# Approximation Algorithms for Channel Allocation Problems in Broadcast Networks[*]

Rajiv Gandhi [†] Samir Khuller [‡] Aravind Srinivasan [§] Nan Wang [¶]

**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 requests which are subsets of topics. There are a fixed number of channels that can carry an arbitrary number of topics. All the topics of each request must be broadcast on some channel. The load on any channel is the number of topics 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 topics for at most $k$ requests, 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 the (non-distributed) *Edge Partitioning Problem* of graphs to the case of hypergraphs.

**Keywords:** channel allocation problem; edge partitioning problem; distributed approximation algorithm.

## 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

[*] A preliminary version of this work appeared in the *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 47–58, 2003.

[†]Department of Computer Science, Rutgers University, Camden, NJ 08102. Part of this work was done while the author was a student at the University of Maryland and was supported by NSF Award CCR-9820965. E-mail: rajivg@camden.rutgers.edu.

[‡]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.

[§]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This material is based upon work supported in part by the National Science Foundation under Grant No. 0208005 and under ITR Award CNS-0426683. E-mail: srin@cs.umd.edu.

[¶]Department of Computer Science, University of Maryland, College Park, MD 20742. This material is based upon work supported in part by the National Science Foundation under Grant No. 0208005. E-mail: nwang@cs.umd.edu.

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, 10, 11, 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 requests. Each request is a subset of topics. There are a fixed number of channels that can each carry an arbitrary amount of topics. All the topics of each request must be broadcast on some channel. The load on any channel is the number of topics 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 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 $w$; [1] 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 assigned to a channel should be at most $k$.[2] Subject to these constraints, the objective is to minimize $\sum_j |C_j|$. This problem was studied by Goldschmidt *et al.* [12] 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 [12] considers the case $w = 2$, and presents an $O(\sqrt{k})$–approximation algorithm.

We give an $O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}}\right)$–approximation algorithm for CHA; this is obtained by taking the better of a random construction and the output of a suitable set-cover problem. Using a simplification of our original proof due to one of the referees, we show that for any given $w \geq 2$, CHA cannot be approximated better than $(w+2)/(w+1)$, unless 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 *induced*

[1]In order to achieve the claimed approximation ratios, our analysis needs to assume that $\max_i |R_i| = w$ is a constant.

[2]Note that we use the same notation $k$ to denote two different parameters in CHA and EP.

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 $\epsilon$. We also show that the problem is NP-hard for $w = 2$, even when there are only two channels.

For EP, we obtain an $O\left(w \cdot n^{\frac{w-1}{w+1}}\right)$–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 [12] 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 [16, 17, 22]. We show that this can be implemented as a *polylogarithmic-time distributed* algorithm, where each user who makes the new request only communicates with the servers handling the topics that the user's request 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 [8, 13, 18, 20]. 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 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.* [9]. The problem is not even known to be MAX-SNP hard. Also, Daskin *et al.* [7] 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 requests correspond to the PCBs, the topics 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 (topic) and PCB (request) 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.

**Notation.** Throughout the paper, log and ln denote logarithms to the base 2 and base $e$, respectively. Also, we let $\Pr[\cdot]$ and $\mathbf{E}[\cdot]$ denote probability and expectation, respectively.

**Organization of the paper.** Sections 2 and 3 discuss CHA and its special case where all requests are of size at most two, respectively. We then present a sequential approximation algorithm for EP in Section 4, and a distributed algorithm for the graph case of EP in Section 5. Finally, we conclude in Section 6.

## 2 THE CHANNEL ALLOCATION PROBLEM

In this section, we describe our algorithm for CHA (Section 2.1) and analyze it in Section 2.2. In Section 2.3, we show that the problem is MAX-SNP hard.

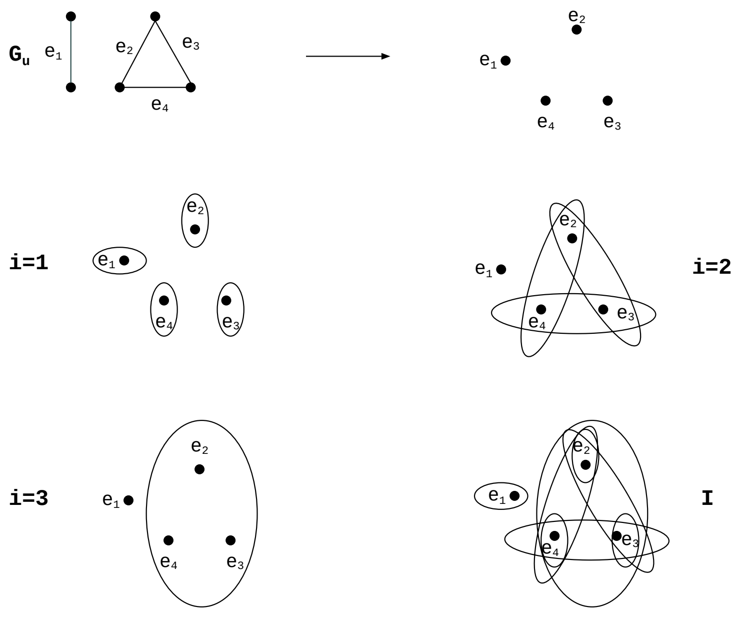

Figure 1: Construction of the set cover instance. Graph $G_u$ represents the requests. The topics are represented by vertices and the requests are represented by edges. $t = 3, w = 2$.

## 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, 19, 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. This construction is illustrated in Figure 1 for $t = 3, w = 2$. 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 2.1 we can convert it into a feasible solution using $k$ channels.

## 2.2 Analysis

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 $2\ln m \leq k \leq (\frac{n}{\ln n})^w$.
In the following discussion we refer to a solution to CHA that consists of $y$ channels and each with a load of at most $X$ as a $(X, y)$ solution.

**Lemma 2.1** *Let $k' > k$. If there exists an $(L, k')$ solution to CHA, then there exists a feasible $(\left\lceil \frac{k'}{k} \right\rceil L, k)$ solution.*

**Proof** Partition the channels into groups of size $\left\lceil \frac{k'}{k} \right\rceil$. Combine the load of each group and place it on to one channel. This gives us a $(\left\lceil \frac{k'}{k} \right\rceil L, k)$ solution. ■

**Lemma 2.2** *With a sufficiently high probability, the randomized algorithm gives an $\left(O\left(n\left(\frac{\log n}{k}\right)^{\frac{1}{w}}\right), k\right)$ solution.*

**Proof** Let $ON_{ij}$ be an indicator random variable that is 1 if topic $t_j$ is on channel $i$. Let $L_i = \sum_j ON_{ij}$ be the random variable that represents the load on channel $i$. Let $L$ be the random variable $\max_i L_i$. For any fixed $i, j$, $\mathbf{E}[ON_{ij}] = \Pr[ON_{ij} = 1] = p$. Thus, by linearity of expectation, $\mu \doteq \mathbf{E}[L_i] = np$. Since $L_i = \sum_j ON_{ij}$ is the sum of bounded and independent random variables, the Chernoff-Hoeffding bounds [5, 14] show that

$$\Pr[L_i \geq \mu(1+\delta)] \leq e^{-\frac{\mu\delta^2}{3}}, 0 \leq \delta \leq 1.$$

For any constant $c_1 \geq 1$, let $\delta = c_2\sqrt{\ln n/\mu} = c_2\sqrt{\ln n/np}$, where $c_2 = \sqrt{3c_1}$. We will ensure that $c_2\sqrt{\ln n/np} \leq 1$. We have

$$\Pr[L_i \geq \mu(1+\delta)] \leq \frac{1}{n^{c_1}}$$

for each $i$ and any fixed, desired $c_1 > 0$. Now, $\Pr[L \geq \mu(1+\delta)] = \Pr[\bigvee_i L_i \geq \mu(1+\delta)]$. By the union bound we have

$$\Pr\left[\bigvee_i L_i \geq \mu(1+\delta)\right] \leq \sum_i \Pr[L_i \geq \mu(1+\delta)]$$

which gives us $\Pr[L \geq \mu(1+\delta)] \leq k/n^{c_1} < n^{w-c_1}$ (for example we could pick $c_1 = w+1$). To summarize, we have proved that with high probability the maximum load on any channel is $O(np)$. Now we will calculate the probability that our randomized algorithm returns an infeasible solution. The infeasibility may arise as some requests may not be satisfied by any channel, i.e., no channel has all the topics in that request. We will calculate the probability of this bad event. Let $E_i$ be the event of request $R_i$ being satisfied and $\overline{E_i}$ the event of request $R_i$ not being satisfied. $\Pr\left[\overline{E_i}\right] \leq (1-p^w)^k$. Using the union bound, we get

$$\Pr\left[\bigvee_i \overline{E_i}\right] \leq \sum_i \Pr\left[\overline{E_i}\right] \leq m(1-p^w)^k \leq me^{-p^w k}.$$

We want this probability to be some small constant $c$, $0 < c < 1$. Solving $me^{-p^w k} \leq c$, for any desired, fixed $c$, gives us that $p$ of the form $\Theta((\ln n/k)^{\frac{1}{w}})$, (since $m \leq \binom{n}{w}$ and $w$ is a constant) is an appropriate choice. This choice of $p$ ensures that $c_2\sqrt{\ln n/np} \leq 1$ as required, since $2\ln m \leq k \leq (\frac{n}{\ln n})^w$. Let $E_o$ denote the event that the cost of our solution is greater than $\mu(1+\delta)$ and $E_u$ denote the event that some edges are not covered. The probability that any one of these bad events occur is given by $\Pr[E_o \bigvee E_u] \leq n^{w-c_1} + c$. This probability can be reduced by repeating the random process. This completes the proof. ■

**Lemma 2.3** *The set cover approach gives an $(O((L_{OPT})^w), k)$ solution.*

**Proof** The set cover obtained using the greedy algorithm has size at most $O(\log M)$OPT. But we do not know what OPT is for the set cover instance. However, we can find an upper bound on OPT as follows. Let $(L_{OPT}, k)$ be an optimal solution to the CHA instance. If we can convert an $(L_{OPT}, k)$ solution into a $(t, k')$ solution ($t$ is the constant used in the algorithm) then $k' \geq$ OPT. We can do this as follows. Consider a channel $h$ with at most $L_{OPT}$ elements. Partition the $L_{OPT}$ elements on $h$ into $\lceil wL_{OPT}/t \rceil$ groups containing at most $t/w$ elements each. Place each subset consisting of $w$ groups in a separate channel. Thereby every request that was satisfied by $h$ is still satisfied by the new placement. On the other hand, for each channel of load $L_{OPT}$ we create at most $\binom{\lceil wL_{OPT}/t \rceil}{w}$ channels of load $t$, giving us a $(t, O((L_{OPT})^w k))$ solution. Thus, OPT$\leq O((L_{OPT})^w k)$. Hence, the number of sets in our set cover is at most $O((\log M)(L_{OPT})^w k) = O((L_{OPT})^w k)$ which implies that we have a $(t, O((L_{OPT})^w k))$ solution to CHA. Using the method described in Lemma 2.1, we can convert this infeasible solution into an $(O((L_{OPT})^w), k)$ feasible solution. ■

**Theorem 2.4** *There is a polynomial-time algorithm for CHA that gives an $O(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}})$-approximate solution.*

**Proof** Our algorithm employs the two approaches and chooses a better solution. Hence, the cost of our solution is $\min\left\{O\left(n\left(\frac{\log n}{k}\right)^{\frac{1}{w}}\right), O((L_{OPT})^w)\right\}$. We calculate the approximation guarantee by considering the following cases.

<u>Case I</u>: $L_{OPT} \leq n^{\frac{1}{w+1}}$

The cost of our solution as upper bounded by the second term is at most $O(L_{OPT}^{w-1} L_{OPT}) = O(n^{\frac{w-1}{w+1}} L_{OPT})$. Hence the approximation ratio is $O\left(n^{\frac{w-1}{w+1}}\right)$.

<u>Case II</u>: $L_{OPT} > n^{\frac{1}{w+1}}$ and $k \geq n^{\frac{w}{w+1}}$

The cost of our solution as upper bounded by the first term is

$$\begin{aligned}
&\leq O\left(\frac{n}{(n^{\frac{w}{w+1}})^{\frac{1}{w}}}(\log n)^{\frac{1}{w}}\right)\\
&= O\left(n^{\frac{w}{w+1}}(\log n)^{\frac{1}{w}}\right)\\
&= O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}} n^{\frac{1}{w+1}}\right)\\
&\leq O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}}\right) L_{OPT}
\end{aligned}$$

This gives us an approximation guarantee of $O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}}\right)$.

<u>Case III</u>: $k < n^{\frac{w}{w+1}}$

Note that the average load of all channels is $\frac{n}{k}$, therefore $L_{OPT} \geq \frac{n}{k}$. The cost of our solution is upper bounded by the first term which is equal to $O\left(n(\frac{\log n}{k})^{\frac{1}{w}}\right) = O\left(\frac{n}{k}(\log n)^{\frac{1}{w}} k^{\frac{w-1}{w}}\right) \leq O\left((\log n)^{\frac{1}{w}} k^{\frac{w-1}{w}}\right) L_{OPT}$. This gives an approximation guarantee of $O\left(k^{\frac{w-1}{w}}(\log n)^{\frac{1}{w}}\right) \leq O\left((n^{\frac{w}{w+1}})^{\frac{w-1}{w}}(\log n)^{\frac{1}{w}}\right) = O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}}\right)$. ■

### 2.3 Hardness of Approximation

We now prove that CHA is MAX-SNP hard; we acknowledge the referee, whose proof that we present here is a substantial simplification of our original proof.

Consider the "Edge Partition into Triangles" problem: given an undirected graph $G = (V, E)$, we wish to determine if there exists a partition of $E$ into subsets, each of which is isomorphic to the triangle $K_3$. This problem is NP-complete [15]. For any given constant $w \geq 2$, we can transform this into an instance of CHA as follows. The set of topics is $T$ is the disjoint union of $V$ and a set $V'$ of $w-2$ "dummy" topics; so if $|V| = n$, then we have $n + w - 2$ topics in total. For each edge $\{u, v\} \in E$, we create a request $\{u, v\} \cup V'$. Thus, we have $|E|$ requests, each of cardinality $w$, and each containing $V'$. The number of channels $k$ equals $|E|/3$. Now note that any optimal solution for this problem will have $V'$ contained in every channel. Next, it is easy to see that $G$ can be edge partitioned into triangles if and only if there is a solution to this CHA instance with maximum load $3 + w - 2 = w + 1$. If $G$ cannot be so partitioned, then any solution to this CHA instance has to have maximum load at least $w + 2$. Thus we get:

**Theorem 2.5** *Consider the family of CHA instances for any given constant value (that is at least* 2*) for the parameter* $w$*. This family is NP-hard. Furthermore, this family cannot be approximated to within a factor smaller than* $(w+2)/(w+1)$ *unless* $P = NP$*.*

## 3 CHA INSTANCES WITH SMALL SET-SIZE

In this section we consider the case of CHA instances when requests are of size at most 2. In this case the requests can be modeled as a graph in which the vertices represent the topics and the edges represent the requests, i.e., an edge $(i, j)$ would represent a request's topics $i$ and $j$. The goal is to allocate channels while minimizing $\max_{1 \leq i \leq k} L_i$. In Section 3.1 we show that the problem is NP-complete even when there are only two channels; recall that the hardness proof of Section 2.3 requires an unbounded number of channels. In Section 3.2 we use an algorithm for the Dense $k$ Subgraph problem to obtain an approximation algorithm for CHA.

### 3.1 NP-Hardness

**Theorem 3.1** *CHA is NP-hard when each request is of size two and there are two channels.*

**Proof** The decision version of CHA in which each request is of size two can be posed as follows.

**CHA**: Given a graph $G_u$ that represents requests, and positive integers $k$ and $L$. Is there a channel allocation in which $\max_{1 \leq i \leq k} L_i \leq L$?

Our NP-hardness proof builds on the intractability of the Graph Bisection Problem (GBP) which is defined as follows.

**GBP**: Given a graph $G_b = (V_b, E_b), |V_b| = n, n$ is even, and a positive integer $W$. Is there a partition of $V_b$ into $V_b^1$ and $V_b^2$ such that $|V_b^1| = |V_b^2| = \frac{n}{2}$ and $|\text{cut}(V_b^1, V_b^2)| \le W$, where $\text{cut}(V_b^1, V_b^2)$ is the set of edges with one endpoint in $V_b^1$ and other in $V_b^2$.

We will first transform the graph $G_b$ into a graph $G_u$ that will represent the requests. For each vertex $v_i \in V_b$, we create a clique $C_i$ of size $n^2$. Label the vertices in each clique as $1, 2, \ldots, n^2$. Connect vertex $j$ in $C_i$ to a vertex $i$ in $C_j$ iff $(v_i, v_j) \in E_b$. Note that any node in a clique is connected to at most one node that is not in the clique. Setting $k = 2$ and $L = \frac{n^3}{2} + \left\lceil \frac{W}{2} \right\rceil$ completes the reduction from a GBP instance to a CHA instance.

**Claim** If $G_b$ has a bisection $(V_b^1, V_b^2)$ such that $|\text{cut}(V_b^1, V_b^2)| \le W$ then there is a channel allocation of cost $L = \frac{n^3}{2} + \left\lceil \frac{W}{2} \right\rceil$.

To prove this claim, let the cliques corresponding to vertices in $V_b^1$ constitute channel $CH_1$ and the cliques corresponding to vertices in $V_b^2$ constitute channel $CH_2$. Let $E_c$ be the set of edges that have one endpoint in $CH_1$ and other endpoint in $CH_2$. Note that no two edges in $E_c$ share a common vertex. Let $E_c = E_c^1 \cup E_c^2$ where $E_c^1$ is an arbitrary set of $\left\lceil \frac{W}{2} \right\rceil$ edges in $E_c$ and $E_c^2 = E_c \setminus E_c^1$. For each edge $(u, v) \in E_c^1$ such that $u \in CH_1$ and $v \in CH_2$, we move $v$ to $CH_1$. Similarly, for each edge in $(u, v) \in E_c^2$ such that $u \in CH_1$ and $v \in CH_2$, we move $u$ to $CH_2$. Thus $L_2 \le L_1 = \frac{n^3}{2} + \left\lceil \frac{W}{2} \right\rceil = L$.

**Claim** If there is a channel allocation of cost $L = \frac{n^3}{2} + \left\lceil \frac{W}{2} \right\rceil$ then $G_b$ has a bisection $(V_b^1, V_b^2)$ such that $|\text{cut}(V_b^1, V_b^2)| \le W$.

This claim is proved by observing that each clique must be contained completely on one channel. Also, both the channels must have the same number of cliques, otherwise the load on one of the channels would be at least $n^3/2 + n^2$. Since $W < n^2$, by keeping the number of cliques the same on both the channels, we can get a reduced load of at most $n^3/2 + W < n^3/2 + n^2$. Note also that no clique can appear completely on both the channels as only vertices $1, 2, \ldots, n$ of any clique have neighbors outside the clique and hence at most $n$ vertices of any clique appear on both the channels. Now, given a solution to CHA we can get a solution to the bisection problem by placing vertices in $V_b^1$ that correspond to cliques of size $n^2$ in $CH_1$ and placing vertices in $V_b^2$ that correspond to cliques of size $n^2$ in $CH_2$. On one channel there are $\left\lceil \frac{W}{2} \right\rceil$ vertices that do not belong to any of the $n^2$-sized cliques on that channel. Since the edges between the $n^2$-sized cliques on $CH_1$ and the $n^2$-sized cliques on $CH_2$ form a matching, on the other channel there are $\left\lfloor \frac{W}{2} \right\rfloor$ vertices. Each such vertex corresponds to an edge in $\text{cut}(V_b^1, V_b^2)$. Hence, $|\text{cut}(V_b^1, V_b^2)| \le W$. ■

## 3.2 Algorithm

We next give an approximation algorithm for CHA. Our algorithm uses the solution for the Dense $k$-Subgraph problem (DkS) described in Section 1. Specifically, we use the approximation algorithm $\text{DkS}(G, k)$ due to [9].

**Algorithm:** Guess the optimal load by trying out all possible values. Consider a guess $L$. Invoke $\text{DkS}(G, L)$, which returns an approximate solution for the densest subgraph on $L$ vertices. Place these $L$ vertices returned by $\text{DkS}(G, L)$ onto a new channel. Remove all the covered edges from $G$. If any edges remain uncovered invoke DkS again. The pseudo-code for the algorithm is given below.

```
Channel-Allocation($G(V,E), k$)
1  $n \leftarrow |V|$
2  for $L = 2$ to $n$ do
3      $i \leftarrow 0$
4      $G_L(V_L, E_L) \leftarrow G(V,E)$
5      while($|E_L| > 0$) do
6          $i \leftarrow i + 1$
7          $CH_i \leftarrow$ DkS($G, L$)
8          $E_L \leftarrow E_L \setminus C_i$          // Note: $C_i$ are edges covered on $CH_i$
9      $S_L \leftarrow \left\lceil \frac{i}{k} \right\rceil L$
10 return $\min\{S_1, S_2, \ldots, S_n\}$
```

Note that the algorithm could be made more efficient, for example, we may not need to test all values of $L$. However, for clarity of exposition, we will ignore the issue of efficiency.

**Lemma 3.2** *The algorithm* Channel-Allocation($G, k$) *gives an $O(\rho \ln n)$-approximate solution, where $\rho$ is the approximation guarantee for the DkS problem.*

**Proof** Let us consider the iteration of the algorithm in which we guess the optimal load for the CHA instance, i.e., $L = L_{OPT}$. Let $(L_{OPT}, k')$ be the solution found during this iteration. We will now upper bound the value of $k'$. Let $m$ denote the number of edges in the graph. Let $m_i$ denote the number of uncovered edges at the beginning of the $i$th iteration of the while loop. Thus $m_1 = m$. An optimal solution to CHA would have one channel covering at least $m/k$ edges. The first call to DkS would return us a solution covering at least $\frac{m}{\rho k}$ edges, where $\rho$ is the performance guarantee of DkS. Thus, after the first round, the number of uncovered edges $m_2 \leq m_1 \left(1 - \frac{1}{\rho k}\right)$. In general, after $l$ rounds, $m_{l+1} \leq m_l \left(1 - \frac{1}{\rho k}\right)$. Since the algorithm terminates after $k'$ iterations, we must have at least one uncovered edge at the start of iteration $k'$, i.e., $m_{k'} \geq 1$. Thus we get

$$m_{k'} \leq m_{k'-1}(1 - 1/(\rho k)); \text{ i.e., } 1 \leq m(1 - 1/(\rho k))^{k'-1} \leq m \cdot e^{-\frac{k'-1}{\rho k}}.$$

Hence, $k' \leq \rho k \ln m + 1$. Using Lemma 2.1, we convert the $(L_{OPT}, k')$ solution to an $\left(\left\lceil \frac{k'}{k} \right\rceil L_{OPT}, k\right) = \left(\left\lceil \frac{\rho k \ln m + 1}{k} \right\rceil L_{OPT}, k\right)$ solution for CHA. Since our algorithm tries all possible values for the optimal load and returns the best solution, the cost of our solution is at most $\lceil \rho \ln m + 1 \rceil L_{OPT}$, which is an $O(\rho \ln m) = O(\rho \ln n)$-approximate solution. ■

**Theorem 3.3** *For a certain constant $a < 1/3$, there is an $O(n^a \ln n)$-approximation algorithm for CHA.*

**Proof** The best known approximation guarantee for DkS is $\rho \leq n^a, a < 1/3$ [9]. Combining this fact with Lemma 3.2 completes the proof. ■

## 4 THE EDGE PARTITIONING PROBLEM

We now present a sequential approximation algorithm for EP. We will throughout use hypergraph covering terminology: given a hypergraph $H = (V, E)$ with $n$ vertices and $m$ edges (each having at most $w$ 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"). We assume without loss of generality that $k \leq m$.

## 4.1 Algorithm and analysis

We now present a deterministic $O\left(w \cdot n^{\frac{w-1}{w+1}}\right)$-approximation algorithm; see Theorem 4.6. Recall that the *degree* of a vertex in a hypergraph is the number of edges incident to it (i.e., containing it). Let $H = (V, E)$ be the given hypergraph. We start by considering the following greedy algorithm where we keep removing vertices of lowest degree in the remaining hypergraph until the remaining hypergraph has at most $k$ edges.

```
EDGE PARTITION($H = (V, E), k$)
1   $F \leftarrow \emptyset$
2   while $|E| > k$ do
3       Remove the isolated vertices from $V$
4       $H' = (V', E') \leftarrow H = (V, E)$
5       $L \leftarrow \emptyset$
6       while $|E'| > k$ do
7           $u \leftarrow$ a lowest degree vertex in $H'$
8           $L \leftarrow$ {edges in $E'$ that are incident to $u$}
9           $V' \leftarrow V' \setminus \{u\}$
10          $E' \leftarrow E' \setminus L$
11      end
12      $R \leftarrow E' \bigcup L$
13      Arbitrarily remove some edges from $R$ to make $|R| = k$
14      $F \leftarrow F \bigcup \{R\}$ (i.e., $R$ is the set of edges assigned to a new channel)
15      $H \leftarrow H \backslash R$
16  end
17  $F \leftarrow F \bigcup \{E\}$
```

**Lemma 4.1** *For each iteration of the outer* **while** *loop (Lines 2–16), 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.*

**Proof** For each iteration of the outer **while** loop (Lines 2–16) we only need to calculate the number $p$ of iterations of the inner **while** loop (Lines 6–11) being executed. Because each iteration of the inner loop removes exactly one vertex from $V'$ (Line 9), the number of vertices in $R$ is upper bounded by $n' - p + 1$. Let $m'_i$ $(0 \le i \le p-1)$ be the number of edges in $E'$ and $n'_i$ be the number of vertices in $V'$ at the beginning of the $i$th iteration of the inner loop (or at the end of the $(i-1)$st iteration of the inner loop). Let $m'_0 = m'$ and $n'_0 = n'$. Since in the $i$-th iteration the average degree of each vertex is $\frac{wm'_i}{n'_i}$, removing a vertex of the minimum degree (Line 7) removes at most $\frac{wm'_i}{n'_i}$ edges from $E'$ (Line 8), therefore $m'_{i+1} \ge m'_i(1 - \frac{w}{n'_i})$ and thus $m'_{i+1} \ge m' \prod_{j=0}^{i}\left(1 - \frac{w}{n'_j}\right)$. Let $m'_p$ be the number of remaining edges after the inner while loop (Line 12). By construction of the algorithm, we know $n'_j = n' - j$, $m'_p \le k$ and $m'_{p-1} > k$. Meanwhile, since at the beginning of the $p$-th inner iteration there are at least $k+1$ edges in $H'$, there are at least $w+1$ vertices in $H'$. Hence $n' - (p-1) \ge w + 1$ and $m'_p \ge m' \prod_{j=0}^{p-1}\left(1 - \frac{w}{n'_j}\right)$. Thus we have $m' \prod_{j=0}^{p-1}\left(1 - \frac{w}{n'-j}\right) \le k$. Equivalently, $\frac{k}{m'} \ge \prod_{j=0}^{p-1} \frac{n'-j-w}{n'-j} = \prod_{i=0}^{w-1} \frac{n'-p-i}{n'-i}$. Since $n' - p \ge w$, we have $n' - p - i \ge \frac{n'-p}{w}$ for

$0 \leq i \leq w-1$. Hence we have $\prod_{i=0}^{w-1}(n'-p-i) \geq \left(\frac{n'-p}{w}\right)^w$. Also, since $\prod_{i=0}^{w-1}(n'-i) \leq (n')^w$, we get $n'-p+1 \leq w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n'+1 \simeq w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n'$. ■

**Lemma 4.2** *The total number of vertices in the edge partition is at most* $\frac{wn}{(1-1/w)}\left(\frac{m}{k}\right)^{1-1/w}$.

**Proof** Let $n_i$ and $m_i$ be the number of vertices and edges in the graph $H$ of the $i$-th iteration of the outer while loop. From Lemma 4.1 we know that the total number of vertices is at most

$$S = \sum_{i=0}^{\lfloor \frac{m}{k} \rfloor -1} w\left(\frac{k}{m_i}\right)^{\frac{1}{w}} \cdot n_i.$$

Since $n_i \leq n$ and $m_i = m - ik$ for $0 \leq i \leq \frac{m}{k}-1$, we get

$$S \leq \frac{wn}{k^{(1-1/w)}} \sum_{i=0}^{\lfloor \frac{m}{k} \rfloor -1} \frac{k}{(m-ik)^{1/w}} < \frac{wn}{k^{(1-1/w)}} \int_0^m \frac{dx}{x^{1/w}} = \frac{wn}{(1-1/w)}\left(\frac{m}{k}\right)^{1-1/w}.$$

■

**Lemma 4.3** *The optimal solution has at least* $\max\left\{n, \frac{w/e}{k^{1-1/w}}m\right\} \geq n^{1/w} \cdot \frac{(w/e)^{1-1/w}}{k^{(1-1/w)^2}} \cdot m^{1-1/w}$ *vertices.*

**Proof** Suppose the optimal solution partitions the set of edges into $t$ parts, and part $i$ contains $m_i$ edges, $1 \leq i \leq t$. If $n_i$ is the number of vertices in part $i$, then $\binom{n_i}{w} \geq m_i$ (since a complete hypergraph on $n_i$ vertices has $\binom{n_i}{w}$ edges). Using Stirling's formula, it is easy to verify that $n_i \geq \left(\frac{w}{e}\right) m_i^{\frac{1}{w}}$. Therefore the total number of vertices is $S \geq \frac{w}{e}\sum_{i=1}^t m_i^{\frac{1}{w}}$. Obviously, $0 < m_i \leq k$ and $\sum_{i=1}^t m_i = m$. Because the function $f(x) = x^{\frac{1}{w}}$ is a concave function for $x \geq 0$ (since $f''(x) < 0$), hence we have $x^{\frac{1}{w}} > \frac{x}{k^{1-1/w}}$ for $0 < x \leq k$. Therefore $S > \frac{w/e}{k^{1-1/w}}m$. On the other hand, any solution has at least $n$ vertices. Hence the number of vertices in the optimal solution is at least $\max\left\{n, \frac{w/e}{k^{1-1/w}}m\right\}$. Since $\max\{a,b\} \geq a^p b^q$ for $a,b,p,q > 0$ and $p+q=1$, let $a=n$, $b=\frac{w/e}{k^{1-1/w}}m$, $p=\frac{1}{w}$ and $q = 1-\frac{1}{w}$, and we obtain the claimed result. ■

**Lemma 4.4** *From Lemmas 4.2 and 4.3, 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 [12] works for $w=2$, and their approximation ratio for $w=2$ is about $\sqrt{\frac{k}{2}}$.

**Lemma 4.5** *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 $ek^{1-1/w}$.*

**Proof** The cost of the trivial algorithm is $wm$, while the cost of the optimal solution is at least $\frac{w/e}{k^{1-1/w}}m$. Therefore the approximation ratio is at most $ek^{1-1/w}$. ■

**Theorem 4.6** *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* $O\left(\frac{m}{k}(m+n)\right)$.

**Proof** The approximation ratio is at most $r = \min\left\{ek^{1-1/w}, \frac{w^2}{w-1}\left(\frac{en}{wk^{1/w}}\right)^{1-1/w}\right\}$. If $k \leq \left(\frac{w^2}{e(w-1)}\right)^{\frac{w^2}{w^2-1}} \left(\frac{en}{w}\right)^{\frac{w}{w+1}}$, then $r \leq ek^{1-1/w} \leq w\left(\frac{e}{w-1}\right)^{\frac{w}{w+1}} n^{\frac{w-1}{w+1}}$. Else, $r \leq \frac{w^2}{w-1}\left(\frac{en}{wk^{1/w}}\right)^{1-1/w} \leq w\left(\frac{e}{w-1}\right)^{\frac{w}{w+1}} n^{\frac{w-1}{w+1}}$. $f(x) = \left(\frac{e}{x-1}\right)^{\frac{x}{x+1}}$ is a decreasing function when $x \geq 2$ (since $f'(x) < 0$). Hence, $\left(\frac{e}{w-1}\right)^{\frac{w}{w+1}} \leq e^{2/3} < 2$. Thus the approximation ratio is at most $r < 2w \cdot n^{\frac{w-1}{w+1}}$. Furthermore, the running-time bound is easily verified. ∎

## 5 A DISTRIBUTED ALGORITHM FOR EP ON GRAPHS

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 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.

One representative use of such distributed algorithms is as follows. Suppose, via statistical knowledge of user-requests, the servers have been clustered so that most users will request their (two) topics from within the same cluster. (For instance, the topics could be in different languages, suggesting a natural clustering: since a typical user may request two topics of the same language, we could cluster all topics of the same language together.) Then, a distributed algorithm will primarily require local communication, and will not have much need for inter-cluster communication.

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 – we assume that a channel can 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 $c$.) We first consider the strong model in Section 5.1, and then show how to modify the algorithm to work in the weak model, in Section 5.2.

## 5.1 The Strong Model

Consider the strong model. Define

$$k_0 \doteq (n^2 m (\log n)^4 (\log\log n)^4)^{1/3}.$$

For a technical reason that will become apparent later, we will first do the following. If $k > k_0$, we will artificially assume that $k = k_0$; i.e., if the channel capacity is more than $k_0$, we will restrict ourselves to using at most $k_0$ capacity per channel. 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 $2m$. Like in Section 4.1, our main 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. (Each processor can see which is the better of the two solutions, since it knows the value of $m$. Also, we emphasize that when we say "$O(n\sqrt{m/k})$" a few sentences above, we mean "$O(n\sqrt{m/k'})$, where $k' = \min\{k, k_0\}$".) Such an $O(n^{1/3})$–approximation can be justified as follows. Recall from Lemma 4.3 that $\Omega(\max\{n, m/\sqrt{k}\})$ is a lower bound on the objective function. Since for any $a, b > 0$ and $p, q > 0$ such that $p + q = 1$, we have $\min\{a, b\} \le a^p b^q$, therefore $\min\{m, n\sqrt{m/\min\{k, k_0\}}\} \le m^{1/3}(n\sqrt{m})^{1/3}$. Also due to the facts that $\max\{n, m/\sqrt{k}\} \ge n$ and $m = O(n^2)$, it is easy to verify that for all $k$ (whether greater than $k_0$ or not),

$$\frac{\min\{m, n\sqrt{m/\min\{k, k_0\}}\}}{\max\{n, m/\sqrt{k}\}} = O(n^{1/3}).$$

Henceforth, we assume that $k \le k_0$.

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 $p$. If $k \ge \log^2 n$, we take, say, $6(m/k)\log n$ channels and $p = k/(2m)$; each edge tries each channel with probability $p$, and goes to the first channel it chose. If $k < \log^2 n$, we take $p = n^{-5}$ and $(6/p)\log n$ channels. We now verify that with high probability, these schemes yield feasible solutions. In the case where $k \ge \log^2 n$, the expected load on a channel is $mp = k/2$; by the Chernoff-Hoeffding bounds [5, 14], the probability that some particular channel gets more than $k$ edges is $e^{-\Omega(k)}$. Since there are only $6(m/k)\log n$ channels and since $k \ge \log^2 n$, a union bound implies that the probability of getting an overloaded channel is negligible. Furthermore, for any particular edge $f$, the probability that it does not choose any channel is

$$(1-p)^{6(m/k)\log n} \le e^{-6p(m/k)\log n} = e^{-3\log n}. \tag{1}$$

Since there are at most $n^2$ edges, we see by the union bound that the probability of some edge not getting assigned to any channel is negligible. Thus, with high probability, we get a feasible solution. We argue similarly for the case where $k < \log^2 n$: recall that we take $p = n^{-5}$ and $(6/p)\log n$ channels in this case. The probability that a particular channel gets two or more edges assigned to it is at most $m^2p^2 \le n^{-6}$; a union bound over the $6n^5\log n$ channels shows that with high probability, no channel gets overloaded. As for each edge getting assigned to some channel, we argue similarly as in (1).

For the rest of this discussion, we assume $k \geq \log^8 n$, say; if $k$ is smaller, the above trivial algorithm already results in a polylog$(n)$ approximation, since the optimal solution value is $\Omega(m/\sqrt{k})$ by Lemma 4.3. Assuming $k \geq \log^8 n$, we now show how to distributively construct a feasible solution with objective function value $O(n\sqrt{m/k})$.

We first give an informal description of our algorithm. The algorithm will proceed in iterations, and each iteration has a preprocessing step followed by a random selection step. Define $\bar{d} = \left\lceil \frac{2m}{n} \right\rceil$, 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 $\lceil deg(v)/\bar{d} \rceil$ 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 $2n$, as follows. For any vertex $v$, we have $1 \leq deg(v) \leq n-1$. Let $l = \lceil (n-1)/\bar{d} \rceil$. Suppose there are $n_1$ vertices having degrees in $[1, \bar{d}]$, $n_2$ vertices having degrees in $[\bar{d}+1, 2\bar{d}]$, $n_3$ vertices having degrees in $[2\bar{d}+1, 3\bar{d}]$, ..., $n_l$ vertices having degrees in $[(l-1)\bar{d}+1, l\bar{d}]$. Then we have $n_1 + n_2\bar{d} + 2n_3\bar{d} + 3n_4\bar{d} + \cdots + (l-1)n_l\bar{d} \leq n\bar{d}$. Hence $n_2 + 2n_3 + \cdots + (l-1)n_l \leq n$. Therefore $n_1 + 2n_2 + 3n_3 + \cdots + ln_l \leq 2n$. Thus, the new number of vertices is in the range $[n, 2n]$, there are $m$ edges, and the maximum degree at most $2m/n$; so, the maximum degree is at most twice the average degree. We will see below that controlling the maximum degree in this fashion greatly helps our analysis. The random selection step is: every vertex will try a channel with probability approximately $\sqrt{k/m}$, so that approximately $k$ edges will get assigned to the channel. More precisely, the choices for the virtual copies of an original vertex $v$ are all made independently by $v$. Thus, each iteration successfully assigns some edges, and the remaining edges move on to the next iteration.

The above informal description assumes that every vertex knows the residual number of edges; however, this number can only be estimated. We now give details of this estimation. (Getting all vertices to determine this number exactly by flooding, i.e., sending to the neighbors their local information and information received from other neighbors, would take time proportional to the diameter of the graph.) The algorithm actually proceeds in two phases, where the iterations of Phase I are informally described above, and Phase II is an invocation of the trivial algorithm. We now describe the two phases more formally.

**Phase I.** We first wish to define $a = a(n)$ to be a quantity that is asymptotically slightly smaller than $1/\log n$; we will take

$$a = \frac{1}{(\log n)\log\log n}. \tag{2}$$

Recall that we have assumed that $k \leq k_0$; we may also assume without loss of generality that the graph $G$ is connected, and hence that $m \geq n-1$. Thus, if $m \leq k/a^2$, we must then have that $n, m$ and $k$ are all within polylogarithmic factors of each other, and the trivial algorithm will hence be a polylogarithmic approximation. So, suppose $m > k/a^2$. The number of residual edges will decay by a factor of about $(1-a/2)$ in every iteration; define $i^*$ to be the largest nonnegative integer $i$ such that

$$m(1-a/2)^i \geq k/a^2. \tag{3}$$

Note that $i^* \leq O((\log n)/a)$. Phase I proceeds in iterations numbered $i = 0, 1, 2, \ldots, i^*$; the $i$th iteration is as follows. Let $E_i$ be the set of edges at the beginning of the iteration; i.e., these edges have not been assigned successfully until now. All vertices start with an estimate $m_i$ of $|E_i|$.

Initially, $m_0 = m = |E|$ is obtained from the system. We will show that the number of residual edges decays, with high probability, by a factor very close to $(1 - a/2)$ in every iteration; thus we define

$$m_i = m(1 - a/2)^i, \ 1 \leq i \leq i^*, \tag{4}$$

and will show later that $|E_i| = m_i(1 \pm o(1))$ with high probability, for all $i$. Now, in the $i$th iteration, the preprocessing step is as described in the informal description, with $m$ replaced by $m_i$. The server chooses $am_i/k$ new channels. Each vertex then independently goes (following its copies) into each of these channels with probability $p_i = \sqrt{k/2m_i}$. (The choices for all virtual copies of an original vertex $v$ are 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.

**Phase II.** Once we succeed in proving that $|E_i| = m_i(1 \pm o(1))$ with high probability for all $i$, we see from (3) that with high probability, we will have $O(k/a^2)$ edges left at the end of Phase I. In Phase II, we run the trivial algorithm to assign these edges. We will prove that the cost of Phase I is $O(n\sqrt{m/k})$ with high probability; also, the cost of Phase II is $O(k/a^2)$. Thus, the total cost is

$$O(n\sqrt{m/k}) + O(k/a^2) = O(n\sqrt{m/k});$$

this is where we use the assumption that $k \leq k_0$.

We will describe and analyze only Phase I of the algorithm below. The $i$th iteration, $0 \leq i \leq i^*$, of Phase I is as follows. The channel-server allocates $am_i/k$ new channels, where $a = \frac{1}{(\log n)\log\log n}$. The pseudocode for each vertex $v$ is:

```
EDGE PARTITION($G(V,E), k, v, i$)   [Strong Model]
1    Let $m_i$ be as in (4);
2    $\bar{d}_i \leftarrow \lceil 2m_i/n \rceil$; $p_i \leftarrow \sqrt{k/(2m_i)}$;
3    if all edges incident to $v$ have been assigned
4        then return // preprocessing step
5    $deg(v) \leftarrow$ number of unassigned edges incident to $v$;
6    $v$ makes $\lceil deg(v)/\bar{d}_i \rceil$ virtual copies of itself
     and equitably distributes its edges to the copies; // random selection step
7    Independently put each copy of $v$ in each channel with probability $p_i$;
8    // Let $u_c$ be a virtual copy of vertex $u$, $v_c$ be a virtual copy of $v$;
9    if $\exists$ a channel CH s.t. $u_c \in$ CH and $v_c \in$ CH
10       then edge $(u,v)$ is assigned to channel CH.
```

### 5.1.1 The Strong Model: Analysis

We now turn to analyzing the algorithm. Define an event to have *negligible* probability if its probability is $n^{-\omega(1)}$, i.e., goes to zero faster than any inverse polynomial of $n$. In the other direction, define an event to happen *with very high probability* ("w.v.h.p.") if its complement has negligible probability. We will also assume as stated before that $\log^8 n \leq k \leq k_0$. Define $U_i$ to be the following event that is desirable for us:

$$m(1 - a/2 - a^2/4)^i \leq |E_i| \leq m(1 - a/2 + a^2/4)^i. \tag{5}$$

We aim to show that w.v.h.p: all the events $U_i$ hold, no channel is ever overloaded, and that the total cost of Phase I is $O(n\sqrt{m/k})$.

**Remark.** Note that $U_i$ implies that $|E_i| = m_i(1 \pm o(1))$. If $U_i$ holds, then

$$|E_i| = m(1 - a/2)^i \cdot (1 \pm O(a^2))^i = m_i \cdot (1 \pm O(a^2 i));$$

since $i \leq O((\log n)/a)$ and $a = o(1/\log n)$, we get that $|E_i| = m_i(1 \pm o(1))$.

We now proceed to show that the required events happen w.v.h.p. Fix an iteration $i$. Let $\mathcal{E}$ be the event $U_0 \wedge U_1 \wedge \cdots \wedge U_i$. Conditional on $\mathcal{E}$, we will show that the following three events happen w.v.h.p:

**(C1)** $U_{i+1}$,

**(C2)** no channel is overloaded in iteration $i$, and

**(C3)** the total cost of iteration $i$ is $O(a \cdot n \cdot \sqrt{m_i/k})$.

Given this, a union bound over all $i$ implies that w.v.h.p., all events $U_j$ hold, and no channel is ever overloaded. It also implies that w.v.h.p., the total cost is at most

$$O\left(\sum_{i \geq 0} a \cdot n \cdot \sqrt{m(1 - a/2)^i/k}\right) = O(n \cdot \sqrt{m/k}).$$

Thus, we are studying iteration $i$, and are conditioning on $\mathcal{E}$. We will only require the fact that $U_i$ holds, which is implied by $\mathcal{E}$. Let us first show (C3). Consider any channel; the load on it is a sum of independent binary random variables (one for each vertex attempting to use the channel in this iteration), and the mean load is $O(n\sqrt{k/m_i})$. Thus, by a Chernoff-Hoeffding bound, the probability that the load is more than twice its mean is at most

$$e^{-\Omega(n\sqrt{k/m_i})} \leq e^{-\Omega(\sqrt{k})} \leq e^{-\Omega(\log^4 n)},$$

which is negligible. Thus, by a union bound, (C3) holds w.v.h.p.

We now move on to (C1) and (C2), where the situation is much more complicated due to correlations among neighboring edges. However, we will see that the fact "maximum degree is within a constant times the average degree," which follows from our pre-processing in iteration $i$, will help a great deal in bounding these correlations.

**Lower-bounding $|E_{i+1}|$.** We first show a lower bound on $|E_{i+1}|$ that will hold w.v.h.p., given that $U_i$ holds. We will use Janson's lower-tail bound [16], which is as follows. Let $\Phi$ be a finite set, and $R \subseteq \Phi$ determined by the experiment in which each element $r \in \Phi$ is independently included in $R$ with probability $p_r$. Let $\{A_i \mid i \in I\}$ be a family of subsets of $\Phi$, and denote by $B_i$ the event that $A_i \subseteq R$. Write $i \sim j$ if $i \neq j$ and $A_i \cap A_j \neq \emptyset$. Define $\Delta = \sum_{i \sim j} \Pr[B_i \wedge B_j]$ (the sum is over ordered pairs). Let $X = \sum_i X_i$, where $X_i$ is an indicator variable for the event $B_i$, let $\mu_i = \mathbf{E}[X_i] = \Pr[B_i]$ and $\mu = \mathbf{E}[X] = \sum_i \mu_i$.

**Theorem 5.1** *(Janson's inequality [16].) With the notation as above and with* $0 \leq \delta \leq 1$,

$$Pr[X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu/(2 + \Delta/\mu)}.$$

Let $q$ be the probability that a particular edge is assigned to at least one channel; we have

$$q = 1 - (1 - k/(2m_i))^{am_i/k} \geq \frac{a}{2}\left(1 - \frac{a}{4}\right), \tag{6}$$

where we have used the inequality $(1-\psi)^r \leq 1 - r\psi + r^2\psi^2/2$, valid for $0 \leq \psi \leq 1$ and integral $r \geq 1$. In the setting of Theorem 5.1, we take $\Phi = V(G)$, and each $A_i$ is an edge. Let $X_e$ be the event that edge $e$ is assigned to some channel. Define $e \sim f$ iff the edges $e$ and $f$ are different, and have a common end-point. Since $\Pr[X_e \wedge X_f] \leq \Pr[X_e] = q$, we have

$$\Delta \doteq \sum_{(e,f):\ e\sim f} \Pr[X_e \wedge X_f] = \sum_e \sum_{f:\ e\sim f} \Pr[X_e \wedge X_f] \leq 2|E_i|(\bar{d_i} - 1)q.$$

Letting $X = \sum_e X_e$ and $\mu = \mathbf{E}[X] = q|E_i|$, we get $\Delta/\mu \leq 2|E_i|(\bar{d_i}-1)q/(q|E_i|) = 2(\bar{d_i}-1)$; this in turn is $O(m_i/n)$, since $U_i$ is assumed to hold. Also, $\mu = \Theta(m_i q)$ since $U_i$ is assumed to hold. So, since

$$\mu = \Theta(m_i q) \text{ and } \Delta/\mu \leq O(m_i/n), \tag{7}$$

Janson's inequality shows that

$$\Pr[X \leq \mu(1 - a/4)] \leq e^{-\Omega(m_i q a^2/\max\{1, m_i/n\})} = e^{-\Omega(\min\{m_i, n\}\cdot a^3)}, \tag{8}$$

which is negligible since $m_i \geq k \geq \log^8 n$. Thus, we have w.v.h.p. (conditional on $\mathcal{E}$) that

$$|E_{i+1}| \leq |E_i| \cdot (1 - q \cdot (1 - a/4)) \leq |E_i| \cdot (1 - a/2 + a^2/4). \tag{9}$$

**Upper-bounding the channel load.** To establish (C1) and (C2), we next show that no channel is overloaded, w.v.h.p; we use recent tail-bounds of [17, 22]. In particular, we adopt the following formulation of the Janson-Ruciński's inequality [17].

If $\Gamma$ is a set and $\kappa \geq 1$ a natural number, let $[\Gamma]^\kappa$ denote the family of all subsets $I \subset \Gamma$ with $|I| = \kappa$, and $[\Gamma]^{\leq\kappa} = \bigcup_{j=0}^{\kappa} [\Gamma]^j$. Let $\mathcal{A}$ and $\Gamma$ be finite index sets, and suppose the following scenario holds. There is a family $\xi_\alpha$, $\alpha \in \mathcal{A}$, of *independent* random variables. We also have *non-negative* random variables $Y_I$, $I \in \mathcal{H} \subseteq [\Gamma]^{\leq\kappa}$, with the following properties:

- we have a family of subsets $\mathcal{A}_I \subseteq \mathcal{A}$, $I \in [\Gamma]^{\leq\kappa}$, such that each $Y_I$ is a function of the tuple $\langle \xi_\alpha : \alpha \in \mathcal{A}_I \rangle$ alone;
- $\mathcal{A}_\emptyset = \emptyset$ and $\mathcal{A}_I \cap \mathcal{A}_J = \mathcal{A}_{I\cap J}$ for all $I, J \in [\Gamma]^{\leq\kappa}$.

Let $X = \sum_{I\in\mathcal{H}} Y_I$. We want an upper-tail bound for $X$; to do so, we define certain quantities $X_I$, $\mu_I$ and $\mu_l$ next.

Define $X_I = \sum_{J\supseteq I} Y_J$. Next, define

$$\mu_I = \sup \mathbf{E}[X_I \mid \xi_\alpha, \alpha \in \mathcal{A}_I];$$

the supremum is taken over all possible values for the tuple $\langle \xi_\alpha : \alpha \in \mathcal{A}_I \rangle$ that can occur with nonzero probability. Further let, for $l \leq \kappa$,

$$\mu_l = \max_{|I|=l} \mu_I.$$

Note that $\mu_0 = \mu = \mathbf{E}[X]$. We are now ready to state the upper-tail bound:

**Theorem 5.2** *(Janson-Ruciński's inequality [17].) With notation as above and $N = |\Gamma|$, for every $t > 0$ and $r_1, \ldots, r_\kappa$ such that*

$$r_1 \cdots r_j \cdot \mu_j \leq t, \quad j = 1, \ldots, \kappa,$$

*we have, with $c = \frac{1}{8\kappa}$,*

$$Pr[X \geq \mu + t] \leq \left(1 + \frac{t}{\mu}\right)^{-cr_1} + \sum_{j=1}^{\kappa-1} N^j \left(1 + \frac{t}{r_1 \cdots r_j \mu_j}\right)^{-cr_{j+1}}.$$

Armed with this theorem, we can now show that for any particular channel, it does not get overloaded, w.v.h.p. In the discussion now, we consider all virtual copies of a vertex as separate vertices. Fix a channel. In the setting of Theorem 5.2, we take $\Gamma$ to be the vertex set $V$, $\kappa = 2$, and $\mathcal{H}$ as the set of all vertices and edges of the current residual graph. Let $Y_e$ be the indicator variable that edge $e$ gets assigned to the channel (i.e., both of its end-points attempt to join the channel). Define $Y_{\{v\}} \equiv 0$ for all $v \in V$. Let $\mathcal{A} = V$, and let $\xi_v$ be the event that $v$ attempts to join the channel. We now see what the quantities $X_I$ and $\mu_l$ are in our context. We have $X_v = \sum_{e:v\in e} Y_e$. Since $U_i$ holds, $\mu_1 = \max_v \mathbf{E}[X_v \mid \xi_v] = \Theta(\frac{\sqrt{k m_i}}{n})$; $\mu_2 = \max_e \sup X_e = 1$. Let $X = \sum_e Y_e$. Then, $\mu = \mathbf{E}[X] = |E_i| \cdot \frac{k}{2m_i}$; since $U_i$ holds, $\mu \sim k/2$. Let $t = \mu a/2$; note that $t \sim ka/4$. The constraints required by Theorem 5.2 are that $r_1\mu_1 \leq t$ and $r_1 r_2 \mu_2 \leq t$. We choose $r_1$ of the form $\Theta(a\sqrt{k})$ and $r_2$ of the form $\Theta(\sqrt{k})$ to satisfy these constraints. With these choices, Theorem 5.2 yields

$$\Pr\left[X \geq (1 + a/2) \cdot |E_i| \cdot \frac{k}{2m_i}\right] \leq e^{-\Omega(\sqrt{k}\delta^2)} + ne^{-\Omega(\sqrt{k})}, \tag{10}$$

which is negligible since $k \geq \log^8 n$.

Now, (10) implies that (C2) holds w.v.h.p. Furthermore, we get that w.v.h.p., the total number of edges assigned in this iteration is at most

$$(am_i/k) \cdot (1 + a/2) \cdot |E_i| \cdot \frac{k}{2m_i} = (a/2) \cdot (1 + a/2) \cdot |E_i|. \tag{11}$$

Therefore,

$$|E_{i+1}| \geq |E_i| \cdot (1 - a/2 - a^2/4). \tag{12}$$

Along with (9), this implies that since $U_i$ is true, $U_{i+1}$ is also true; thus, we have established (C1) also. This completes our analysis of Phase I of the algorithm, and hence also the analysis of our algorithm for the Strong model.

## 5.2 The Weak Model

We may assume here that $k \geq \log^2 n$. Indeed, if $k < \log^2 n$, we can run the trivial algorithm for the "$k < \log^2 n$" case from the early paragraphs of Section 5.1, to achieve a polylogarithmic approximation. (Note that this algorithm does not require knowledge of the value of $m$.) We show a simple algorithm to guess the value of $m$ with high accuracy by a contention-resolution-type procedure; we can then run the algorithm for the Strong model.

Let $\epsilon > 0$ be a sufficiently small constant. We proceed for $\log_{1+\epsilon} n^2 = O(\log n)$ iterations, numbered $0, 1, 2, \ldots$. In iteration number $i$, the server allocates a new channel, and each edge independently tries the channel with probability $1/(1+\epsilon)^i$. Recall that $k \geq \log^2 n$, and that $\epsilon$ is a constant. A simple application of the Chernoff-Hoeffding bounds shows that the following hold with high probability:

- if $(1+\epsilon)^i \leq m(1-\epsilon)/k$, then the channel gets overloaded in iteration $i$; and
- if $(1+\epsilon)^i \geq m(1+\epsilon)/k$, then the channel does not get overloaded in iteration $i$.

Thus, the server is able to come up with a very accurate estimate of $m$ with high probability, which it can then advertise; this information can then be read by the vertices. We can now run the algorithm for the Strong model.

## 6 CONCLUSION

We have presented new approximation algorithms for two related problems in broadcast-scheduling; a natural open question is to find the approximation threshold for these problems. As a related problem, we note again that it is not even known if the DkS problem is MAX-SNP hard; this problem is fairly basic, and it would be good to understand its approximability. Regarding our second general theme of distributed algorithms, an immediate open question is whether we can get good distributed approximation algorithms for the hypergraph version of EP. In general, we believe that more research is required in the area of distributed approximation algorithms.

**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, *Networks* referees, and Douglas Shier for their helpful comments; in particular, we thank a *Networks* referee for her/his proof that we have presented in Section 2.3.

## References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, Broadcast disks: Data management for asymmetric communication environments, *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1995, pp. 199–210.

[2] S. Acharya, M. Franklin, and S. Zdonik, Balancing push and pull for data broadcast, *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1997, pp. 183–194.

[3] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik, Research in data broadcast and dissemination, *Proc. of the International Conference on Advanced Multimedia Content Processing*, 1998, pp. 194–207.

[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**(1952), pp. 493–509.

[6] V. Chvátal, A greedy heuristic for the set-covering problem, *Mathematics of Operations Research* **4**3(1979), pp. 233–235.

[7] 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**(1997), pp. 1617–1638.

[8] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 717–724.

[9] U. Feige, G. Kortsarz, and D. Peleg, The dense $k$-subgraph problem, *Algorithmica* **29**(2001), pp. 410–421.

[10] M. Franklin and S. Zdonik, A framework for scalable dissemination-based systems, *Proc. of the ACM Conference on Object Oriented Programming Systems, Languages and Applications*, 1997, pp. 94–105.

[11] M. Franklin and S. Zdonik, "Data in your face": push technology in perspective, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998, pp. 516–519.

[12] O. Goldschmidt, D. Hochbaum, A. Levin, and E. Olinick, The SONET edge partition problem, *Networks* **41**(2003), pp. 13–23.

[13] D. A. Grable and A. Panconesi, Nearly optimal distributed edge coloring in $O(\log \log n)$ rounds, *Random Structures & Algorithms* **10**(1997), pp. 385–405.

[14] W. Hoeffding, Probability inequalities for sums of bounded random variables, *American Statistical Association Journal* **58**(1963), pp. 13–30.

[15] I. Holyer, The NP-completeness of some edge-partition problems, *SIAM Journal on Computing* **10**(1981), pp. 713–717.

[16] S. Janson, Poisson approximations for large deviations, *Random Structures & Algorithms* **1**(1990), pp. 221–230.

[17] S. Janson and A. Ruciński, The deletion method for upper tail estimates, Technical Report 2000:28, Department of Mathematics, Uppsala University, Sweden, 2000.

[18] L. Jia, R. Rajaraman, and T. Suel, An efficient distributed algorithm for constructing small dominating sets, *Proc. of the 20th ACM Symposium on Principles of Distributed Computing*, 2001, pp. 33–42.

[19] D. S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences* **9**(1974), pp. 256–278.

[20] F. Kuhn and R. Wattenhofer, Constant-time distributed dominating set approximation, *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing*, 2003, pp. 25–32.

[21] L. Lovász, On the ratio of optimal integral and fractional covers, *Discrete Mathematics* **13**(1975), pp. 383–390.

[22] V. H. Vu, Concentration of non-Lipschitz functions and applications, *Random Structures & Algorithms* 20(2002), pp. 262–316.

[23] J. Wong, Broadcast delivery, *Proc. of the IEEE* **76**(1988), pp. 1566–1577.