Lower Bounds on the Competitive Ratio for Mobile User Tracking and Distributed Job Scheduling

Noga Alon*
Department of Mathematics
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel-Aviv University
Tel Aviv, Israel

Gil Kalai†
Institute of Mathematics and Computer Science
Hebrew University
Jerusalem, Israel

Moty Ricklin
Computer Science Department
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel-Aviv University
Tel Aviv, Israel

Larry Stockmeyer
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA

Abstract

We prove a lower bound of $\Omega(\log n/\log\log n)$ on the competitive ratio of any (deterministic or randomized) distributed algorithm for solving the mobile user problem introduced by Awerbuch and Peleg [5, 6], on certain networks of n processors. Our lower bound holds for various networks, including the hypercube, any network with sufficiently large girth, and any highly expanding graph. A similar $\Omega(\log n/\log\log n)$ lower bound is proved for the competitive ratio of the maximum job delay of any distributed algorithm for solving the distributed scheduling problem of Awerbuch, Kutten and Peleg [4] on any of these networks. The proofs combine combinatorial techniques with tools from linear algebra and harmonic analysis and apply, in particular, a generalization of the vertex isoperimetric problem on the hypercube, which may be of independent interest.

Published in: *Theoretical Computer Science* 130 (1994), pp. 175–201,

A preliminary version of this paper appeared in: *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, 1992, pp. 334–343.

---

*Research supported in part by a United States Israeli BSF grant. Work performed in part while visiting Bellcore, Morristown, NJ.

†Work performed in part while visiting the IBM Almaden Research Center, San Jose, CA.

1 Introduction

In this paper we establish an $\Omega(\log n/\log\log n)$ lower bound for the competitive ratio of two natural problems in distributed computing on various networks of n processors. These problems are the mobile user problem introduced in [5, 6] and the distributed job scheduling problem introduced in [4]. For both problems there is a known polylogarithmic upper bound proved in the papers cited above, by applying the sparse partitions technique for any network with n processors. Our results thus imply that this is not far from being optimal, showing that sparse partitions yield nearly optimal bounds for these two problems. Here is a more precise formulation of the results.

1.1 Mobile users

In large communication networks, it may be desirable to allow users to move freely around the network and communicate with other users who are also free to move, an example being cellular telephone networks. A related situation is one where the mobile objects are files which may move through the network. Motivated by problems of this type, where the goal is to access an object which can change location in the network, Awerbuch and Peleg [5, 6] defined the problem of *on-line tracking of a mobile user*. We are given a data communication network, and one unique object called the *mobile user*. Initially, the mobile user is located at some node s, known to all the processors. The mobile user moves among the nodes (processors) of the network. From time to time requests are invoked at the nodes. There are two types of requests. The first is the move request, $Move(u, w)$, invoked at the current location u of the mobile user; the result of $Move(u, w)$ should be to move the user from node u to node w. The second is the find request, $Find(v)$, which can be invoked at any node v; the result should be to send a message from node v to the current location of the user. So that the "current location of the user" is well defined, we assume that requests occur in a sequential order.

We use a simple standard model for synchronous communication networks. The network is modeled as an undirected graph, where vertices represent processors and edges represent communication links. Each processor is modeled as a state machine (where the number of states can be infinite) which can receive and send messages. A processor with d incident edges has $d + 1$ input channels and d output channels. d of the input and output channels are used for communicating with its neighbors. The extra input channel is used for receiving input, such as *Move* and *Find* requests. The processors operate in lock-step synchrony. At each step, based on its current state and the messages (if any) on its input channels, a processor may change state and send any number of messages on output channels. In the case of randomized algorithms, the choice of new state and messages can be made probabilistically. A message sent from u to a neighbor v at step t is received by v at step $t + 1$. A more detailed description of the model is given in Section 2.

As in [5, 6], we measure the cost of a distributed algorithm for the mobile user problem by its *communication cost*, the total number of messages sent. Each sending of a single message over a single link adds one to this cost. We assume no bound on the size of a "single message" when proving our lower bounds. A tracking algorithm $\mathcal{A}$ receives a sequence σ of *Move* and *Find* requests, and must execute them on-line. Let $Cost_{\mathcal{A}}(\sigma)$ be the total communication cost incurred by $\mathcal{A}$ on the request sequence σ. If $\mathcal{A}$ is randomized, then $Cost_{\mathcal{A}}(\sigma)$ is the expected value. Here we assume that a request is not invoked until all the messages of the algorithm, sent while serving the previous request, reach their destinations. This assumption only strengthens our lower bound.

Again as in [5, 6], the efficiency of an on-line algorithm $\mathcal{A}$ is given by its *competitive ratio*, which measures the cost of $\mathcal{A}$ relative to the *optimal cost* of a "global" algorithm which knows the entire sequence σ in advance. For the mobile user problem, the optimal cost $Cost_{opt}(\sigma)$ is easy to calculate. It is just the sum of the optimal costs of all the requests in σ, where the optimal cost of $Move(u, w)$ is the distance from u to w, and the optimal cost of $Find(v)$ is the distance from v to the current location of the user.

The algorithm $\mathcal{A}$ has *competitive ratio* c if there is a constant a such that, for every request sequence σ,

$$Cost_{\mathcal{A}}(\sigma) \leq c \cdot Cost_{opt}(\sigma) + a. \tag{1}$$

In using this definition, we will be concerned with classes of networks of arbitrarily large size, and corresponding families of algorithms, one algorithm for each network in the class. In such cases, we let c (and sometimes a) be a function of n, the number of vertices in the network.

The competitive analysis of algorithms was introduced in [19] and has been considered in numerous recent papers. Three representative examples that deal with problems related to the ones we consider here are [5], [7] and [8]. While the problems we consider are on-line problems, it should be noted that the distributed nature of the problems make them somewhat different than the well-known k-server problem, and, more generally, problems which fit the framework of request-answer games [10]. In such problems, the difficulty arises because, when answering a particular request, the on-line algorithm has no knowledge of future requests, although it is usually assumed that the algorithm has perfect knowledge of all past requests. In the problems we consider, the algorithm might also have incomplete knowledge of past requests. For example, when $Find(v)$ is invoked, the processor at v will not in general know the past sequence of *Move* operations which brought the user to its current location.

It is easy to construct algorithms for the mobile user problem with competitive ratio $O(1)$ for certain classes of graphs, examples being trees and rings. For general graphs, one simple strategy, the *centralized* strategy, works by having a particular node, say s, be always informed of the current location of the mobile user. The competitive ratio of this algorithm is proportional to the diameter of the graph, which could be $\Omega(n)$. At another extreme, the *no information* strategy simply moves the user without informing other nodes of the new location, and the algorithm responds to $Find(v)$ by performing a breadth-first search from v. On certain graphs, the competitive ratio is again $\Omega(n)$. Awerbuch and Peleg [5] show that a much better competitive ratio is possible. They give a (deterministic) on-line distributed algorithm for the mobile user problem on any network with n processors, with a competitive ratio $O(\log^2 n)$. In the present paper we show that there are networks in which no algorithm can have a competitive ratio smaller than $\Omega(\log n / \log\log n)$. In proving this result, we show that a "bad" request sequence can be chosen by an *oblivious adversary* (cf. [10]) which must choose the entire sequence knowing only the algorithm. This adversary is weaker than adaptive adversaries which choose each request based on the response of the algorithm to previous requests. Of course, using a weaker adversary makes our lower bound results stronger.

Theorem 1 *There is a class of networks for which the competitive ratio of any (deterministic or* *randomized) on-line distributed algorithm for the mobile user problem is* $\Omega(\log n/\log\log n)$ *(against an oblivious adversary), where* n *is the number of processors in the network.*

In this result, the additive term a in (1) can be any function of n. We obtain the above lower bound for networks of three different types, described in Section 1.3. Among these networks are ones which have been considered extensively in the design of distributed networks, including the hypercube and highly expanding graphs.

1.2 Distributed job scheduling

In [4], Awerbuch, Kutten and Peleg define the *distributed scheduling* problem. We discuss here only a special case of the problem. We use the same synchronous network model as above. Initially, at step $t = 0$, some processors are given sets of jobs to execute. At each step, a processor can execute at most one job and can send any number of jobs and messages to neighboring processors. The algorithm continues until all the jobs are executed. More precisely, let $j_v(t)$ denote the number of jobs residing at processor v just before it executes step t. An "input" σ is an assignment of non-negative integers to $j_v(0)$ for all v. If processor v executes k_{ex} jobs at step t, sends a total of k_{out} jobs to neighbors at step t (where $k_{ex} \leq 1$ and $k_{ex} + k_{out} \leq j_v(t)$), and receives a total of k_{in} jobs which neighbors of v send to v at step t, then $j_v(t+1) = j_v(t) + k_{in} - k_{out} - k_{ex}$. The algorithm continues until the step t_0 such that $j_v(t_0) = 0$ for all v, and this t_0 is the *cost* of the algorithm. Thus, the cost measures the maximum delay of a job. The definition of competitive ratio is the same as in (1), where $Cost_{opt}(\sigma)$ is the maximum delay of a job in an optimal schedule produced by a "global" scheduling algorithm which knows σ completely. Here, we assume that the additive term a is a constant which is independent of n. (Indeed, it is not hard to see, for this special case of the problem, that no nontrivial lower bound for the competitive ratio holds if a can depend on n.)

In [4], a more general problem is considered, where jobs can be initiated at any time unit, not only at time 0. The cost of an algorithm is defined as the sum of all job delays. The authors give a polylogarithmic upper bound for their measure. They also prove that their distributed on-line algorithm is competitive under a much stronger measure of competitiveness, which immediately implies a polylogarithmic upper bound under the maximum delay measure. Moreover, for the above special case, where jobs are initiated only at time 0 and a job can be executed in one time unit, the algorithm described in Section 4 of [4] has competitive ratio $O(\log n)$. We have the following lower bound for the special case.

Theorem 2 *There is a class of networks for which the competitive ratio of any (deterministic or randomized) distributed algorithm for the distributed scheduling problem, under the maximum delay measure, is* $\Omega(\log n/\log\log n)$*, even in the case that all jobs are initiated at time 0.*

As is the case with Theorem 1 we prove the above theorem for networks of all three types described in the next subsection. In addition, for the more general problem we can improve the lower bound to $\Omega(\log n)$.

Most of our results still hold if the model is restricted so that at most a constant number of jobs can be transferred over each edge at each step. The only exception is that we do not have

Theorem 2 in this case for high-girth graphs of degree $d = o(\log n / \log\log n)$. Indeed, we cannot, since there is a distributed job scheduling algorithm for the special case with competitive ratio $O(d)$ in any graph of degree d, if at most a constant number of jobs can be sent over each edge at each step. In this algorithm, all jobs initiated at processor v are executed by v itself.

1.3 The networks

The first class of networks we consider is the one we call here the *high-girth networks*. These are networks of size n whose underlying graph is d-regular and its girth, i.e., the size of the shortest cycle in it, as well as its diameter, are both $\Theta(\log n / \log d)$. It is well known that such graphs exist (see, e.g., [11]). For any d in the range $3 \le d \le (\log n)^b$, where b is any constant, we can prove the lower bounds in Theorems 1 and 2 for any such network. We note that it has been observed by several researchers, including the first author, Awerbuch, Peleg and Schäffer, that networks of high girth supply easily lower bounds on the complexity of network synchronizers [3] and graph spanners [17].

The second class of networks considered is the class of hypercube networks. The *d-dimensional hypercube*, denoted $\mathbf{C}^d$, is the graph with vertices $\{0,1\}^d$ where two vertices are adjacent if the Hamming distance between them is 1, i.e., if they differ in precisely one coordinate. The number of vertices here is $n = 2^d$. Unlike the case of the high-girth networks, where the proofs are purely combinatorial, the proofs here (and especially the proof of Theorem 1) are more complicated and require some additional tools. We first apply a Harmonic Analysis technique, used previously in [14], to derive a certain fractional isoperimetric inequality for the cube, which may be of independent interest. In the traditional isoperimetric problem, we are given the size $|A|$ of a set A of vertices of $\mathbf{C}^d$ and an integer m. The goal is to place a lower bound on the size of any set X such that X contains all vertices that lie within distance m from some vertex in A. In this case, Harper [13] has shown that $|X|$ is minimized when A is as close as possible to a ball in $\mathbf{C}^d$. Given this result, a tight lower bound on $|X|$ can be calculated. We need a generalization of this problem where, for each $v \in A$, among all the vertices within distance m of v, we know only that some fraction γ of them belong to X (so the traditional problem is the case $\gamma = 1$). Although we do not have tight bounds on $|X|$ for the case $\gamma < 1$, we can prove a lower bound which is sufficient for our purposes. Roughly speaking, we show that $|X|/|A|$ must grow exponentially in m, provided that γ is not too small and $|A|$ is not too large. This lower bound is the key to proving Theorem 1. The proof that Theorem 2 holds for the cube is simpler. We also define a probabilistic generalization of the *regional matchings* defined in [5, 6], and give a lower bound on their complexity. This lower bound is used to prove a lower bound on the trade-off between the *move-stretch* and the *find-stretch* of any mobile user algorithm on the cube; these are, informally, the competitive ratios restricted to the *Move* and *Find* operations, respectively, in σ.

Although the proofs for the cube are more complicated technically than the ones for the high-girth networks, the effort seems worthwhile, as the hypercube is one of the more popular architectures considered by researchers working in the design of distributed networks.

The third class of networks we consider is the class of *highly expanding graphs*. Such graphs have also been suggested to be the underlying graphs of distributed networks, as their high connectivity properties may be helpful in the implementation of various communication protocols (see, for example, [18]). Intuitively, a graph is highly expanding if every sufficiently small set U of vertices expands by some sufficiently large factor when all neighbors of U are included. It is well known (see [1] and its references) that there is a tight correspondence between the expansion properties of a graph and the eigenvalues of its adjacency matrix. We thus prefer to define our class of graphs in terms of their eigenvalues. Let us call a graph an (n, d, λ)-graph if it is a d-regular graph on n vertices and the absolute value of every eigenvalue of its adjacency matrix, except the largest, is at most λ. It is known ([12],[15],[16]) that for any n and d with nd even there are (n, d, λ)-graphs with $\lambda = O(\sqrt{d})$. We can show that Theorems 1 and 2 hold for any such graph with $d = \Theta((\log n)^b)$, for any constant $b \geq 4$. The proof here combines the Linear Algebra techniques applied commonly in the study of expanders with the basic combinatorial idea used in the proof for the hypercubes.

The rest of this paper is organized as follows. The model we use for distributed algorithms is defined more precisely in Section 2. Section 3 deals with the mobile user problem, and Section 4 concerns the distributed scheduling problem. Sections 3 and 4 each contain more details about the definition of each problem and contain subsections for the three types of networks.

2 The model

Let $G = (V, E)$ be an undirected graph, where $V = \{u_1, \ldots, u_n\}$. We imagine that there is a processor p_i associated with vertex u_i, and we often identify a processor with its associated vertex. A particular (deterministic) distributed algorithm is specified by a set $\mathcal{M}$ of *messages*, a set $\mathcal{I}$ of *external inputs*, and for each processor p_i, a set Q_i of states, an *initial state* $q_{i,0}$, and a *transition function* δ_i. The sets $\mathcal{M}$, $\mathcal{I}$, and Q_i can be infinite. If p_i has d incident edges, say $\{e_1, \ldots, e_d\}$, then for each $q \in Q_i$, $x \in \mathcal{I} \cup \{\emptyset\}$, and $m_1, \ldots, m_d \in \mathcal{M} \cup \{\emptyset\}$, the transition function $\delta_i(q, x, m_1, \ldots, m_d) = (q', m'_1, \ldots, m'_d)$ for some $q' \in Q_i$ and some $m'_1, \ldots, m'_d \in \mathcal{M} \cup \{\emptyset\}$. An *input sequence* is given by $\sigma(t, i) \in \mathcal{I} \cup \{\emptyset\}$ for integer $t \geq 0$ and $1 \leq i \leq n$, meaning that p_i receives external input $\sigma(t, i)$ at step t. The symbol $\emptyset$ means that no input is received. The transition functions and an input sequence induce, in the natural way, a synchronous computation of the algorithm. Initially, at step 0, each p_i is in its initial state. Thereafter, at each step $t \geq 0$, if p_i is in state q, receives external input x, and receives message m_j over the edge e_j for $1 \leq j \leq d$, then p_i enters state q' and sends m'_j over the edge e_j. As above, $m_j = \emptyset$ (resp., $m'_j = \emptyset$) means that no message was received (resp., is sent). A message sent by p_i over the edge $\{p_i, p_j\}$ at step t is received by p_j at step $t + 1$.

In the case of a randomized algorithm, $\delta_i(q, x, m_1, \ldots, m_d)$ is a probability distribution on $Q_i \times (\mathcal{M} \cup \{\emptyset\})^d$. In this case, the transition functions and an input sequence induce a distribution on computations.

We will be concerned only with input sequences which are essentially finite, meaning that there is some T such that $\sigma(t, i) = \emptyset$ for all $t \geq T$ and all i. Given such an input sequence, the *communication complexity* of the algorithm is the (expected) total number of messages sent by all processors at all steps. If the action $(q', m'_1, \ldots, m'_d)$ is selected by p_i at step t, then the number of messages sent by p_i at step t is $|\{\, j \mid m'_j \neq \emptyset \,\}|$.

State q of p_i is *quiescent* if $\delta_i(q, \emptyset, \emptyset, \ldots, \emptyset) = (q, \emptyset, \ldots, \emptyset)$, with probability 1 in the case of a randomized algorithm. The system is quiescent if all p_i are in quiescent states. We assume that all initial states are quiescent.

Given a graph $G = (V, E)$ and $u, v \in V$, let $dist_G(u, v)$ denote the distance between u and v in G. Let $B_G(u, r) = \{ v \mid dist_G(u, v) \leq r \}$ denote the ball of radius r centered at u. When G is clear from context, the subscript G is omitted.

From now on we assume, whenever this is needed, that n is sufficiently large. Logarithms with no specified base are in base 2. To simplify the notation we also omit all floor and ceiling signs whenever these are not crucial. Also for simplicity, we have not tried to optimize the constants in our results.

3 Tracking a mobile user

First we say more about the correct behavior of an algorithm $\mathcal{A}$ for the mobile user problem on a graph $G = (V, E)$. For this problem, G has a distinguished vertex s, the initial location of the mobile user. All of our results hold for any choice of $s \in V$. The set of external inputs is $\mathcal{I} = \{ \mathit{Move}(u, w), \mathit{Find}(v) \mid u, w, v \in V \}$. Let $\sigma = \sigma_1, \ldots, \sigma_k$ be a sequence of requests from $\mathcal{I}$. Let l_i be the location of the user just before σ_i is executed. (That is, if there is a $j < i$ with $\sigma_j = \mathit{Move}(u, w)$, then l_i is the destination w of the last such *Move*; otherwise, $l_i = s$.) We consider only sequences σ where, if $\sigma_i = \mathit{Move}(u, w)$, then $u = l_i$, i.e., the move must start from the current location l_i. Initially, the system is quiescent. Suppose that $\mathcal{A}$ has executed the prefix $\sigma_1, \ldots, \sigma_{i-1}$ for some $i \geq 1$ and that the system is quiescent. If $\sigma_i = \mathit{Move}(u, w)$, we give input σ_i to u. We require that, with probability 1, the system eventually reaches quiescence. If $\sigma_i = \mathit{Find}(v)$, we give input $\mathit{Find}(v)$ to v. We require that, with probability 1, the system eventually reaches quiescence and that l_i receives a message at some step between the step when $\mathit{Find}(v)$ is given to v and the step at which quiescence is reached. We say that a request *completes* when the system reaches quiescence after the invocation of the request.

Let $\mathit{Cost}_{\mathcal{A}}(\sigma)$ be the communication complexity of $\mathcal{A}$ on the input σ. (Note that this is well-defined since no messages are sent when the system is quiescent.) $\mathit{Cost}_{opt}(\sigma)$ is the sum of the optimal costs of the requests in σ, where the optimal cost of $\sigma_i = \mathit{Move}(u, w)$ is $dist(u, w)$, and the optimal cost of $\sigma_i = \mathit{Find}(v)$ is $dist(v, l_i)$.

To give a very high-level outline of the method used to prove lower bounds on the complexity of the mobile user problem, focus for the moment on the special case of a deterministic algorithm $\mathcal{A}$ and a request sequence of the form $\mathit{Move}(s, w), \mathit{Find}(v)$, where we know a bound m such that $dist(v, w) \leq m$. Suppose we want to show that either $\mathcal{A}$'s *Move* cost is at least δ_W or $\mathcal{A}$'s *Find* cost is at least δ_R. When $\mathcal{A}$ executes $\mathit{Move}(s, w)$, it may write information at vertices in some set $W(w)$; we imagine that the algorithm "colors" these vertices. If $\mathcal{A}$'s communication cost when executing $\mathit{Move}(s, w)$ is at most $\delta_W - 1$, then $|W(w)| \leq \delta_W$. Define $R(v)$, the *read set* of v, to be the first $\delta_R - 1$ vertices which receive messages when $\mathcal{A}$ executes $\mathit{Find}(v)$, assuming that $\mathit{Move}(s, w)$ has *not* yet been executed. As before, $|R(v)| \leq \delta_R$; furthermore, $R(v)$ induces a connected subgraph. (Each set $W(w)$ also induces a connected subgraph, although this is not used in the proof.) If we can find vertices v and w with $dist(v, w) \leq m$ and $R(v) \cap W(w) = \emptyset$, then we can conclude that one of the two desired lower bounds holds since, on the request sequence $\mathit{Move}(s, w), \mathit{Find}(v)$, the execution of $\mathit{Find}(v)$ will not send a message to w unless some colored vertex receives a message.

The proof thus rests on intersection properties of families of sets $W(w)$ and $R(v)$ of bounded size, where each read set induces a connected subgraph. To prove a lower bound on competitive ratio, we must consider longer request sequences, consisting of a *Move* followed by several *Find*'s, so that the total optimal *Find* cost dominates the optimal *Move* cost $dist(s, w)$ (which could be as large as the diameter of the network). Moreover, to cancel the effect of a large additive term a in the definition of competitive ratio, we must repeat the construction for several phases.

In Section 3.1 we prove Theorem 1 for high-girth graphs of degree $d = \log^3 n$. The proof for this case is particularly clean and simple, so it serves as an introductory example for the more complicated proofs to follow. In Section 3.2 we outline the method used to prove Theorem 1 for the other networks. In Section 3.3 the method is applied to high-girth graphs of smaller degree. Section 3.4 considers the generalization of the vertex isoperimetric problem described in Section 1.3. In Section 3.5, a result from Section 3.4 is used to prove Theorem 1 for hypercubes. Theorem 1 is proved for highly expanding graphs in Section 3.6. The final two subsections contain related results on regional matchings (Section 3.7) and the trade-off between move-stretch and find-stretch for any mobile user algorithm on the cube (Section 3.8).

3.1 An illustrative example

.

In this section we prove Theorem 1 for high-girth networks of degree $d = \log^3 n$, girth g that satisfies $g \geq \log_d n = \Omega(\log n / \log\log n)$ as well as $g \leq 2\log_d n < \log n$, and diameter $\Theta(\log n / \log\log n)$. The fact that such graphs exist for every (sufficiently large) n is well known; see, e.g., [11].

We first prove the result for deterministic algorithms. Fix n and let the mobile user start at vertex s of a network as above. We prove the lower bound by constructing a request sequence in phases: the cost for the on-line algorithm in each phase is $\Omega(\log^2 n / \log\log n)$; the optimal cost for each phase is $O(\log n)$; and the number of phases can be as big as we wish. The large number of phases is needed to handle a possibly large value of the additive term a (which may even depend on n) in the definition (1) of competitive ratio. Each phase ends by having the mobile user return to s, to ensure that the work done during one phase does not help during future phases.

Let us say a vertex u is *colored* at a particular time t in a particular phase if some message sent during this phase, or the mobile user itself, reached u by time t. Consider a *Find* request invoked at some vertex v. After the request, messages are sent in the network. Let $R(v)$ (the *read set of* v) denote the set of $g/4$ vertices which these messages visit first, assuming that no vertex other than s has been colored yet in this phase. More precisely, starting with the system in the quiescent configuration at the beginning of the phase, invoke $Find(v)$ at v, and place v in $R(v)$. In the synchronous computation of $Find(v)$, let R_t be the set of vertices (processors) which receive a message at step t. For $t = 1, 2, 3, \ldots$, set $R(v) \leftarrow R(v) \cup R_t$ until the size of $R(v)$ reaches $g/4$, or until the system becomes quiescent. If setting $R(v) \leftarrow R(v) \cup R_t$ causes the size of $R(v)$ to exceed $g/4$ at some step t, then arbitrarily select enough vertices from R_t so that the size of $R(v)$ is exactly $g/4$. Note that if we invoke $Find(v)$ after some vertices have been colored by previous *Move* and *Find* operations in this phase, and if $R(v)$ does not contain a colored vertex, then $Find(v)$ sends at least $g/4$ messages. This is true because $Find(v)$ will not send a message to the current location of the mobile user unless some colored vertex receives a message. Referring to the high-level outline above, we take $m = 1$, $\delta_W = \log^2 n$, and $\delta_R = g/4$.

Lemma 3 *For every phase, there exists a vertex w in the network for which no more than $\log n$ of its neighbors v choose $w \in R(v)$.*

Proof. Fix a phase. There are at most $n \cdot g/4 < \frac{1}{4} n \log n$ pairs $\langle w, v\rangle$ such that $w \in R(v)$. If the lemma is not correct, then there would be at least $n \log n > \frac{1}{4} n \log n$ such pairs, a contradiction. $\square$

Fix some w which satisfies the conclusion of Lemma 3.

Lemma 4 *There exists a sequence of vertices $\{v_i\}_{i=1}^{\log n}$ which are neighbors of w, such that the following holds:*

> *Consider the request sequence $Move(s, w), Find(v_1), \ldots, Find(v_{\log n}), Move(w, s)$. Let W_0 be the set of vertices colored at the completion of $Move(s, w)$, and for $1 \le i \le \log n$ let W_i be the set of vertices colored at the completion of $Find(v_i)$. For $0 \le i < \log n$, if the number of vertices in W_i is smaller than $\log^2 n$, then $R(v_{i+1})$ contains no vertex in W_i.*

Proof. Let $N(w)$ denote the set of neighbors of w whose read set does not include w. By Lemma 3 and the assumption $d = \log^3 n$, $|N(w)| \ge \log^3 n - \log n$.

We construct the sequence v_j inductively. Assume we have already chosen the first i vertices in the sequence. We claim that if $i < \log n$, and the number of vertices in W_i is smaller than $\log^2 n$, then there exists a vertex $v \in N(w)$ such that $R(v) \cap W_i = \emptyset$. To prove this claim, suppose it is false. The number of vertices in $N(w) - W_i$ is bigger than the number of vertices in W_i. Thus there is a vertex in W_i which belongs to the read set of at least two vertices in $N(w) - W_i$. However, each read set is connected and has size at most $g/4$, and hence a cycle of size at most $2(g/4) + 2 < g$ exists in the network, a contradiction. $\square$

The last lemma clearly implies that the total cost of the on-line algorithm, given the request sequence described in the lemma, is at least $(\log n)(g/4) = \Omega(\log^2 n / \log\log n)$. Since the optimal cost is only $O(\log n)$ this completes the proof of the theorem for deterministic algorithms.

It is not difficult to modify this proof for randomized algorithms. By the easy direction of the main result of [20], in order to prove a $c(n)$ lower bound for the competitive ratio of any randomized algorithm for our problem it suffices to find a probability distribution on the inputs (i.e., the request sequences) so that for every deterministic algorithm the expectation (over the request sequences) of the ratio between the cost of the on-line algorithm and the optimal cost is at least $c(n)$. The probability distribution chosen on the request sequences is the following. Let $P = P(n)$ be a fixed large number, which will be the number of phases. In each phase independently, choose a random vertex w according to a uniform distribution, and then choose randomly and independently according to a uniform distribution $\log n$ neighbors $v_1, \ldots, v_{\log n}$ of w. The request sequence for the phase is: $Move(s, w), Find(v_1), \ldots, Find(v_{\log n}), Move(w, s)$. The proof of Lemma 3 implies that at least half of the vertices w satisfy the assertion of the lemma. The proof of Lemma 4 implies that if the total cost incurred by the on-line algorithm in a phase is less than $\log^2 n$ then the probability that a randomly chosen neighbor of w will not meet the colored part in its read set is at least, say, 1/3. Linearity of expectation now implies that the expected cost of the on-line algorithm during a phase is $\Omega((\log n)g) = \Omega(\log^2 n / \log\log n)$, whereas the optimal cost is $O(\log n)$. Thus the lower bound holds for randomized algorithms as well. $\Box$

We note that since the diameter of our network is $\Theta(\log n/\log\log n)$, the deterministic centralized strategy (where s is always informed of the current location of the user) has competitive ratio $O(\log n/\log\log n)$. Thus our bound is optimal for the specific network used.

3.2 Outline of the proof method

Let $\mathcal{G}$ be a class of graphs. The method is the following:

1. Choose appropriate functions $\delta_R(n), \delta_W(n), m(n)$, and choose a constant α, $0 < \alpha \leq 1$. On networks having n vertices, $\delta_R(n)$ is an upper bound on the size of read sets, $\delta_W(n)$ is an upper bound on the number of colored vertices, and $m(n)$ is an upper bound on the distance from the finder to the current location of the user. (We may write simply δ_R, δ_W, and m, when the number n of vertices is clear from context.)

2. Fix an arbitrary $G = (V, E)$ from the class $\mathcal{G}$, and let $n = |V|$. Consider a family of "read sets" $\{R(v) \mid v \in V\}$ where $v \in R(v)$ and $|R(v)| \leq \delta_R$ for all v, and where each $R(v)$ induces a connected subgraph of G. Recall that $B(w, m)$ is the set of vertices within distance m from w. Say that a vertex w is *costly* if, for any set $W \subseteq V$ with $|W| \leq \delta_W$, at least a fraction α of the vertices $v \in B(w, m)$ satisfy $R(v) \cap W = \emptyset$.

3. Show that, no matter how the read sets are chosen, at least a fraction α of the vertices of G are costly.

The set W corresponds to the set of "colored" vertices. Given a deterministic algorithm $\mathcal{A}$, the read sets are defined as in the previous section, using δ_R as an upper bound on the size of a read set. In the proof of the previous section, for example, we took $\delta_R = g/4$, $\delta_W = \log^2 n$, and $m = 1$.

Let $\mathcal{A}$ be a deterministic mobile user algorithm for G. Consider the probability distribution on request sequences obtained by choosing w uniformly at random from V, and choosing $v_1, \ldots, v_{\log n}$ uniformly at random from $B(w, m)$; then the sequence is $\sigma = \mathit{Move}(s, w), \mathit{Find}(v_1), \ldots, \mathit{Find}(v_{\log n}), \mathit{Move}(w, s)$. Let $D(n)$ be the maximum diameter of an n-vertex graph in $\mathcal{G}$. Then the expected cost of $\mathcal{A}$ (over the request sequences) and the optimal cost can be bounded as

$$\begin{aligned} \mathbf{E}(\mathit{Cost}_{\mathcal{A}}(\sigma)) &\geq \alpha \cdot \min\{\, \delta_W(n),\ \alpha \cdot (\log n) \cdot \delta_R(n) \,\}, \\ \mathit{Cost}_{opt}(\sigma) &\leq 2D(n) + (\log n) \cdot m(n). \end{aligned}$$

Some explanation of the first bound may be helpful. The reasoning is the same as in the previous section. First, w is costly with probability at least α. Suppose that the chosen w is costly. As before, let W_i be the set of vertices colored by $\mathit{Move}(s, w), \mathit{Find}(v_1), \ldots, \mathit{Find}(v_i)$. If $|W_i| \leq \delta_W$, then the expected cost of $\mathit{Find}(v_{i+1})$ is at least $\alpha\delta_R$, since with probability at least α we choose a v_{i+1} with $R(v_{i+1}) \cap W_i = \emptyset$. If $|W_i| > \delta_W$, then the cost of the algorithm is at least δ_W. If $\mathcal{A}$ is randomized, we again use [20] to conclude that there is a request sequence σ such that $\mathit{Cost}_{\mathcal{A}}(\sigma)$ is at least as large as the first bound above. As before, we repeat the construction for a large enough number of phases. In applications, we will have $\delta_W(n)$ large enough and $D(n)$ small enough that the lower bound on competitive ratio becomes simply $\Omega(\delta_R(n)/m(n))$.

3.3 High-girth graphs

In this section we prove Theorem 1 for d-regular graphs whose girth g and diameter are both $\Theta(\log n/\log d)$, for any degree $d = d(n)$ in the range $3 \le d(n) \le (\log n)^b$ for any constant b. We follow the outline described in the previous section. As in the example of Section 3.1, we take $\delta_R = g/4$ and $\delta_W = \log^2 n$. Letting $l(n) = \lceil \log_{d-1} \log n \rceil$, take $m(n) = 6l(n)$. We can take $\alpha = 1/8$. Note that this choice of the parameters gives a lower bound on the competitive ratio of $\Omega(\delta_R(n)/m(n))$ which is $\Omega(\log n/\log\log n)$.

Let $G = (V, E)$ be a high-girth graph as above. For any choice of d in the range above, $g = \Omega(\log n/\log\log n)$ and $l = O(\log\log n)$. Therefore, for any $w \in V$, the subgraph induced by $B(w, 6l)$ in G is a tree which we denote T_w. For $w \in V$, let $L(w, r) = \{v \mid \mathit{dist}(v, w) = r\}$ be the r-th "level" of T_w. Since $L(w, m)$ contains at least half of the vertices in $B(w, m)$, we concentrate on vertices in $L(w, m)$. (Alternatively, when choosing the find requests v_i in the proof outline, we could choose uniformly from $L(w, m)$.) Fix a family of read sets $R(v)$, each of size at most $g/4$.

Let $U(w)$ be the set of vertices $v \in L(w, 6l)$ whose read set $R(v)$ goes "up" into T_w at least distance $3l$, i.e.,

$$U(w) = \{\, v \in L(w, 6l) \mid R(v) \cap L(w, 3l) \neq \emptyset \,\}.$$

The proof is completed by proving the following two statements:

1. At least half of the $w \in V$ have $|U(w)| \le \log^5 n$.
2. If $|U(w)| \le \log^5 n$ then w is costly.

To prove the first statement, let

$$P = \{\, \langle x, v\rangle \mid x \in R(v) \,\},$$

and note that $|P| \le (g/4)n = O(n \log n)$. Let

$$U' = \{\, \langle w, x, v\rangle \mid v \in U(w) \text{ and } x \in R(v) \cap L(w, 3l) \,\}.$$

Assuming for contradiction that at least half of the $w \in V$ have $|U(w)| > \log^5 n$, we would have $|U'| > \frac{1}{2} n \log^5 n$. Let $k = \max_{x\in V} |L(x, 3l)|$, and note that $k = O(\log^3 n)$. For each pair $\langle x, v\rangle \in P$ there are at most k triples $\langle w, x, v\rangle \in U'$, since $\mathit{dist}(x, w) = 3l$ for any such triple. Therefore, $|U'| \le k|P| = O(n \log^4 n)$, a contradiction.

To prove the second statement, let W be any set of size at most $\log^2 n$, and suppose for contradiction that at least half of the vertices $v \in L(w, 6l) - U(w)$ have $R(v) \cap W \neq \emptyset$. Therefore, there is a particular $z \in W$ and a set $Y \subseteq L(w, 6l) - U(w)$ with

$$|Y| \ge \frac{\log^6 n - \log^5 n}{2 \log^2 n} = \Omega(\log^4 n),$$

such that $z \in R(v)$ for every $v \in Y$. Let $v, v' \in Y$ be such that a, the least common ancestor of v and v' in T_w, satisfies $dist(v, a) > 3l$ (this is possible because of the lower bound on the size of Y). Since $R(v)$ and $R(v')$ do not go up into T_w farther than distance $3l$, it follows that $R(v) \cap R(v')$ contains no vertex of T_w. In particular, z is not in T_w. Therefore, there is a cycle of length at most $2(g/4) + 12l < g$ in G passing through a and z. This contradiction shows that $R(v) \cap W = \emptyset$ for at least half of the vertices v in $L(w, 6l) - U(w)$, and therefore at least $1/8$ of the vertices v in $B(w, m)$. This shows that w is costly, and completes the proof of the theorem for high-girth graphs. $\square$

3.4 Fractional neighborhoods

The subject of this section is the generalization of the vertex isoperimetric problem on the hypercube mentioned in Section 1.3. We first define the problem for a general graph $G = (V, E)$. For $A, X \subseteq V$, $m \geq 1$, and $0 < \gamma \leq 1$, we say that X is a *γ-fractional m-neighborhood* of A if, for every $v \in A$,

$$|B(v, m) \cap X| \geq \gamma |B(v, m)|.$$

The goal is to place a lower bound on $|X|$ in terms of $|A|$, m, and γ. The reader who is not interested in the details of the proof of this lower bound can simply note the statement of Theorem 6 and skip directly to Section 3.5, since these details are not needed later.

To establish such a bound for hypercubes, we first introduce some tools by giving two definitions and stating a known inequality. Let f be a real-valued function defined on $\mathbf{C}^d$. The p-norm of f is

$$\|f\|_p = \left(\frac{1}{2^d} \sum_{v \in \mathbf{C}^d} |f(v)|^p \right)^{1/p}.$$

For $0 < \varepsilon < 1$, let $T_\varepsilon(f)$ be the function defined on $\mathbf{C}^d$ as

$$T_\varepsilon(f)(v) = \sum_{x \in \mathbf{C}^d} f(x) \left(\frac{1+\varepsilon}{2} \right)^{d - dist(x,v)} \left(\frac{1-\varepsilon}{2} \right)^{dist(x,v)}.$$

For $X \subseteq \mathbf{C}^d$, let 1_X denote the characteristic function of X (for $v \in \mathbf{C}^d$, $1_X(v) = 1$ if $v \in X$, or 0 otherwise).

The operator $f \to T_\varepsilon(f)$ was considered in [14]. The definition in [14] is given in terms of the Fourier transform of f, but it is easy to see that it coincides with our definition. Two important lemmas of Beckner [9] allow us to relate the 2-norm of $T_\varepsilon(f)$ to the $(1 + \varepsilon^2)$-norm of f. As shown in [14], Beckner's lemmas give the following inequality:

Lemma 5 *For any real-valued function f on $\mathbf{C}^d$ and any $0 < \varepsilon < 1$, if $p = 1 + \varepsilon^2$ then*

$$\|T_\varepsilon(f)\|_2 \leq \|f\|_p.$$

In [14] the operator $T_\varepsilon(f)$ is given in terms of the Fourier transform of f. We mention now briefly how Lemma 5 follows from Beckner's results (without using the Fourier transform language). Write T_ε^d to indicate the operator T_ε on real-valued functions on $\mathbf{C}^d$. Clearly $\mathbf{C}^d$ is the Cartesian product of d copies of $\mathbf{C}^1$. First we note that T_ε^d is the tensor product of d copies of T_ε^1. To show that $T_\varepsilon^d = T_\varepsilon^1 \otimes \cdots \otimes T_\varepsilon^1$, it is enough to notice that, for every pair of points $(a_1, a_2, \ldots, a_d)$ and $(b_1, b_2, \ldots, b_d)$ of $\mathbf{C}^d$,

$$T_\varepsilon^d(1_{\{(a_1,a_2,\ldots,a_d)\}})(b_1, b_2, \ldots, b_d) = \prod_{i=1}^{d} T_\varepsilon^1(1_{\{a_i\}})(b_i).$$

This is immediate from the definition of T_ε^d.

Let Z_1, Z_2 be two sets and let R_i be an operator on real-valued functions defined on Z_i. Let $R_1 \otimes R_2$ be the tensor product of R_1 and R_2 defined on the space of real-valued functions on $Z_1 \times Z_2$. Now Beckner's Lemma 2 [9] (whose proof is straightforward) asserts that if $\|R_i(f)\|_2 \le \|f\|_p$ for every i and every real function f on Z_i, then $\|(R_1 \otimes R_2)(f)\|_2 \le \|f\|_p$ for every real function f on $Z_1 \times Z_2$. Thus, it is enough to show Lemma 5 for $d = 1$. This is exactly the content of Lemma 1 of Beckner [9]. In this case we have a (rather subtle) inequality between three real numbers.

We can now state and prove a result on fractional neighborhoods.

Theorem 6 *There is a constant $c_1 > 0$ such that the following holds. Let $d \ge 1$, $0 < \gamma \le 1$, $1 \le m < d/2$, and let A and X be two sets of vertices of $\mathbf{C}^d$. If X is a γ-fractional m-neighborhood of A, then*

$$|X| \ge \frac{c_1 2^m \gamma^2}{m} |A|^{1-m/d}.$$

If, in addition, $|A| \le 2^{d/2}$, then

$$|X| \ge \frac{c_1 2^{m/2} \gamma^2}{m} |A|.$$

Proof. The second inequality is immediate from the first. To prove the first inequality, we apply Lemma 5 with $f = 1_X$. Let $v \in A$. The definition of $T_\varepsilon(1_X)(v)$ involves a sum over all $x \in \mathbf{C}^d$. By taking the sum only over those $x \in B(v, m) \cap X$, and using the assumption that the size of this set is at least $\gamma |B(v, m)|$,

$$T_\varepsilon(1_X)(v) \ge \gamma \binom{d}{m} \left(\frac{1+\varepsilon}{2}\right)^{d-m} \left(\frac{1-\varepsilon}{2}\right)^{m}.$$

Therefore,

$$\|T_\varepsilon(1_X)\|_2^2 \ge \frac{|A|}{2^d} \gamma^2 \binom{d}{m}^2 \left(\frac{1+\varepsilon}{2}\right)^{2(d-m)} \left(\frac{1-\varepsilon}{2}\right)^{2m}.$$

Letting $p = 1 + \varepsilon^2$,

$$\|1_X\|_p^2 = (|X|/2^d)^{2/p}.$$

Substituting these bounds into the inequality of Lemma 5 and choosing $\varepsilon = 1 - 2m/d$ gives

$$|X| \geq 2^d \left[\frac{|A|}{2^d} \gamma^2 \binom{d}{m}^2 \left(1 - \frac{m}{d}\right)^{2(d-m)} \left(\frac{m}{d}\right)^{2m} \right]^{p/2}.$$

Since the quantity in square brackets can be at most 1, we can replace the exponent $p/2 = (1+\varepsilon^2)/2$ by the larger quantity $1 - m/d$. Since $2^d(2^{-d})^{1-m/d} = 2^m$,

$$|X| \geq (2^m \gamma^2 |A|^{1-m/d}) \left(\binom{d}{m} \left(1 - \frac{m}{d}\right)^{d-m} \left(\frac{m}{d}\right)^m \right)^{2(1-m/d)}. \tag{2}$$

Since $z!$ is $(z/e)^z \sqrt{z}$ to within constant factors, we have for some constant $c > 0$,

$$\begin{aligned} \binom{d}{m} &\geq c \sqrt{\frac{d}{m(d-m)}} \frac{(d/e)^d}{(m/e)^m((d-m)/e)^{d-m}} \\ &\geq \frac{c}{\sqrt{m}} \left(\frac{d}{m}\right)^m \left(\frac{d}{d-m}\right)^{d-m}. \end{aligned}$$

Therefore, the second factor in (2) is at least $(c/\sqrt{m})^{2(1-m/d)}$. So this factor is at least c_1/m for some c_1. □

Remark. For hypercubes, Harper [13] has shown that, given $|A|$, the size of a 1-fractional m-neighborhood X is minimized when A is a Hamming ball, i.e., $B(u,r) \subseteq A \subseteq B(u, r+1)$ for some u and r. This is not always true for $\gamma < 1$, as the following simple example illustrates. Let $d = 3$, $|A| = 4$, $m = 1$ and $\gamma = 3/4$. If A is a 2-dimensional subcube, say $\{(0,*,*)\}$, then we can take $X = A$. If A is a Hamming ball, however, the smallest $(3/4)$-fractional 1-neighborhood has size 5.

3.5 Hypercubes

In this section we prove Theorem 1 for hypercubes, using Theorem 6. Let $\mathbf{C}^d = (V, E)$, and let $\{R(v) \mid v \in V\}$ be a family of read sets. In addition to the upper bound δ_R on the maximum size of a read set, it will be useful also to have an upper bound ρ_R on the maximum radius of a read set, where the *radius* of $R(v)$ is $\max\{dist(u,v) \mid u \in R(v)\}$. Of course, since each $R(v)$ is connected, the upper bound δ_R on size is also an upper bound on radius. The number of vertices is $n = 2^d$.

Given δ_W, m and α as in the proof outline, recall that w is *costly* if, for every set $W \subseteq V$ with $|W| \leq \delta_W$,

$$|\{\, v \in B(w,m) \mid R(v) \cap W = \emptyset \,\}| \geq \alpha |B(w,m)|.$$

The following is the key lemma.

Lemma 7 *There is a constant $c_2 > 0$ such that for any sufficiently large d, any m, δ_R, δ_W, ρ_R, and α, and any family of read sets $\{R(v) \mid v \in V\}$ of maximum size δ_R and maximum radius ρ_R, if $1 \leq m < d/2$, $\rho_R \leq d/10$, and $\delta_W(\delta_R)^2 < c_2(1-\alpha)^6\, 2^{m/2}/m$, then at least a fraction α of the vertices of $\mathbf{C}^d$ are costly.*

Proof. Suppose that at least a fraction $1-\alpha$ of the $w \in V$ are not costly. For each such w, there is a set $W(w)$ with $|W(w)| \leq \delta_W$ such that $R(v) \cap W(w) \neq \emptyset$ for at least a fraction $1-\alpha$ of the $v \in B(w, m)$. Let S be the size of a ball of radius m. So there are at least $(1-\alpha)^2 Sn$ pairs $\langle w, v\rangle$ such that $R(v) \cap W(w) \neq \emptyset$ and $dist(v, w) \leq m$. Let $\beta = (1-\alpha)^2/2$. A simple counting argument shows that, if we let V' be the set of $v \in V$ such that $R(v) \cap W(w) \neq \emptyset$ for at least a fraction β of the $w \in B(v, m)$, then $|V'| \geq \beta n$. (For otherwise, the number of pairs $\langle w, v\rangle$ as above would be at most $(\beta + (1-\beta)\beta)Sn < 2\beta Sn = (1-\alpha)^2 Sn$.)

With each $v \in V'$ we associate a "special" vertex $\chi(v)$ as follows. By definition of V', $R(v)$ must intersect $W(w)$ for at least a fraction β of the w's in $B(v, m)$. Since $|R(v)| \leq \delta_R$, there must be a particular $z \in R(v)$ such that $z \in W(w)$ for at least a fraction $\gamma = \beta/\delta_R$ of the w's in $B(v, m)$. Define $\chi(v) = z$, and note that $dist(v, \chi(v)) \leq \rho_R$.

In the following, we restrict attention to those z's such that $z = \chi(v)$ for some $v \in V'$. Let

$$\begin{aligned} A_z &= \{\, v \mid \chi(v) = z \,\} \\ X_z &= \{\, w \mid z \in W(w) \,\}. \end{aligned}$$

It follows from the definitions of $\chi(v)$, A_z, and X_z that, for every $v \in A_z$, the set X_z intersects $B(v, m)$ in at least a fraction γ of the points of $B(v, m)$. In other words, X_z is a γ-fractional m-neighborhood of A_z.

Since $A_z \subseteq B(z, \rho_R)$ for all z, and since $\rho_R \leq d/10$ by assumption, $|A_z| \leq |B(z, d/10)| \leq 2^{d/2}$ for d sufficiently large.

From Theorem 6 we have, for all z,

$$|X_z| \geq M|A_z| \quad \text{where} \quad M = \frac{c_1 2^{m/2}\gamma^2}{m} = \frac{c_1 2^{m/2}\beta^2}{m(\delta_R)^2}.$$

Since the sets A_z partition V', the number of pairs $\langle w, z\rangle$ such that $w \in X_z$ is at least

$$\sum_z |X_z| \geq \sum_z M|A_z| = M|V'| \geq M\beta n.$$

Therefore, there exists a w such that $w \in X_z$ for at least βM different z's, so $|W(w)| \geq \beta M$ and $\delta_W \geq \beta M$. Substituting the values of M and β into $\delta_W \geq \beta M$, this contradicts the assumed upper bound on $\delta_W(\delta_R)^2$, if we take $c_2 = c_1/8$. $\square$

It is now easy to establish Theorem 1 for hypercubes. Let $\delta_R(n) = \rho_R(n) = d/10 = (\log n)/10$, $\delta_W(n) = \log^2 n$, $m(n) = 10 \log\log n$, and $\alpha = 1/2$. So all the conditions of Lemma 7 are satisfied for all sufficiently large n. The diameter of an n-vertex hypercube is $D(n) = \log n$. As before, this gives a lower bound on competitive ratio of $\Omega(\delta_R(n)/m(n))$.

3.6 Highly expanding graphs

As indicated in the previous section, any graph for which a sufficiently good lower bound on the size of fractional neighborhoods can be given will satisfy the assertion of Theorem 1. The next lemma gives such a lower bound for (n, d, λ)-graphs, provided λ is much smaller than d.

Lemma 8 *Let $G = (V, E)$ be an (n, d, λ)-graph, suppose $\gamma d \geq 6$, and let A and X be sets of vertices, where $|A| \leq \frac{\gamma}{2(d+1)}n$. If X is a γ-fractional 1-neighborhood of A then*

$$|X| \geq \frac{\gamma^2 d^2}{9\lambda^2}|A|.$$

Proof. Clearly we may assume that X is minimal, i.e., any proper subset of it is not a γ-fractional 1-neighborhood of A. In this case, every member of X either is in A or is a neighbor of some vertex of A, and hence $|X| \leq |A|(d+1) \leq \frac{\gamma}{2}n$.

Put $|X| = xn$ and observe that by the last paragraph $x \leq \frac{\gamma}{2}$. For any vertex $v \in V$ let $N(v)$ denote the set of all neighbors of v. A simple Linear Algebra argument (see [2], page 122) implies that

$$\sum_{v \in V} (|N(v) \cap X| - xd)^2 \leq \lambda^2 x(1-x)n.$$

Therefore

$$\sum_{v \in A} (|N(v) \cap X| - xd)^2 \leq \lambda^2 x(1-x)n. \tag{3}$$

However, since X is a γ-fractional 1-neighborhood of A, any $v \in A$ has at least $\gamma d - 1$ neighbors in X. Since $x \leq \gamma/2$ and $\gamma d \geq 6$, this together with (3) implies that

$$\frac{|A|\gamma^2 d^2}{9} \leq \lambda^2 x(1-x)n \leq \lambda^2 |X|,$$

implying the desired result. $\square$

In the remainder of this section we assume $\lambda = O(\sqrt{d})$, so Lemma 8 gives

$$|X| = \Omega(\gamma^2 d|A|). \tag{4}$$

Using Lemma 8 in place of Theorem 6, a version of Lemma 7 holds for highly expanding graphs, where $m = 1$ and where the three inequalities in the hypothesis are replaced by

$$6 \leq \frac{\beta d}{\delta_R} \tag{5}$$

$$d^{\rho_R+1} \leq \frac{\beta n}{2\delta_R(d+1)} \tag{6}$$

$$\delta_W(\delta_R)^2 < c_3(1-\alpha)^6 d \tag{7}$$

where $\beta = (1-\alpha)^2/2$ as before, and where the constant $c_3 > 0$ depends on the constant c such that $\lambda \leq c\sqrt{d}$. Since the proof is very similar, we only describe the differences.

Recalling that $\gamma = \beta/\delta_R$, (5) gives $\gamma d \geq 6$. Since $A_z \subseteq B(z, \rho_R)$, we use (6) to conclude

$$|A_z| \leq d^{\rho_R+1} \leq \frac{\beta n}{2\delta_R(d+1)} = \frac{\gamma n}{2(d+1)}.$$

So we can apply (4) to conclude $|X_z| \geq M|A_z|$ where $M = \Omega(\gamma^2 d)$. Now $\delta_W \geq \beta M$ contradicts (7) for a suitable choice of c_3.

This version of the lemma is used to prove Theorem 1 for highly expanding graphs with $d = (\log n)^b$ for any constant $b \geq 4$. We first note that the expansion properties of such graphs imply that the diameter of any n-vertex highly expanding graph is $O(\log n / \log d)$. For a suitable constant c, we take $\delta_R(n) = \rho_R(n) = c \log n / \log\log n$ and $\delta_W(n) = \log^2 n$. Recall that $m = 1$. It is easy to check that (5), (6), and (7) hold if c is small enough.

3.7 Regional matchings

Let $G = (V, E)$ be a graph with n vertices. As defined by Awerbuch and Peleg [5, 6], an *m-regional matching* for G is a family of "read sets" $R(x)$ and "write sets" $W(x)$ for $x \in V$ such that, if $dist(v, w) \leq m$, then $R(v) \cap W(w) \neq \emptyset$. Although Awerbuch and Peleg used regional matchings as a tool for constructing algorithms for the mobile user problem, the development above shows that they are also a useful concept for proving lower bounds.

The complexity of a regional matching is given by the maximum sizes and radii of the read sets and write sets. (Recall that the *radius* of $R(v)$ is the maximum, over $u \in R(v)$, of $dist(u, v)$.) For any G, m, and k, Awerbuch and Peleg [5] construct an m-regional matching for G where all radii are at most $(2k+1)m$ and all sizes are at most $2kn^{1/k}$. (It is natural to express the maximum radius as a multiple of m, since the "optimal" maximum radius is obviously $m/2$.) In this section, we define a probabilistic generalization of regional matchings. For hypercubes, we prove a lower bound on the trade-off between expected radius and expected size which has a form similar to the upper bound of [5] for deterministic regional matchings in an arbitrary graph. Specifically, if the maximum expected radius is at most km then the maximum expected size is $n^{\Omega(1/k)}$.

Let

$$\mathcal{R} = \{ \mathbf{R}(x), \mathbf{W}(x) \mid x \in V \}$$

be a family of mutually independent random variables taking values in 2^V. $\mathcal{R}$ is a *p-probabilistic m-regional matching* for G if, for all $v, w \in V$ with $dist(v, w) \leq m$,

$$\Pr[\, \mathbf{R}(v) \cap \mathbf{W}(w) \neq \emptyset \,] \geq p.$$

Let the *expected read-size* $\overline{\delta}_R(\mathcal{R})$ denote the maximum, over all $x \in V$, of the expected value of $|\mathbf{R}(x)|$. Similarly, the *expected read-radius* $\overline{\rho}_R(\mathcal{R})$ is the maximum, over all x, of the expected radius of $\mathbf{R}(x)$. The *expected write-size* $\overline{\delta}_W(\mathcal{R})$ and the *expected write-radius* $\overline{\rho}_W(\mathcal{R})$ are defined similarly in terms of the $\mathbf{W}(w)$. (In these notations, $\mathcal{R}$ is omitted when it is clear from context. The papers [5, 6] define the radii normalized to m, which in our notation would be $\overline{\rho}_R/m$ and $\overline{\rho}_W/m$.)

The following gives a lower bound on the product of read-size and write-size for any probabilistic regional matching for the cube, provided that the read-radius is not too close to the diameter d of $\mathbf{C}^d$. Of course, the same result holds with the roles "read" and "write" interchanged. (If read-radius and write-radius can both be as large as the diameter of the graph, then there is a trivial solution by taking all $R(x)$ and $W(x)$ to be the same singleton set.)

Theorem 9 *There is a constant $c_4 > 0$ such that for all sufficiently large d, all m with $1 \leq m < d/2$, and any p-probabilistic m-regional matching for $\mathbf{C}^d$ with $\overline{\rho}_R \leq pd/40$,*

$$\overline{\delta}_W(\overline{\delta}_R)^3 \geq c_4 2^{m/2} p^6/m.$$

Proof. The proof is similar to that of Theorem 7. We first define the vertex $\chi(v)$ associated with v. Fix v. Define a random variable $\mathbf{W}$ as follows: First pick w uniformly at random from $B(v,m)$, and then pick a set W according to the distribution of $\mathbf{W}(w)$. By definition of a probabilistic regional matching,

$$\Pr[\,\mathbf{R}(v) \cap \mathbf{W} \neq \emptyset\,] \geq p.$$

With probability at most $p/4$, the size of $\mathbf{R}(v)$ exceeds the upper bound $\overline{\delta}_R$ on its expected value by more than the factor $4/p$, and similarly for the radius of $\mathbf{R}(v)$. Therefore, there must be a set $S \subseteq V$ with $|S| \leq 4\overline{\delta}_R/p$, $\max\{\mathit{dist}(u,v) \mid u \in S\} \leq 4\overline{\rho}_R/p \leq d/10$, and

$$\Pr[\,S \cap \mathbf{W} \neq \emptyset\,] \geq p/2.$$

(For otherwise, $\Pr[\,\mathbf{R}(v) \cap \mathbf{W} \neq \emptyset\,]$ would be at most $p/2 + (1-p/2)(p/2) < p$.)

Since $|S| \leq 4\overline{\delta}_R/p$, there is a $z \in S$ such that

$$\Pr[\,z \in \mathbf{W}\,] \geq p^2/(8\overline{\delta}_R). \tag{8}$$

Define $\chi(v) = z$. Note that $\mathit{dist}(v,z) \leq d/10$.

We next observe that (8) implies the following:

Claim. At least a fraction $p^2/(16\overline{\delta}_R)$ of the w's in $B(v,m)$ satisfy $\Pr[\,z \in \mathbf{W}(w)\,] \geq p^2/(16\overline{\delta}_R)$.

To verify this claim, let

$$L = \{\,w \in B(v,m) \mid \Pr[\,z \in \mathbf{W}(w)\,] \geq p^2/(16\overline{\delta}_R)\,\}.$$

From (8),

$$\begin{aligned}
p^2/(8\overline{\delta}_R) &\leq \Pr[\,z \in \mathbf{W}\,] \\
&= \Pr[\,z \in \mathbf{W}(w) \mid w \in L\,] \cdot \Pr[\,w \in L\,] + \Pr[\,z \in \mathbf{W}(w) \mid w \notin L\,] \cdot \Pr[\,w \notin L\,] \\
&\leq \Pr[\,w \in L\,] + p^2/(16\overline{\delta}_R).
\end{aligned}$$

So $\Pr[\,w \in L\,] \geq p^2/(16\overline{\delta}_R)$, which proves the claim.

Let $\gamma = p^2/(16\overline{\delta}_R)$. Letting

$$\begin{aligned}
A_z &= \{\,v \mid \chi(v) = z\,\}, \\
X_z &= \{\,w \mid \Pr[\,z \in \mathbf{W}(w)\,] \geq \gamma\,\},
\end{aligned}$$

it follows from the claim that X_z is a γ-fractional m-neighborhood of A_z.

Letting $M = c_1 2^{m/2}\gamma^2/m$, and proceeding as in the proof of Theorem 7, there is a vertex w such that $w \in X_z$ for at least M different z's. That is, $\Pr[\,z \in \mathbf{W}(w)\,] \geq \gamma$ for at least M different

z's. By another simple argument, the expected size of $\mathbf{W}(w)$ is at least $M\gamma$. So $\overline{\delta}_W \geq M\gamma$, which gives the desired inequality. $\square$

The following corollary gives a lower bound on the trade-off between read-radius (viewed as a multiple k of m) and size. We state the corollary in terms of a family $\{\mathcal{R}_d \mid d \geq 1\}$ where $\mathcal{R}_d$ is a probabilistic m-regional matching for $\mathbf{C}^d$. The various parameters are given as functions of $n = 2^d$, the number of vertices of $\mathbf{C}^d$.

Corollary 10 *There is a constant $a_1 > 0$ such that the following holds. Fix a $p > 0$ and let $1 \leq k(n) \leq a_1 p \log n / \log\log n$. There is a function $m(n)$ such that for any family of p-probabilistic $m(n)$-regional matchings for $\mathbf{C}^d$ with $\overline{\rho}_R(n) \leq k(n)m(n)$,*

$$\overline{\delta}_W(n)\,\overline{\delta}_R(n) = n^{\Omega(1/k(n))}.$$

Proof. Set $m(n) = pd/(40\,k(n))$ $(= p\log n/(40\,k(n)))$. By Theorem 9, for some constant c,

$$\overline{\delta}_W(n)\,\overline{\delta}_R(n) \geq n^{cp/k(n)}(c_4 p^6/m(n))^{1/3}.$$

Since $k(n) \leq a_1 p \log n/\log\log n$, $m(n) \leq \log n$, and p is fixed, the term $(c_4 p^6/m(n))^{1/3}$ can be absorbed into c, provided that a_1 is sufficiently small. $\square$

3.8 Move-stretch versus find-stretch

In [5, 6], the competitive ratio is broken into two parts, the *move-stretch* and the *find-stretch*, which are, informally, the competitive ratios restricted to the *Move* and *Find* operations, respectively, in the request sequence (and the additive term a is taken to be zero). In this section we give a lower bound on the trade-off between move-stretch and find-stretch for any mobile user algorithm on the cube.

Given a randomized mobile user algorithm $\mathcal{A}$ and a request sequence σ, let *Move-Cost*$_{\mathcal{A}}(\sigma)$ be the expected communication cost of $\mathcal{A}$ while executing the *Move* requests in σ. Let *Move-Cost*$_{opt}(\sigma)$ be the sum of the optimal costs of the *Move* requests in σ. The *expected move-stretch* of $\mathcal{A}$ is the supremum, over all σ, of *Move-Cost*$_{\mathcal{A}}(\sigma)$/*Move-Cost*$_{opt}(\sigma)$. The *Find-Cost*$_{\mathcal{A}}(\sigma)$, *Find-Cost*$_{opt}(\sigma)$, and the *expected find-stretch* of $\mathcal{A}$ are defined analogously for the *Find* requests.

For any G on n vertices, Awerbuch and Peleg [6] describe a deterministic tracking strategy which allows a trade-off between move-stretch and find-stretch; specifically, letting D be the diameter of G, move-stretch is $O(k \log D)$ and find-stretch is $O(k^2 n^{1/k})$. For example, when $k = \log n$, both are $O(\log^2 n)$.

We have the following lower bound on the trade-off for any randomized mobile user algorithm for hypercubes. Similar to Corollary 10, we state the result in terms of a family $\{\mathcal{A}_d \mid d \geq 1\}$ of algorithms for $\mathbf{C}^d$, with move-stretch μ and find-stretch φ given as functions of $n = 2^d$.

Theorem 11 *There is a constant $a_2 > 0$ such that the following holds. For any family of randomized algorithms for the mobile user problem on $\mathbf{C}^d$, having expected move-stretch $\mu(n)$ and expected find-stretch $\varphi(n) \leq a_2 \log n/\log\log n$,*

$$\mu(n) = n^{\Omega(1/\varphi(n))}.$$

Moreover, this holds even for request sequences of the form Move(s, w), Find(v).

Proof. Choose $a_2 \leq a_1/4$. We apply Corollary 10 with $p = 1/2$ and $k(n) = 2\varphi(n)$. This gives an $m(n) = O((\log n)/\varphi(n))$. Note that $k(n) \leq a_1 p \log n / \log\log n$.

For each d and $n = 2^d$, we obtain a p-probabilistic $m(n)$-regional matching $\mathcal{R}_d$ from the algorithm $\mathcal{A}_d$ as follows. For each $w \in V$, let $\mathbf{W}(w)$ be the set of vertices that receive a message when $\mathcal{A}_d$ executes *Move*(s, w); also include s and w in $\mathbf{W}(w)$. Let $\mathbf{R}(v)$ be the set of $2\varphi(n)m(n)$ vertices which receive messages first when $\mathcal{A}_d$ executes *Find*(v) in the configuration where all processors are in their (quiescent) initial states; also include v in $\mathbf{R}(v)$. A more precise definition of $\mathbf{R}(v)$ is like the definition of $R(v)$ in Section 3.1, the difference being that, since $\mathcal{A}_d$ is randomized, $\mathbf{R}(v)$ is a random set. To see that $\mathcal{R}_d$ is a $(1/2)$-probabilistic $m(n)$-regional matching, let $w, v \in V$ with $dist(w, v) \leq m(n)$. The event that $\mathbf{R}(v) \cap \mathbf{W}(w) = \emptyset$ implies the event that, on the request sequence *Move*(s, w), *Find*(v), the communication cost of $\mathcal{A}_d$ while executing *Find*(v) exceeds the upper bound $\varphi(n)m(n)$ on its expected value by at least the factor 2. This occurs with probability at most $1/2$.

Since $dist(s, w) \leq \log n$ for all w,

$$\begin{aligned}
\overline{\delta}_W(n) &\leq \mu(n)(\log n) + 2, \\
\overline{\delta}_R(n) &\leq 2\varphi(n)m(n) + 1, \\
\overline{\rho}_R(n) &\leq 2\varphi(n)m(n) = k(n)m(n).
\end{aligned}$$

(In fact, the last two bounds hold for the worst-case read-size and read-radius.)

So we can use Corollary 10 to conclude, for some constant $c > 0$, that

$$(\mu(n)(\log n) + 2)(2\varphi(n)m(n) + 1) \geq \overline{\delta}_W(n)\,\overline{\delta}_R(n) \geq n^{c/\varphi(n)}.$$

For sufficiently small a_2, the conclusion follows since $\varphi(n)m(n) = O(\log n)$, and $n^{c/\varphi(n)} \geq (\log n)^{c/a_2}$.
□

A consequence of Theorem 11 is $\max\{\varphi(n), \mu(n)\} = \Omega(\log n / \log\log n)$. This also follows from Theorem 1.

4 Distributed scheduling

In the distributed scheduling problem, processors are from time to time given sets of jobs to execute. At each step, a processor can execute at most one job and can send jobs to neighboring processors. The cost measure is the maximum delay of a job, i.e., the maximum, over all jobs j, of the number of steps that j remains in the system before being executed. We now make this more precise.

Let J be an (infinite) set of job names. The set $\mathcal{I}$ of inputs is the set of finite subsets of J. The input $\sigma(t, i) \subseteq J$ means that processor p_i is given the set of jobs $\sigma(t, i)$ at step t. We only consider finite input sequences, i.e., there is a time T such that $\sigma(t, i) = \emptyset$ for all $t \geq T$ and all i. We assume that each job is input at most once, i.e., if $(t, i) \neq (t', i')$ then $\sigma(t, i) \cap \sigma(t', i') = \emptyset$. Define $start(j) = t$ for all $j \in \sigma(t, i)$. Each state q of each processor p_i has a component $jobs(q)$

which represents the set of jobs residing at p_i when p_i is in state q. If q is an initial state, then $jobs(q) = \emptyset$. Each message m also has a component $jobs(m)$ which is the set of jobs transferred from p_i to p_j when p_i sends m to p_j. At each computation step, jobs must be conserved. More precisely, suppose that p_i at step t receives messages $m_1, \ldots, m_d$ from neighbors, sends messages $m'_1, \ldots, m'_d$ to neighbors, executes the job in set E (where $|E| \leq 1$), and changes from state q to state q'. Let $jobs_{in}$ (resp., $jobs_{out}$) be the union of $jobs(m_k)$ (resp., $jobs(m'_k)$) over $1 \leq k \leq d$. Then $jobs(q')$, E, and $jobs_{out}$ are pairwise disjoint, and

$$jobs(q') \cup E \cup jobs_{out} = jobs(q) \cup \sigma(t,i) \cup jobs_{in}.$$

If $E = \{j\}$ in this situation, define $finish(j) = t+1$. Given an input sequence σ and a (randomized) scheduling algorithm $\mathcal{A}$, let $Cost_{\mathcal{A}}(\sigma)$ be the (expected) maximum of $finish(j) - start(j)$ over all jobs j such that $j \in \sigma(t,i)$ for some t, i.

In the next three subsections we prove Theorem 2 for the three types of graphs. Recall that Theorem 2 concerns the special case of the problem where $\sigma(t,i) = \emptyset$ for all $t > 0$ and all i. We also obtain a slightly larger lower bound for the more general problem.

4.1 High-girth graphs

For any d with $3 \leq d \leq (\log n)^b$, let G be a d-regular high-girth graph whose girth g and diameter are both $\Theta(\log n / \log d)$.

Fix n. For every vertex v we define a scenario S_v as follows. Each vertex w in $B(v, g/4)$ receives g jobs at time 0. All other vertices do not get jobs. The subgraph of G induced by $B(v, g/2-1)$ is a tree which we denote T_v. A global algorithm can simply have each w that got jobs send its jobs, in the direction away from v in T_v, to a set of g vertices that are at distance $l = \log_{d-1} g$ from w. Since $l < g/4$ for sufficiently large n, all jobs are sent to vertices in T_v. Since T_v is a tree, each vertex receives at most one job to execute. So the optimal cost is at most $l+1$.

We prove the lower bound, again, by using the easy direction of [20]. The basic idea, which is typical of scenario arguments, is that the distributed algorithm does not know which scenario holds. In particular, if the cost of the algorithm is sufficiently smaller than $g/4$ (the radius used in defining scenarios), then a vertex x will behave the same in many different scenarios. This confusion leads to a contradiction. Pick at random one of the scenarios S_x according to a uniform distribution on the vertices. We show that for every (deterministic) distributed algorithm, the average cost over this distribution is at least $g/16$. Indeed, assume this is false and there exists a distributed algorithm whose average cost is less than $g/16$, and let us reach a contradiction. Let U be the set of all vertices u such that the cost of the algorithm is less than $g/8$ in the scenario S_u. Clearly, $|U| \geq n/2$ since otherwise the expectation of our algorithm exceeds $g/16$. For every $u \in U$ and every vertex x, we define $charge_u(x)$ to be the number of jobs that x executes among those assigned initially to the vertex u in the scenario S_u. Define also $Charge(x) = \sum_{u \in U} charge_u(x)$.

Consider a scenario S_x. Let y be a vertex in U such that $charge_y(x) > 0$. Obviously the distance between y and x is smaller than $g/8$. The vertex x must behave exactly the same in the scenarios S_x and S_y at the first $g/8$ time steps, since all the vertices in $B(x, g/8)$ receive the same number of jobs in both S_x and S_y. Similarly, y behaves the same in S_x and S_y. Thus x will execute $charge_y(x)$

jobs originated at the processor y under the scenario S_x. Moreover, it will execute them sometime during the first $g/8$ time steps. It follows that for every vertex x, $Charge(x) \leq g/8 < g/2$. Hence the total sum of $Charge(x)$ over all vertices x is strictly less than $n \cdot (g/2)$.

However, since $|U| \geq n/2$, the total number of jobs which are assigned to central vertices u of scenarios S_u with $u \in U$ is at least $(n/2) \cdot g$. Since each of these jobs contributes to $charge_u(x)$ for some u and x, the sum of $Charge(x)$ over all x must be at least $(n/2) \cdot g$, a contradiction.

The competitive ratio is $\Omega(g/\log_{d-1} g) = \Omega(\log n/\log\log n)$ since $g = \Theta(\log n/\log d)$. □

Note that if $d = \Omega(\log n/\log\log n)$, then $d = \Omega(g)$. In this case the optimal cost is $O(1)$ and it can be achieved by a global algorithm which sends at most one job over each edge at each step.

The special case of the distributed scheduling problem can always be solved by sending all jobs to a particular processor which then distributes the jobs evenly to all the processors. For a graph of diameter D, the competitive ratio of this algorithm is $O(D)$. Therefore, for high-girth graphs of degree $d = (\log n)^b$ where $b > 0$, our lower bound is optimal to within a constant factor.

For the general scheduling problem (i.e., not all jobs initiated at time $t = 0$), we have the following for high-girth graphs.

Theorem 12 *There is a class of networks for which the competitive ratio of any (deterministic or randomized) distributed algorithm for the distributed scheduling problem, under the maximum delay measure, is $\Omega(\log n)$, if jobs can be initiated at any time.*

Proof. Without any attempt to optimize the constants, consider, for a given n, a $d = 64$-regular graph $G = (V, E)$ with n vertices and girth $g = \Omega(\log n)$.

For every vertex v we define a scenario S_v as follows. Each vertex w in $B(v, g/4)$ receives a set of $d = 64$ jobs at time $t = 2i$ for all $i \in \{0, 1, \ldots, g/64 - 1\}$. Therefore, altogether each vertex in $B(v, g/4)$ receives g jobs, and the last set of jobs is initiated at time $t = g/32 - 2$. All other vertices receive no jobs. As before, let T_v denote the induced subgraph of G on the set of vertices $B(v, g/2 - 1)$, which is a tree. A global algorithm can complete the execution of each job at most two time units after its arrival time. This is done as follows. At time $t = 2i$ each vertex that receives jobs executes one of them and sends the other $d - 1$ to its neighbors in the direction away from v in T_v. At time $t = 2i + 1$ every vertex will have at most one job which can be completed in this time unit. Hence the optimal cost is at most 2, a constant.

We prove the lower bound as before, by using the easy direction of [20]. Pick at random one of the scenarios S_x according to a uniform distribution on the vertices. We show that for every (deterministic) distributed algorithm, the average *completion time*, i.e., time t in which all the jobs will be done, is at least $g/16$. Note that since all the jobs were initiated at times $t < g/32$, this will imply that the average value of the maximum delay of a job is at least $g/32 = \Omega(\log n)$ and hence complete the proof. The proof is almost identical to the previous one. Assume this is false and there exists a distributed algorithm whose average completion time is less than $g/16$, and let us reach a contradiction. Let U be the set of all vertices u such that the completion time of the algorithm is less than $g/8$ in the scenario S_u. Clearly, $|U| \geq n/2$. For every $u \in U$ and every vertex x, we define $charge_u(x)$ to be the number of jobs that x executes among those assigned initially to the vertex u in the scenario S_u. Define also $Charge(x) = \sum_{u \in U} charge_u(x)$.

Consider a scenario S_x. Let y be a vertex in U such that $charge_y(x) > 0$. Obviously the distance between y and x is smaller than $g/8$. The vertices x and y must behave exactly the same in S_x and S_y at the first $g/8$ time steps, since all vertices in $B(x, g/8)$ and in $B(y, g/8)$ receive jobs according to the same pattern in both S_x and S_y. Thus x will execute $charge_y(x)$ jobs originated at the processor y under the scenario S_x. Moreover, it will execute them sometime during the first $g/8$ time steps. It follows that for every vertex x, $Charge(x) \leq g/8 < g/2$. Hence the total sum of $Charge(x)$ over all vertices x is strictly less than $n \cdot (g/2)$.

However, since $|U| \geq n/2$, the total number of jobs which are assigned to central vertices u of scenarios S_u with $u \in U$ is at least $(n/2) \cdot g$. Since each of these jobs contributes to $charge_u(x)$ for some u and x, the sum of $Charge(x)$ over all x must be at least $(n/2) \cdot g$, a contradiction. $\square$

4.2 Hypercubes

The proof is similar to that given for high-girth graphs in Section 4.1. The scenario S_v gives $d = \log n$ jobs to each vertex within distance $d/10$ from v. We first show that the optimal scheduler can finish in time $O(\log d)$ by sending jobs "out", i.e., in the direction away from v. In fact, we show that this can be done even if we only allow a constant number of jobs to be transferred along a single edge in one time unit. By symmetry, we may assume that v is the all 0 vector, and hence the vertices in $B(v, d/10)$ are all the vertices whose Hamming weight is at most $d/10$. We need the following lemma.

Lemma 13 *Let U be a set of vertices of $\mathbf{C}^d$ where the Hamming weight of each member of U is at most $d/3 - 1$, and suppose each vertex in U has a set of at most b jobs. Then each such vertex of Hamming weight i can distribute its jobs among its neighbors with Hamming weight $i + 1$ in such a way that no vertex receives more than $\lceil b/2 \rceil$ jobs. Moreover, the maximum number of jobs sent along an edge is at most $\lceil 3b/2d \rceil$.*

Proof. For each $u \in U$, let $j(u)$ ($\leq b$) denote the number of jobs of u. Consider the following network-flow instance. The network consists of a source s, a sink t, and two layers of vertices, U and W, where W is the set of all vertices of the cube with Hamming weight at most $d/3$. For each $u \in U$, su is a directed edge of capacity $j(u)$. For each $w \in W$, wt is a directed edge of capacity $\lceil b/2 \rceil$. In addition, if $u \in U$ and $w \in W$ are neighbors in the cube, and the Hamming weight of w exceeds that of u by 1, then uw is a directed edge of capacity $\lceil 3b/2d \rceil$. It is easy to see that the value of the maximum flow in this network is precisely the sum $\sum_{u \in U} j(u)$. Indeed, the cut consisting of all su edges shows that the maximum flow cannot exceed this value. On the other hand, one can define a flow by having each $u \in U$ receive $j(u)$ units of flow from s and by having u distribute this flow equally among its out-neighbors $w \in W$. Since each $u \in U$ has at least $2d/3$ neighbors in W, this flow sends at most $3j(u)/2d \leq 3b/2d$ flow units on every uw edge. Moreover, since each $w \in W$ has at most $d/3$ neighbors in U and each neighbor sends it at most $3b/2d$ flow units, each w receives at most $b/2$ flow units, which it can now send to the sink t.

Since all the capacities in our network are integral, there is an integral flow whose value is $\sum_{u \in U} j(u)$. We can now distribute the jobs of each $u \in U$ according to this flow, completing the proof of the lemma. $\square$

By repeatedly applying Lemma 13 $\lceil \log d \rceil$ times, it follows that an optimal global scheduler can, in $\log d$ steps, distribute all the jobs initiated according to the scenario S_v among the vertices within distance $d/10 + \log d$ ($\leq d/3 - 1$) from v, so that each of these receives at most a single job, which it can now execute in one additional time unit. Therefore, an optimal scheduler can finish in time $O(\log d)$. It remains to show that any on-line algorithm needs time $\Omega(d)$. This can be proved by a simple modification of the argument in Section 4.1. Since the argument is very similar to the previous one, we omit the details.

4.3 Highly expanding graphs

In order to obtain Theorem 2 for highly expanding graphs it is useful to first show that every assignment of d jobs to each processor in *any* set of n/d processors in an (n, d, λ)-graph with $\lambda = O(\sqrt{d})$ can be completed (by a global algorithm) in constant time. This can be proved by combining Lemma 8 (with $\gamma = 1$) with Hall's Marriage Theorem. Indeed, by this lemma and Hall's Theorem one can easily show that there exists a constant $c > 0$ (depending on the constant c' for which $\lambda < c'\sqrt{d}$), so that for every set A of at most n/d vertices in an (n, d, λ)-graph $G = (V, E)$ there is a collection of pairwise disjoint subsets S_a, $a \in A$, of V so that each S_a is a set of neighbors of a of cardinality at least cd. By having the vertices in S_a perform the jobs initiated at a it is clear that a global algorithm can complete all jobs in time $O(1/c) = O(1)$.

The basic combinatorial idea in Section 4.1 can then be used to prove that Theorem 2 holds for such graphs with an appropriate choice of d. E.g., $d = (\log n)^b$, for any constant $b \geq 1$ will do. Simply consider, for each vertex v, the scenario S_v in which all vertices in $B(v, s)$ for, say, $s = \log_{d-1}(\sqrt{n})$, get d jobs each at time $t = 0$. Then the optimal cost is a constant, whereas it can be shown as before that the expected cost of any on-line algorithm is $\Omega(s) = \Omega(\log n / \log \log n)$. The details are almost identical to the ones above and are thus omitted.

Acknowledgement. We would like to thank Y. Afek, B. Awerbuch, Y. Bartal, D. Dolinsky, C. Dwork, S. Kutten and A. Rosén for helpful discussions and comments.

References

[1] N. Alon, Eigenvalues and expanders, *Combinatorica* **6** (1986) 83–96.

[2] N. Alon and J.H. Spencer, *The Probabilistic Method* (Wiley, New York, 1991).

[3] B. Awerbuch, Complexity of network synchronization, *J. ACM* **32** (1985) 804–823.

[4] B. Awerbuch, S. Kutten and D. Peleg, Competitive distributed job scheduling, in: *Proc. 24th ACM Symp. on Theory of Computing* (1992) 571–580.

[5] B. Awerbuch and D. Peleg, Sparse partitions, in: *Proc. 31st IEEE Symp. on Foundations of Computer Science* (1990) 503–513.

[6] B. Awerbuch and D. Peleg, Online tracking of mobile users, Technical Memo TM-410, MIT Lab. for Computer Science, Cambridge, MA, 1989.

[7] Y. Bartal, A. Fiat and Y. Rabani, Competitive algorithms for distributed data management, in: *Proc. 24th ACM Symp. on Theory of Computing* (1992) 39–50.

[8] Y. Bartal and A. Rosén, The distributed k-server problem – a competitive distributed translator for k-server algorithms, in: *Proc. 33rd IEEE Symp. on Foundations of Computer Science* (1992) 344–353.

[9] W. Beckner, Inequalities in Fourier analysis, *Annals of Math.* **102** (1975) 159–182.

[10] S. Ben-David, A. Borodin, R. Karp, G. Tardos and A. Wigderson, On the power of randomization in online algorithms, in: *Proc. 22nd ACM Symp. on Theory of Computing* (1990) 379–386.

[11] B. Bollobás, *Extremal Graph Theory* (Academic Press, New York, 1978), pp. 107–109.

[12] J. Friedman, J. Kahn and E. Szemerédi, On the second eigenvalue in random regular graphs, in: *Proc. 21st ACM Symp. on Theory of Computing* (1989) 587–598.

[13] L.H. Harper, Optimal numberings and isoperimetric problems on graphs, *J. Comb. Th.* **1** (1966) 385–393.

[14] J. Kahn, G. Kalai and N. Linial, The influence of variables on Boolean functions, in: *Proc. 29th IEEE Symp. on Foundations of Computer Science* (1988) 68–80. See also: Tech. Report No. 546, Institute for Mathematical Studies in the Social Sciences, Stanford Univ., Stanford, CA, 1989.

[15] A. Lubotzky, R. Phillips and P. Sarnak, Explicit expanders and the Ramanujan conjectures, in: *Proc. 18th ACM Symp. on Theory of Computing* (1986) 240–246. See also: A. Lubotzky, R. Phillips and P. Sarnak, Ramanujan graphs, *Combinatorica* **8** (1988) 261–277.

[16] G.A. Margulis, Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators, *Problemy Peredachi Informatsii* **24** (1988) 51–60 (in Russian). English translation in: *Problems of Information Transmission* **24** (1988) 39–46.

[17] D. Peleg and A.A. Schäffer, Graph spanners, *J. Graph Th.* **13** (1989) 99–116.

[18] D. Peleg and E. Upfal, The token distribution problem, *SIAM J. Comput.* **18** (1989) 229–243.

[19] D.D. Sleator and R.E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. ACM* **28** (1985) 202–208.

[20] A.C.-C. Yao, Probabilistic computation: toward a unified measure of complexity, in: *Proc. 18th IEEE Symp. on Foundations of Computer Science* (1977) 222–227.