Approximation Algorithms for Channel Allocation Problems in Broadcast Networks

Rajiv Gandhi[1], Samir Khuller[2], Aravind Srinivasan[3], and Nan Wang[4]

[1] Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965.
E-mail: gandhi@cs.umd.edu.
[2] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965 and an NSF CAREER Award CCR-9501355.
E-mail: samir@cs.umd.edu.
[3] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported in part by NSF Award CCR-0208005. E-mail: srin@cs.umd.edu.
[4] Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-0208005.
E-mail: nwang@cs.umd.edu.

Abstract. We study two packing problems that arise in the area of dissemination-based information systems; a second theme is the study of *distributed* approximation algorithms. The problems considered have the property that the space occupied by a collection of objects together could be significantly less than the sum of the sizes of the individual objects. In the *Channel Allocation Problem*, there are users who request subsets of items. There are a fixed number of channels that can carry an arbitrary amount of information. Each user must get all of the requested items from one channel, i.e., all the data items of each request must be broadcast on some channel. The load on any channel is the number of items that are broadcast on that channel; the objective is to minimize the maximum load on any channel. We present approximation algorithms for this problem and also show that the problem is MAX-SNP hard. The second problem is the *Edge Partitioning Problem* addressed by Goldschmidt, Hochbaum, Levin, and Olinick (*Networks, 41:13-23, 2003*). Each channel here can deliver information to at most k users, and we aim to minimize the total load on all channels. We present an $O(n^{1/3})$–approximation algorithm and also show that the algorithm can be made fully distributed with the same approximation guarantee; we also generalize to the case of hypergraphs.

1 Introduction

We develop approximation algorithms for certain packing problems arising in broadcast systems; these have the property that the objects to be packed "overlap". In other words, the space occupied by a collection of objects together

could be significantly less than the sum of the sizes of the individual objects. This is in contrast with traditional packing problems in which the objects to be packed are disjoint. A second theme of our work is that some of our algorithms can also be made completely distributed and implemented to run in polylogarithmic time, with only a constant-factor loss in the approximation guarantee. We study problems that arise in the area of dissemination-based information systems [1, 2, 11, 12, 23]. Such systems are used in application domains such as public-safety systems, election-result servers and stock tickers [3]. One characteristic of dissemination-based applications is that there is a high degree of overlap in the user needs. Since many user-requests in such applications are similar, it would be a waste of resources to transmit the information to each user separately. For users with similar requests, if their requests are grouped and transmitted only once then this wastage of bandwidth could be avoided. On the negative side, the grouped data may contain information that would be irrelevant for some users. Hence, the users would have to process the broadcast information to obtain the data that they want. Thus, there is a trade-off between reducing the bandwidth used by grouping the requests and the amount of processing of the broadcast data that the clients need to do to obtain the data that they requested. In our model, there is a transmitter such as a satellite that broadcasts information on a fixed number of physical multicast channels. Each user is assigned to some channel on which the user gets his/her requested data. Our work deals with satisfying the client requests in a timely manner, while minimizing the amount of bandwidth used.

Problems and Results. The first problem, Channel Allocation, can be defined as follows. There is a set of topics (e.g., news, sports events, stock-market updates), as well as a set of users. Each user requests a subset of items (topics). There are a fixed number of channels that can each carry an arbitrary amount of information. Each user must get all of the requested items from one channel, i.e., all the data items of each request must be broadcast on some channel. The load on any channel is the number of items that are broadcast on that channel, and the goal is to minimize the maximum load on any channel. Formally, we are given: (i) a set of topics $T = \{t_1, t_2, \ldots, t_n\}$, (ii) a collection of user-requests $R = \{R_1, R_2, \ldots, R_m\}$, where $R_i \subseteq T$ for all i, and $\max_i |R_i|$ is a constant w; and (iii) a positive integer k denoting the number of channels. Our goal is to construct a family $C = \{C_1, C_2, \ldots, C_k\}, C_i \subseteq T$, such that for each set $R_i \in R$, there exists a C_j such that $R_i \subseteq C_j$. For all j, C_j constitutes the set of topics on channel j. If $R_i \subseteq C_j$ then we say that request R_i is satisfied by channel j. The load on channel j is the number of topics placed on it: i.e., $|C_j|$. The objective function is to minimize the maximum load on any channel, i.e., to minimize $\max_j |C_j|$. We will denote this problem as CHA.

The second problem, Edge-Partitioning (EP), basically arises by bounding the number of requests that any channel can handle, in CHA. The setting is the same as in CHA, with the additional constraint that each R_i must be assigned to some channel C_j for which $R_i \subseteq C_j$ holds; furthermore, the number of requests (i.e., users) assigned to a channel should be at most k. Subject to

these constraints, the objective is to minimize $\sum_j |C_j|$. This problem was studied by Goldschmidt *et al.* [14] for the special case of $w = 2$, in the context of optical network design. (That is, given a graph G, we seek to cover the edges by subgraphs containing at most k edges each, and we aim to minimize the total number of vertices in the chosen subgraphs.) The work of [14] considers the case $w = 2$, and presents an $O(\sqrt{k})$–approximation algorithm.

We give an $O(n^{\frac{w-1}{w+1}}(\lg n)^{\frac{1}{w}})$–approximation algorithm for CHA; this is obtained by taking the better of a random construction and the output of a suitable set-cover problem. We also show that the problem is MAX-SNP hard for all $w \geq 4$; thus, a polynomial time approximation scheme for the problem would imply that $P = NP$. For the case $w = 2$, CHA is the following graph problem: cover all the edges of a given graph by a given number of subgraphs, minimizing the maximum number of vertices in these subgraphs. Here, we obtain an $O(n^{1/3-\epsilon})$–approximation algorithm for some positive constant ϵ. We also show that the problem is NP-hard for $w = 2$, even when there are only two channels.

For EP, we obtain an $O(w \cdot n^{\frac{w-1}{w+1}})$–approximation algorithm, by taking the better of a simple approach and a greedy algorithm. Recall that an $O(\sqrt{k})$–approximation algorithm was developed in [14] for the case $w = 2$; in this case, our bound of $O(n^{1/3})$ is incomparable with $O(\sqrt{k})$ (note that k can take on values from 1 up to m, the number of edges in the graph). We then present an alternative approach with the same approximation guarantee for the case $w = 2$, with the help of certain tail bounds for sums of correlated random variables [17, 18, 22]. We show that this can be implemented as a *polylogarithmic time, distributed* algorithm, where each arriving user only communicates with the servers handling the topics that the user is interested in. This brings us to the next main theme of this paper: that of *distributed* approximation algorithms. Given the emergence of various contexts where distributed agents (e.g., in the Internet) make decisions using only local information, it is natural to ask whether the notion of approximation algorithms can be brought to bear fruitfully in such contexts. Not many polylogarithmic-time distributed approximation algorithms are known: the few that we are aware of include [15, 19, 9]. We hope that the intriguing mix of approximation and the constraint of locality will be understood further by research in distributed approximation algorithms.

Related Work. A problem related to the ones we study is the well-known Dense k-Subgraph problem (DkS): given a graph G, select a subset of k vertices whose induced subgraph has the maximum number of edges. In the language of CHA, we have $w = 2$ and one channel with capacity k; we wish to satisfy the maximum number of user requests. This problem is NP-hard, and an $O(n^a)$-approximate solution for some $a < \frac{1}{3}$ was given by Feige *et al.* [10]. The problem is not even known to be MAX-SNP hard. Also, Daskin *et al.* [8] discuss the following related *printed circuit board (PCB) assembly problem.* In this problem we have a list of PCBs and a list of different component types required by each PCB. The machine that produces the PCBs can hold only a fixed number of different component types, and can be loaded any number of times. The goal here is to minimize the sum over all component types, of the number of times

each component type is loaded. The users correspond to the PCBs, the items correspond to the different component types required by a PCB and the channel corresponds to the machine. In other words, the channel capacity is fixed, any number of channels could be used and the objective is to minimize the sum of the channel loads. They show that the problem is NP-hard. For the general version of the problem in which each component type (item) and PCB (user) is associated with a cost, they provide a heuristic solution. They also provide a branch-and-bound algorithm that can optimally solve small to moderate sized instances of the problem.

Due to the lack of space, many proofs are deferred to the full version.

2 The Channel Allocation Problem

2.1 Algorithm

Our approach employs two different algorithms and chooses a solution of lower cost from the two solutions obtained. As we will see, these two algorithms perform "well" on different sets of inputs that cover the entire spectrum of inputs.

The first algorithm is the following simple randomized algorithm. *Independently* place each topic on each channel, $i, 1 \leq i \leq k$, with a probability p which will be determined later. We will show that with a sufficiently high probability we obtain a feasible solution whose cost is close to its expected cost. This probability can be boosted by repeating the random process.

The second algorithm uses the greedy set cover algorithm [6, 20, 21] on the set cover instance, I, that is constructed as follows. The elements of the instance, I, are the requests in R. Let t be some fixed large constant. For all $i, 1 \leq i \leq \binom{t}{w}$, consider all $\binom{m}{i}$ combinations of i elements. For each combination, Z, let S_z be the set of requests corresponding to the elements in Z and let T_z be the topics obtained by taking the union of the requests in S_z. The combination Z forms a set in I iff $|T_z| \leq t$. The size of our set cover instance, $|I| = \sum_{j=1}^{t} \binom{|T|}{j} \leq \sum_{j=1}^{t} |T|^j = O(|T|^t) = O(n^t) = O(m^t)$. Let $M \doteq \max_{S_z \in I}\{|S_z|\} = O(t^w)$ be the size of the largest set in I. Since t and w are constants, $|I|$ is polynomially bounded and M is a constant. Now we use the greedy set cover algorithm on I to obtain a set cover for R. For each set S_z chosen by the set cover algorithm we create a new channel. The topics in T_z constitute this channel and hence the requests in S_z are satisfied by this channel. The set cover covers all requests in R. This solution may be infeasible as it may use more than k channels. By using Lemma 1 we can convert it into a feasible solution using k channels.

We will now analyze our algorithm. Note that we can obtain solutions with good approximation guarantees trivially for the following values of w and k. If $w = 1$ we can get an optimal solution of cost $\lceil m/k \rceil$. If $k < 2 \ln m$, we get a $2 \ln m$ approximation guarantee, since for any k we can obtain a k-approximate solution by placing all topics on each of the k channels. If $k > (\frac{n}{\ln n})^w$, we can partition the requests into groups of size $\lceil (\ln n)^w \rceil$ and place each group on a separate channel. This is a feasible solution as there are at most n^w requests.

The cost of our solution is at most $O(w(\ln n)^w)$, thus giving an approximation guarantee of $O((\ln n)^w)$. For the rest of the analysis we will assume that $w \geq 2$ and $2 \ln m \leq k \leq (\frac{n}{\ln n})^w$.
Let (X, y) solution to CHA denote allocating y channels such that the load on each of the channels is at most X.

Lemma 1. *If (L, k'), where $k' > k$, is a solution to CHA then there exists a feasible solution $(\lceil \frac{k'}{k} \rceil L, k)$.*

Lemma 2. *With a sufficiently high probability, the randomized algorithm gives a $(O(n(\frac{\lg n}{k})^{\frac{1}{w}}), k)$ solution.*

Lemma 3. *The set cover approach gives a $(O((L_{OPT})^w), k)$ solution.*

Theorem 1. *There is a polynomial-time algorithm for CHA that gives an $O(n^{\frac{w-1}{w+1}}(\lg n)^{\frac{1}{w}})$-approximate solution.*

2.2 Hardness of Approximation

We will prove that CHA is MAX-SNP hard via a reduction from 3-Dimensional Matching problem (3DM) which can be formally stated as follows.
3-Dimensional Matching (3DM): Given three disjoint set of elements, X, Y, and Z, such that $|X| = |Y| = |Z| = q$, and a set of triples, C, each triple containing one element from X, Y, and Z. The goal is to find the maximum number of pairwise disjoint triples.
For clarity, we prove Theorem 2 for all $w \geq 10$. We can show that CHA is MAX-SNP hard for all $w \geq 4$ by replacing each request, D, of size 10 in the reduction by $\binom{10}{4}$ requests of size 4, where each new request is a subset of D.

Theorem 2. *Unless $P = NP$, for some fixed $\epsilon > 0$, the channel allocation problem is hard to approximate to within a factor of $(1 + \epsilon)$.*

Proof. Let 3DM(I) denote the cost of an optimal solution to instance I. Similar definitions apply to CHA. 3DM is NP-complete [13]. We will prove the following.

$$I \in \text{3DM} \Longrightarrow \text{CHA}(f(I)) \leq 12 \tag{1}$$

$$I \notin \text{3DM} \Longrightarrow \text{CHA}(f(I)) \geq 13 \tag{2}$$

The function f shows that an approximation algorithm for CHA yielding a solution of cost lower than $\frac{13}{12}OPT$ would imply that P=NP. Our reduction is inspired by the NP-hardness reduction from 3DM to Partition into Triangles [13].

Consider a 3DM instance I. For notational convenience we will drop the parameter I while using the symbols, for e.g., we will use C instead of $C(I)$ to denote the set of triples in I. We now describe the function f that converts I into a CHA instance, $f(I)$, as follows. The CHA instance that we construct will

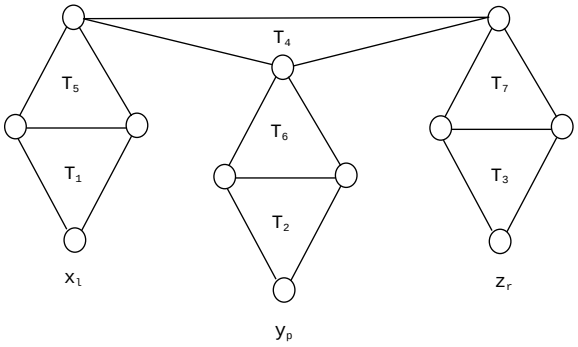

Fig. 1. Gadget corresponding to each triple.

have *big* requests, *small* requests and *dummy* requests. We start by describing the *big* requests. There are $(3q + 9|C|)$ big requests, one for each of the $3q$ elements in I and 9 for each triple in C. The big requests are mutually disjoint. Each big request, B, has 4 topics, $t_i^B, 1 \leq i \leq 4$. We will now describe the *small* requests. For each triple, $C_j \in C$, we will construct the gadget shown in Figure 1. Each gadget consists of 12 big requests (mentioned earlier), 9 of which are unique to the gadget and the other 3 big requests corresponding to the elements in C_j are shared between the gadgets corresponding to the triples containing the elements. Each edge connecting two big requests U and V represents 16 small requests, $\{t_i^U\} \cup \{t_j^V\}$, for all combinations of i, j for $1 \leq i, j \leq 4$. Thus each small request has size 2 and contains one topic from each of the two big requests. We also have $(144|C| - 48q)$ *dummy* requests of size 10 each. The dummy requests are mutually disjoint and disjoint from all other requests. This completes our description of requests. The set of topics is the union of the big requests and dummy requests. The total number of channels is $4q + 3(|C| - q) + 96q + 144(|C| - q) = q + 3|C| + 144|C| - 48q$.

Before we prove (1) and (2), let us define some notation. Consider a gadget representing a triple $C_j \in C$. Let $T_i^j, 1 \leq i \leq 7$, denote the set of 12 corresponding to the big requests that form the triangle T_i^j as seen in Figure 1. For notational convenience, we will drop the superscript j. Note that T_i denotes a set of topics as well as a triangle. The reference will be clear from the context in which it is used. A channel satisfying a triangle T_i would mean that the set of topics, T_i, is placed on the channel and hence the 3 big requests that form the vertices of the triangle and 48 small requests represented by the edges of the triangle are satisfied.

Claim. If $\text{3DM}(I) = q$ then $\text{CHA}(f(I)) \leq 12$.

Claim. If $\text{3DM}(I) < q$ then $\text{CHA}(f(I)) \geq 13$.

3 CHA Instances with Small Set-Size

In this section we consider the case of CHA instances when user requests are of size at most 2. In this case the user requests can be modeled as a graph in which the vertices represent the topics and the edges represent the user requests, i.e., an edge (i, j) would represent a user requesting topics i and j. The goal is to allocate channels while minimizing $\max_{1 \leq i \leq k} L_i$. We can show:

Theorem 3. *CHA is NP-hard when each request is of size two and there are two channels.*

We next give an approximation algorithm for CHA. Our algorithm uses the solution for the the Dense k-Subgraph problem (DkS) described in Section 1. Specifically, we use the approximation algorithm DkS(G, k) due to [10].

Algorithm: Guess the optimal load by trying out all possible values. Consider a guess L. Invoke DkS(G, L), which returns an approximate solution for the densest subgraph on L vertices. Place these L vertices returned by DkS(G, L) onto a new channel. Remove all the covered edges from G. If any edges remain uncovered invoke DkS again.

It is not hard to show that we get an $O(\rho \lg n)$-approximate solution here, where ρ is the approximation guarantee of DkS(G, k). Thus we have

Theorem 4. *For a certain constant $a < 1/3$, there is an $O(n^a \ln n)$-approximation algorithm for CHA.*

4 The Edge-Partitioning Problem

We now present approximation algorithms for EP: a sequential, deterministic algorithm in Section 4.1, and a distributed, randomized one in Section 4.2. We will throughout use hypergraph covering terminology: given a hypergraph $H = (V, E)$ with n vertices and m edges (each having a fixed number w of vertices), we wish to partition the edges into sets of at most k edges each, in order to minimize the sum of the total number of vertices in each set ("each set" here means "each block of the partition").

4.1 A deterministic algorithm

We now present a deterministic $O(w \cdot n^{\frac{w-1}{w+1}})$–approximation algorithm; see Theorem 5. Recall that the *degree* of a vertex in a hypergraph is the number of edges incident to it (i.e., contain it). Let $H = (V, E)$ be the given hypergraph. We start by considering the following greedy algorithm:

EDGE PARTITION$(H = (V, E), k)$
1 $F \leftarrow \emptyset$
2 **While** $|E| > k$ **do**
3 Remove the isolated vertices from V

$H' = (V', E') \leftarrow H = (V, E)$
$L \leftarrow \emptyset$
While $|E'| > k$ **do**
$u \leftarrow$ a lowest degree vertex in H'
$L \leftarrow \{$edges in E' that are incident to $u\}$
$V' \leftarrow V' \setminus \{u\}$
$E' \leftarrow E' \setminus L$
End
$R \leftarrow E' \bigcup L$
Arbitrarily remove some edges from R to make $|R| = k$
$F \leftarrow F \bigcup \{R\}$ (i.e., R is the set of edges assigned to a new channel)
$H \leftarrow H \backslash R$
End
$F \leftarrow F \bigcup \{E\}$

Lemma 4. *For each iteration of the outer* **while** *loop, the number of vertices in R is at most $w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n' + 1 \simeq w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n'$, where $n' = |V'|$, $m' = |E'|$ for the $H' = (V', E')$ being used in that iteration.*

Lemma 5. *The total number of vertices in the edge partition is at most $\frac{wn}{(1-1/w)}\left(\frac{m}{k}\right)^{1-1/w}$.*

Lemma 6. *The optimal solution has at least $\max\{n, \frac{w/e}{k^{1-1/w}}m\} \geq n^{1/w} \cdot \frac{(w/e)^{1-1/w}}{k^{(1-1/w)^2}} m^{1-1/w}$ vertices.*

Lemma 7. *From Lemmas 5 and 6, the approximation ratio of our algorithm is at most $\frac{w^2}{w-1}\left(\frac{en}{wk^{1/w}}\right)^{1-1/w}$.*

Note that in the case of graphs, i.e., $w = 2$, the approximation ratio of our algorithm is at most $4\sqrt{\frac{en}{2\sqrt{k}}}$. Also note that the constant factor of this ratio can be improved in the analysis for $w = 2$. The algorithm of [14] works for $w = 2$, and their approximation ratio for $w = 2$ is about $\sqrt{\frac{k}{2}}$.

Lemma 8. *By partitioning E into m parts such that each part consists of exactly one edge, we obtain a trivial algorithm whose approximation ratio is at most $ek^{1-1/w}$.*

Theorem 5. *By running the first algorithm and the trivial algorithm and taking the best solution, we obtain an algorithm with approximation ratio at most $2w \cdot n^{\frac{w-1}{w+1}}$. The running time of the composite algorithm is $O(\frac{m}{k}(m+n))$.*

4.2 A distributed algorithm for the graph case

We now present a randomized distributed $O(n^{1/3})$–approximation algorithm for the case where the given hypergraph H is a graph $G = (V, E)$. Recall that

in the present case where $w = 2$, each user basically requests two topics. We consider a fully distributed model where each broadcast channel has a server running it, and where each topic also has its own distribution server. A topic-distribution server can communicate with a channel server, if the former wants its topic broadcast on that channel. Each arriving user communicates only with the two topic-distribution servers of interest to it; thus, the model is distributed in the sense that the users need not have any knowledge about each other. By interpreting the topics as vertices and as the two topics of interest to a user as an edge, we thus equivalently get the following familiar distributed point-to-point model. Each vertex in the graph $G = (V, E)$ has a processor which can communicate with its neighbors, as well as with the servers handling the channels. Each processor knows the values of n (which is a static parameter – the number of topics) and k. We now wish to assign each edge to one channel (from among an arbitrary number of channels), such that each channel has at most k edges assigned to it. (The two processors at the end-point of an edge co-operatively decide which channel that edge gets assigned to.) The goal is to minimize the sum, over all channels i, of the total number of vertices that use i (a vertex v uses i iff some edge incident to v is assigned to i). Computation proceeds in *rounds*: in each round, every node communicates with its neighbors, and updates its internal state. The running time of an algorithm is the number of rounds, and hence locality is the main constraint in this model; we aim for polylogarithmic-time algorithms.

We further distinguish two models: *strong* and *weak*. In the weak model, if a channel has more than k edges that attempt to get assigned to it, the channel sends back a "No" message to the end-points of these edges, after which the end-points can retry. In the strong model, even such attempts are disallowed, and if we ever attempt to send more than k edges to a channel, the system enters a "Failed" state. Such a strongly constrained model is less realistic than the weak model – in practice, a channel can typically report that it is getting overloaded, without crashing. However, we also study the strong model and show that if all nodes know the value of m (which can be obtained if each incoming user "registers" with a central server which broadcasts the value of m to all servers), then we can develop an $O(n^{1/3})$–approximation algorithm even for the strong model. (There is a positive probability of entering the Failed state in our algorithm for the strong model – indeed, this seems inevitable – but this probability can be made as small as n^{-c} for any desired constant c.) In the weak model, the processors need not know the value of m.

The algorithm. We first assume the strong model, where the value of m is known to all nodes; we will finally show how to translate our results to the weak model. As in Section 4.1, there is the "trivial algorithm" (which places at most k edges arbitrarily on each channel) whose total objective function value is at most $2m$. The trivial algorithm can be easily implemented in the strong model with a contention-resolution type algorithm, where each edge chooses to be assigned to each channel independently with a suitable probability p. Briefly, if $k \geq \log^2 n$, we take, say, $4(m/k) \log n$ channels and $p = k/(2m)$; each edge

tries each channel with probability p, and goes to the first one on which its trial came up 1. If $k < \log^2 n$, we just take $p = n^{-c}$ and take $(4/p)\log n$ channels, for a suitable constant c. It is easy to show that with high probability, we get a feasible solution with the desired objective function value of $O(m)$. Like in Section 4.1, our focus is on showing how to construct a feasible solution with objective function value $O(n\sqrt{m/k})$; taking the better of this solution and that of the trivial algorithm, will yield an $O(n^{1/3})$–approximation. For the rest of this discussion, we assume $k \geq \log^4 n$, say; if k is smaller, the above trivial algorithm already results in a poylog(n) approximation.

The heart of our algorithm is the following: a preprocessing step followed by a random-selection step. Define $\bar{d} = \lceil \frac{2m}{n} \rceil$, and let $deg(v)$ be the current degree of v. The preprocessing step is as follows; it basically ensures that the maximum degree is not much more than the average degree. Each $v \in V$ makes $\lceil \frac{deg(v)}{\bar{d}} \rceil$ virtual copies of itself; it then distributes its $deg(v)$ incident edges to these copies, so that no copy gets more than $\bar{d}$ edges. Thus we get a new graph with m edges, and maximum degree $\bar{d}$. It is easy to see that the new number of vertices is at most $2n$. So, we have a graph with number of vertices in the range $[n, 2n]$, which has m edges and maximum degree at most $2m/n$. Now, the random-selection step is as follows. Choose am/k new channels, where a is a suitable constant. Each vertex then independently goes into each of these channels with probability $p = \sqrt{k/(2m)}$. (More precisely, the choices for all virtual copies of an original vertex v, are all made independently by v.) An edge is assigned to a channel iff both of its end-points choose to go into that channel; if an edge gets assigned to more than one channel, it chooses one arbitrarily.

The above preprocessing and random-selection constitute the main iteration of the algorithm. Note that the expected number of edges on any channel is $k/2$, and that for any edge, the probability that it was assigned to at least one of the channels is $1 - (1 - k/(2m))^{am/k} \approx 1 - e^{-a/2}$. The expected total load on the channels is $a(m/k) \cdot np = an\sqrt{m/(2k)}$. If everything happens according to expectation, we would have covered a constant fraction $b \sim 1 - e^{-a/2}$ of the edges, at a total cost of $\Theta(n\sqrt{m/k})$. We can then try to iterate this argument on the residual graph, leading to a toal cost of

$$\Theta(\sum_{i \geq 0} n\sqrt{m(1-b)^i/k}) = \Theta(n\sqrt{m/k}); \tag{3}$$

furthermore, the running time is basically the number of iterations, which would be $O(\log m) = O(\log n)$ with high probability.

The above idea on total cost can be carried through by using the Chernoff-Hoeffding bounds [5, 16]. However, bounding the number of edges assigned to a channel is harder, due to correlations; moreover, the correlation among the edges is in the "wrong" direction, as far as proving a concentration of measure is concerned. This is where our preprocessing step helps; intuitively, since it eliminates high-degree vertices, the correlation among the edges is lessened. First of all, to lower-bound the number of edges assigned to any channel, we use Janson's inequality for lower-tail bounds [17]. Fix a particular channel. Let $X_e =$

1 be the event that edge e is assigned to that channel, and $X_e = 0$ otherwise. Define $e \sim f$ iff the edges e and f are different, and have a common end-point. Then, $\Delta \doteq \sum_{(e,f):\ e \sim f} \Pr[X_e = X_f = 1]$ can be bounded by $O(n(\bar{d})^2 p^3) = O(k^{3/2})$; this is because there are $O(n(\bar{d})^2)$ pairs (e, f) such that $e \sim f$, and because $\Pr[X_e = X_f = 1] = p^3$ for any such pair. Thus, by Janson's inequality, the probability that at most $k/4$ edges get assigned to that channel is at most $e^{-\Omega(\mu/(2+\Delta/\mu))}$, which is $e^{-\Omega(\sqrt{k})}$. Since we have assumed that $k \geq \log^4 n$, this is negligibly small. Next, in order to follow the constraint of the strong model, we also need to show that at most k edges get assigned to the channel. Such upper-tail bounds are usually harder, but the recent tail bounds of [18, 22] can be shown to help; they help show that the probability of more than k edges getting assigned to a channel is once again at most $e^{-\Omega(\sqrt{k})}$. (The fact that our preprocessing significantly reduces the maximum degree, once again plays a key role.)

The above brief sketch shows that "everything relevant happens nearly according to expectation", with high probability. The nodes no longer know the exact value of m after one or more iterations, but choose an estimate slightly larger than expectation, and repeat. We can now use the argument following (3) to claim our performance bounds, and this concludes our brief discussion of the main ideas. Finally, for the weak model, we do not know the value of m, but guess it by repeated doubling. More precisely, we first run the above protocol for the strong model assuming $m = 2$; for each surviving edge, its end-points then run the above protocol for $m = 4$, and so on. When we finally hit the correct value of m, we will terminate with high probability. Since the cost function in (3) is proportional to $\sqrt{m}$, our final cost now is just a constant times that of (3) with high probability; the running time remains polylogarithmic.

Acknowledgments. We thank Michael Franklin, Guy Kortsarz, Vincenzo Liberatore, Christine Piatko, I-Jeng Wang and An Zhu for useful discussions. The third author thanks the Institute for Mathematics and its Applications at the University of Minnesota for its pleasant hospitality during his visit in April 2003; part of this author's writing took place during this visit. We also thank the APPROX 2003 referees for their useful comments.

References

1. S. Acharya, R. Alonso, M. Franklin and S. Zdonik. Broadcast Disks: Data management for asymmetric communication environments. Proc. ACM SIGMOD International Conference on Management of Data, San Jose, CA., 1995.
2. S. Acharya, M. Franklin and S. Zdonik. Balancing push and pull for data broadcast. Proc. ACM SIGMOD International Conference on Management of Data, Tuscon, AZ., 1997.
3. D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang and S. Zdonik. Research in Data Broadcast and Dissemination. *International Conference on Advanced Multimedia Content Processing (AMCP)*, Osaka, Japan, 1998.
4. R. Bhatia. Approximation Algorithms for Scheduling Problems. Ph.D. Thesis, University of Maryland at College Park, 1998.

5. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493-509, 1952.
6. V. Chvátal. A greedy heuristic for the set-covering problem. *Math. of Oper. Res.* Vol. 4, 3, 233-235, 1979.
7. A. Crespo, O. Buyukkokten and H. Garcia-Molina. Efficient Query Processing in a Multicast Environment. *Proceedings of the* 16^{th} *International Conference on Data Engineering (ICDE)*, San Diego, 2000.
8. M. S. Daskin, O. Maimon, A. Shtub and D. Braha. Grouping components in printed circuit board assembly with limited component staging capacity and single card setup: problem characteristics and solution procedures. *International Journal of Production Research*, 35, 1617-1638, 1997.
9. D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 717–724, 2003.
10. U. Feige, G. Kortsarz and D. Peleg. The Dense k-Subgraph Problem. *Algorithmica* 29, 410-421, 2001.
11. M. Franklin and S. Zdonik. A framework for scalable dissemination-based systems. *Proc. Object Oriented Programming Systems, Languages and Applications, OOPSLA*, 1997.
12. M. Franklin and S. Zdonik. "Data in your face": push technology in perspective. *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
14. O. Goldschmidt, D. Hochbaum, A. Levin and E. Olinick. The SONET Edge-Partition Problem. *Networks*, 41:13-23, 2003.
15. D. A. Grable and A. Panconesi. Nearly optimal distributed edge coloring in $O(\log \log n)$ rounds. *Random Structures & Algorithms*, 10:385–405, 1997.
16. W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13-30, 1963.
17. S. Janson. Poisson approximations for large deviations. *Random Structures & Algorithms*, 1:221–230, 1990.
18. S. Janson and A. Ruciński. The deletion method for upper tail estimates. Technical Report 2000:28, Department of Mathematics, Uppsala University, Sweden, 2000.
19. L. Jia, R. Rajaraman and T. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. *Proc. ACM Symposium on Principles of Distributed Computing*, pages 33–42, 2001.
20. D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
21. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.
22. V. H. Vu. Concentration of non-Lipschitz functions and applications. *Random Structures & Algorithms*, 20:262–316, 2002.
23. J. Wong. Broadcast Delivery. In *Proc. of the IEEE*, 76(12), 1988.