consistent with S with critical time step t_0, we have*

1. *$\mathcal{B} \in \mathcal{A}$*
2. *$\sum_{i=1}^{n} q_{i,t_0} \leq q \cdot n \cdot k$ holds for $q := 384$*
3. *$|\mathcal{A}| \leq 2^{rnk}$ holds for $r := 3472 + 384 \log d$.*

In particular, this shows that for each $t_0 \in Z_S$, $(\mathcal{Q}(1,t_0), \ldots, \mathcal{Q}(n,t_0))$ is a candidate for the sequence $\mathcal{B}$ satisfying (1) and (2) of the Main Lemma.

Proof:
Part (1): By the definition of $\mathcal{A}$, it is clear that each fragment $(\mathcal{B}, \mathcal{B}', \mathcal{D})$ consistent with S and with $\mathcal{B} = (\mathcal{Q}(1,t_0), \ldots, \mathcal{Q}(n,t_0))$ satisfies $\mathcal{B} \in \mathcal{A}$.

Part (2): Choose $t_0 \in Z_S$ and $r_1, \ldots, r_h$ according to Lemma 3.12. Because the nodes $\mathcal{P} \times \{t_0\} \subseteq \Gamma_G$ are the leaves of the dependency trees $T_{r_1,t_0}, \ldots, T_{r_h,t_0}$, it follows from Lemma 3.12 that

$$\sum_{i=1}^{n} q_{i,t_0} \leq \sum_{j=1}^{h} w_{r_j,t_0} \leq 384 \cdot n \cdot k.$$

Part (3): For any k-inefficient simulation protocol S of a graph $G \in \mathcal{U}[G_0]$, fix $t_0, r_1, \ldots, r_h$ as chosen according to Lemma 3.12. Let $R := \{r_1, \ldots, r_h\}$. Let $F = (V_F, E_F)$ be the forest $T_{r_1,t_0} \cup \cdots \cup T_{r_h,t_0} \subseteq \Gamma_G$. Every node of F has indegree 1 and outdegree at most 2, and $|V_F| \leq h \cdot 48a^2 \leq 12n$. Let $\mathcal{R}$ denote the set of roots of F.

As $\mathcal{P} \times \{t_0\} \subseteq V_F$, the product of the number of choices for $\mathcal{Q}(i,t)$, taken over all $(P_i,t) \in V_F$, is an upper bound on $|\mathcal{A}|$.

Proposition 3.14

(a) *(i)* *There are at most $2^{8 \cdot n \cdot k / \log m}$ possibilities to choose the weights $q_{i,t}$ for all $i \in R$.*

(ii) *Fix $i \in R$. If $q_{i,t}$ is fixed, then there are at most $\binom{m}{q_{i,t}}$ choices of a $\mathcal{Q}(i,t)$ that can appear in k-inefficient simulation protocols.*

(b) *(i)* *There are at most $2^{384 \cdot n \cdot k}$ possibilities to choose all weights $q_{i,t}$ for all $(P_i,t) \in V_F \setminus \mathcal{R}$.*

(ii) *For $(P_i,t) \in V_F \setminus \mathcal{R}$, let $(P_{f(i)}, t-1)$ be its predecessor in F. Assuming $\mathcal{Q}(f(i), t-1)$ being fixed, there are at most*

$$2^{q_{f(i),t-1}} \cdot 2^{2q_{i,t}} \cdot d^{q_{i,t}}$$

choices for $\mathcal{Q}(i,t)$ that can appear in k-inefficient simulation protocols.

Proof:
(a)(ii) is clear, (a)(i) and (b)(i) follow directly from Lemma 3.12. It remains to prove (b)(ii):

Pebbles of type (P_i,t) can only be generated on processors Q_j with $j \in \mathcal{Q}(f(i), t-1)$. From Q_j, they can be sent to neighboring processors of M. Thus every processor $Q_{j'}$, $j' \in \mathcal{Q}(i,t)$, must have a path to some Q_j, $j \in \mathcal{Q}(f(i), t-1)$, in M consisting of processors with index in $\mathcal{Q}(i,t)$ only.

It is easy to show that there are only

$$2^{q_{f(i),t-1}} \cdot 2^{2q_{i,t}} \cdot d^{q_{i,t}}$$

different subforests of fixed size $q_{i,t}$ in M such that every connected component contains a processor Q_j with $j \in \mathcal{Q}(f(i), t-1)$. □

Now it is easy to prove part(3) of Lemma 3.13:

- The number of choices for the sets $\mathcal{Q}(i,t)$, $i \in R$, is less than

$$\begin{aligned} & 2^{8\cdot n\cdot k/\log m} \cdot \prod_{i\in R} \binom{m}{q_{i,t}} \\ \leq\ & 2^{8\cdot n\cdot k} \cdot \prod_{i\in R} m^{q_{i,t}} \\ \leq\ & 2^{8\cdot n\cdot k} \cdot m^{\sum_{i\in R} q_{i,t}} \\ \leq\ & 2^{8\cdot n\cdot k} \cdot m^{8\cdot n\cdot k/\log m} \\ \leq\ & 2^{8\cdot n\cdot k} \cdot 2^{8\cdot n\cdot k} = 2^{16\cdot n\cdot k} \quad . \end{aligned}$$

- The number of choices for the sets $\mathcal{Q}(i,t)$, $(P_i,t) \in V_F \setminus \mathcal{R}$ is less than

$$\begin{aligned} & 2^{384\cdot n\cdot k} \cdot \prod_{(P_i,t)\in V_F\setminus\mathcal{R}} \left(2^{2q_{i,t}} \cdot d^{q_{i,t}} \cdot 2^{q_{f(i),t-1}}\right) \\ \leq\ & 2^{384\cdot n\cdot k} \cdot 2^{2\cdot 384\cdot n\cdot k} \cdot d^{384\cdot n\cdot k} \prod_{(P_i,t)\in V_F} 2^{2\cdot q_{i,t}} \\ \leq\ & 2^{(3456+384\log d)\cdot n\cdot k} \quad . \end{aligned}$$

The lemma is proved by multiplying these two numbers. □

Property (3) of the Main Lemma

We will show part (3) of the Main Lemma by making use of the (α,β)-expander in G_0, and the fact that for a given simulation protocol S, a suitable t_0 can be chosen from a large set Z_S.

Lemma 3.15 *Consider a k-inefficient simulation protocol S for a graph $G \in \mathcal{U}[G_0]$. There is a fragment $(\mathcal{B},\mathcal{B}',\mathcal{D})$ consistent with S at some critical time step $t_0 \in Z_S$ that satisfies property (3) of the Main Lemma with $\gamma = \frac{1}{2}\alpha\cdot(1-\frac{1}{\beta}) > 0$.*

This lemma completes the proof of the Main Lemma, because any such fragment with critical time step $t_0 \in Z_S$ satisfies properties (1) and (2) of the Main Lemma, according to Lemma 3.12.

Proof: By contradiction: Assume that the lemma is false. Let $t_0 \in Z_S$. We want to define a fragment $(\mathcal{B},\mathcal{B}',\mathcal{D})$ consistent with S with critical time step t_0. By the choice of t_0, $\mathcal{B} = (\mathcal{Q}(1,t_0),\ldots,\mathcal{Q}(n,t_0))$ is uniquely defined. We are still free to choose the generating pebbles $b_1 \in \mathcal{Q}'(1,t_0),\ldots,b_n \in \mathcal{Q}'(n,t_0)$ for $\mathcal{B}'$.

Let $\mathcal{P}(j,t) := \{i \in [n] \mid j \in \mathcal{Q}(i,t)\}$ for all $j \in [m]$, $t \in [T]$. The choice of the b_i defines $D_i := \mathcal{P}(b_i,t_0)$ for the sequence $\mathcal{D}$. By assumption, none of these choices for the b_i satisfies property (3) of the Main Lemma, thus for at least $(1-\gamma)\cdot n$ many $i \in [n]$:

$$\forall j \in \mathcal{Q}'(i,t_0): \quad |\mathcal{P}(j,t_0)| > \frac{n}{\sqrt{m}}$$

We call the corresponding pebbles (P_i,t_0) *heavy*.

Let us denote pebbles of types $(P_1, t), \ldots, (P_n, t)$ as *t-pebbles.*

Since $t_0 \in Z_S$, we can conclude from Lemma 3.13 that $\sum_{j=1}^{m} |\mathcal{P}(j, t_0)| = \sum_{i=1}^{n} q_{i,t_0} \leq 384nk$. By averaging, it follows that there are at most $\frac{384nk}{n/\sqrt{m}} = 384\sqrt{m} \cdot k$ processors $j \in [m]$ with $|\mathcal{P}(j, t_0)| > \frac{n}{\sqrt{m}}$. We call these processors t_0-*heavy.* A heavy generating pebble (P_i, t_0) can only emerge on the subset $\mathcal{Q}'(i, t_0)$ of the set of t_0-heavy processors, which is small.

The key idea of the proof is to see that, in order for the simulation to proceed, a lot of generating t_0-pebbles have to be produced in a certain host time frame, some of which must be heavy and cost one of the t_0-heavy processors one step. This is true for every $t_0 \in Z_S$, and the time frames for the different t_0's do not overlap. There are a lot of heavy t_0-pebbles to be produced on only few t_0-heavy processors, thus each time frame must be long.

Let $Z_S = \{t_1, t_2, \ldots, t_\ell\}$ with $\ell \geq T/4$ and, for all $j \in [\ell]$, $t_j < t_{j+1}$. We will divide the simulation time (host time) into parts.

Definition 3.16 *For $t \in [T]$, $\tau \in [T']$, let $E_t(\tau) := \{i \in [n] \mid$ after τ host time steps of the simulation, a generating pebble of type (P_i, t) exists$\}$.*
Of course, $E_t(\tau) \subseteq E_{t'}(\tau')$ for any $t' \leq t$ and $\tau' \geq \tau$. Let $e_t(\tau) := |E_t(\tau)|$. For $j \in [\ell]$, let

$$\tau_j := \min\{\tau \in [T'] \mid e_{t_j - 1}(\tau) \geq \alpha n\}$$

be the earliest time in the simulation at which at least αn different generating $(t_j - 1)$-pebbles exist.

Proposition 3.17 *$e_{t_j}(\tau_j) \leq \frac{\alpha}{\beta} \cdot n$ holds for every $j \in [\ell]$.*

Proof: Assume $e_{t_j}(\tau_j) > \frac{\alpha}{\beta} \cdot n$. Because τ_j is chosen as minimum of a set of suitable time steps, $e_{t_j - 1}(\tau_j - 1) < \alpha \cdot n$ holds. Thus, $e_{t_j}(\tau_j) < \alpha \cdot n$.

On the other hand, the t_j-pebbles can be generated only if the $(t_j - 1)$-pebbles of all of their neighbor nodes did exist after the previous simulation step. We apply the expansion property of G to conclude in contradiction that $e_{t_j - 1}(\tau_j - 1) \geq \beta \cdot e_{t_j}(\tau_j) > \beta \cdot \frac{\alpha}{\beta} \cdot n = \alpha \cdot n$. □

Thus, between host time steps τ_j and τ_{j+1}, at least $\alpha n - \frac{\alpha}{\beta} n = \alpha(1 - \frac{1}{\beta})n$ different generating t_j-pebbles have to be produced. Because $t_j \in Z_S$, at most $\gamma n = \frac{1}{2}\alpha(1 - \frac{1}{\beta})$ of these are not heavy, the remaining $\frac{1}{2}\alpha(1 - \frac{1}{\beta})$ heavy pebbles must be produced on the $384\sqrt{m} \cdot k$ many t_j-heavy processors of M. Therefore,

$$\tau_{j+1} - \tau_j \geq \frac{\frac{1}{2}\alpha(1 - \frac{1}{\beta}) \cdot n}{384\sqrt{m} \cdot k} \quad .$$

This is true for all $j \in [\ell]$, and $\ell \geq T/4$. Hence, we get:

$$\frac{n}{m} \cdot k \cdot T \geq T' \geq \frac{\frac{1}{2}\alpha(1 - \frac{1}{\beta}) \cdot n}{384\sqrt{m} \cdot k} \cdot \frac{T}{4}$$

$$k^2 \geq \frac{\alpha(1 - \frac{1}{\beta})}{3072} \cdot \sqrt{m}$$

and finally,

$$k = \Omega(m^{\frac{1}{4}}) \ .$$

This finishes the proof of Lemma 3.15. ■

4 Conclusions

We have seen that for each n-universal network M with size $m \leq n$, there is a "bad" network G of size n for which the simulation cannot perform better than a simple embedding on the butterfly network. The inefficiency is $\log m$, the diameter of M. This result was obtained by using a non-constructive counting method on the number of graphs that can be simulated with a given slowdown under the pebble game simulation model.

For universal networks, some open questions still remain. In the case of $m \geq n$, it is not known how many processors are needed for a simulation algorithm with slowdown $O(1)$. This paper shows that $m = \Omega(n \log n)$ in this case; in [14], it is shown that $m = O(n^{1+\varepsilon})$ for any $\varepsilon > 0$.

References

[1] Alf-Christian Achilles. Optimal emulation of meshes on meshes of trees. In *Proceedings of the International Conference on Parallel Processing (EURO-PAR)*, pages 193–204, 1995.

[2] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \cdot \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

[3] Sandeep N. Bhatt, Fan R. K. Chung, Jia-Wei Hong, F. Thomson Leighton, Bojana Obrenić, Arnold L. Rosenberg, and Eric J. Schwabe. Optimal emulations by Butterfly-like networks. *Journal of the ACM*, 43:293–330, 1996.

[4] Sandeep N. Bhatt, Fan R. K. Chung, Jia-Wei Hong, F. Thomson Leighton, and Arnold L. Rosenberg. Optimal simulations by Butterfly networks. In *Proceedings of the 20th ACM Symposium on Theory of Computing (STOC)*, pages 192–204, 1988.

[5] Robert Cypher and C. Greg Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. *Journal of Computer and System Sciences*, 47:501–548, 1993.

[6] Zvi Galil and Wolfgang Paul. An efficient general-purpose parallel computer. *Journal of the ACM*, 30:360–387, 1983.

[7] Christos Kaklamanis, Danny Krizanc, and Satish Rao. New graph decompositions and fast emulations in hypercubes and butterflies. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 325–334, 1993.

[8] Christos Kaklamanis, Danny Krizanc, and Satish Rao. Universal emulations with sublogarithmic slowdown. In *Proceeding of the 34th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 341–350, 1993.

[9] Richard Koch, Tom Leighton, Bruce Maggs, Satish Rao, and Arnold Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, pages 227–240, 1989.

[10] Clyde P. Kruskal and Kevin J. Rappoport. Bandwidth-based lower bounds on slowdown for efficient emulations of fixed-connection networks. In *Proceedings of the 6th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 132–139, 1994.

[11] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[12] Tom Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Transactions on Computers*, 34:344–354, 1985.

[13] Friedhelm Meyer auf der Heide. Efficiency of universal parallel computers. *Acta Informatica*, 19:269–296, 1983.

[14] Friedhelm Meyer auf der Heide. Efficient simulations among several models of parallel computers. *SIAM Journal on Computing*, 15:106–119, 1986.

[15] Friedhelm Meyer auf der Heide and Rolf Wanka. Time-optimal simulations of networks by universal parallel computers. In *Proceedings of the 6th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 120–131, 1989.

[16] Burkhard Monien and Hal Sudborough. Embedding one interconnection network in another. *Computing*, 7:257–282, 1990.

[17] Kevin J. Rappoport. On the slowdown of efficient simulations of multibutterflies on butterflies and butterfly-derived networks. In *Proceedings of the 8th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 176–182, 1996.

[18] Eric J. Schwabe. On the computational equivalence of hypercube-derived networks. In *Proceedings of 2nd ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 388–397, 1990.

[19] A. Waksman. A permuting network. *Journal of the ACM*, 15:159–163, 1968.