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 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

In this work 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, without a loss in the approximation guarantee.

We study problems that arise in the area of dissemination-based information systems [1, 2, 11, 12, 20]. Such systems are used in application domains such as public-safety systems, election-result servers and stock tickers [3]. One of the characteristics of dissemination-based applications is that there is a high degree of overlap in the user needs. Since many of the 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

*Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965. E-mail: gandhi@cs.umd.edu.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This research was supported by NSF Award CCR-9820965 and CCR-0113192. E-mail: samir@cs.umd.edu.

‡Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Supported in part by NSF Award CCR-0208005. E-mail: srin@cs.umd.edu.

§Department of Computer Science, University of Maryland, College Park, MD 20742. Supported in part by NSF Award CCR-0208005. E-mail: nwang@cs.umd.edu.

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 the model that we work with, 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 the problem of 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 [16, 17, 19]. 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 [18, 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.

We omit some details in this extended abstract. Proofs of most claims are given in the Appendix.

2 The Channel Allocation Problem

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

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 input that cover the entire spectrum of inputs.

The first algorithm is the following simple randomized algorithm. *Independently* place each topic on each channel, $i, 1 \le i \le k$, 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] 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 $i, 1 \le i \le \binom{t}{w}$, 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 $|T_z| \le t$. The size of our set cover instance, $|I| = \sum_{j=1}^{t} \binom{|T|}{j} \le \sum_{j=1}^{t} |T|^j = O(|T|^t) = O(n^t) = O(m^t)$. 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 the 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

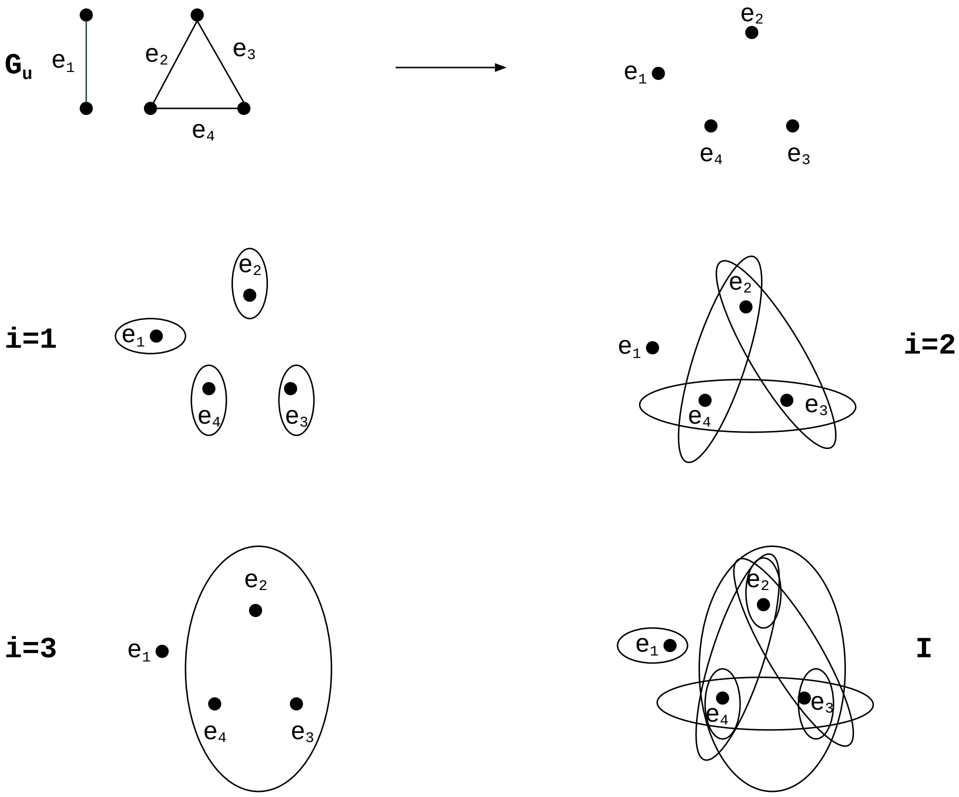

Figure 1: Construction of the set cover instance. Graph G_u represents the user requests. The topics are represented by vertices and the requests are represented by edges. $t = 3, w = 2$.

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

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 2.1 *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 2.2 *With a sufficiently high probability, the randomized algorithm gives a $(O(n(\frac{\lg n}{k})^{\frac{1}{w}}), k)$ solution.*

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

Theorem 2.4 *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.*

2.3 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 in exposition, we prove Theorem 2.5 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 2.5 *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.

$$I \in \text{3DM} \implies \text{CHA}(f(I)) \leq 12 \quad (1)$$
$$I \notin \text{3DM} \implies \text{CHA}(f(I)) \geq 13 \quad (2)$$

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 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 2. 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 2. 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

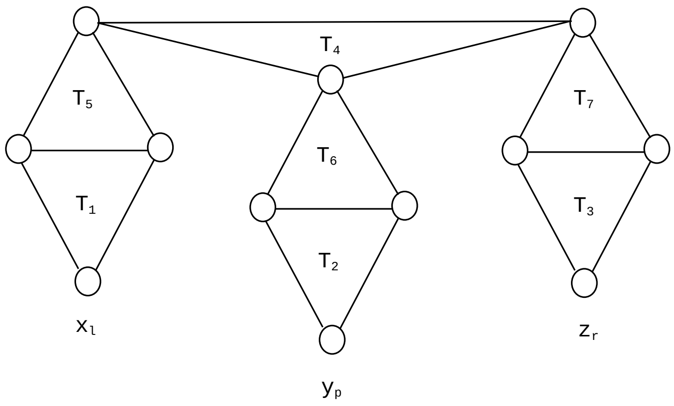

Figure 2: Gadget corresponding to each triple.

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 2.6 *If 3DM(I)* $= q$ *then CHA*$(f(I)) \leq 12$.

Claim 2.7 *If 3DM(I)* $< q$ *then 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 $\max_{1 \leq i \leq k} L_i$. In Section 3.1 we show that the problem is NP-complete even when there are only two 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.*

3.2 Algorithm

In this section we give an approximation algorithm for CHA. Our algorithm uses the solution to Dense k-Subgraph problem (DkS) which is defined below.

DkS: Given a graph G and a positive integer k, find a set of k vertices with maximum number of edges in the subgraph induced by this set.

DkS is NP-hard. Let DkS(G, k) be a procedure that returns a ρ-approximate solution to DkS. We will use this procedure in the algorithm below.

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. The pseudo-code for the algorithm is given below.

CHANNEL-ALLOCATION$(G(V, E), k)$
  $n \leftarrow |V|$
  for $L = 2$ to n **do**
    $i \leftarrow 0$
    $G_L(V_L, E_L) \leftarrow G(V, E)$
    while$(|E_L| > 0)$ **do**
      $i \leftarrow i + 1$
      $CH_i \leftarrow$ DkS(G, L)
      $E_L \leftarrow E_L \setminus C_i$ /* C_i are edges covered on CH_i */
    $S_L \leftarrow \lceil \frac{i}{k} \rceil L$
  return $\min\{S_1, S_2, \ldots, S_n\}$

Note that the algorithm could be made more efficient, for e.g., 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 a* $O(\rho \lg n)$*-approximate solution, where* ρ *is the approximation guarantee for the DkS problem.*

Theorem 3.3 *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 4.6. 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$(H = (V, E), k)$
  $F \leftarrow \emptyset$
  While $|E| > k$ **do**
    Remove the isolated vertices from V

```
        H' = (V', E') ← H = (V, E)
        L ← ∅
        While |E'| > k do
                u ← a lowest degree vertex in H'
                L ← {edges in E' that are incident to u}
                Remove u from V' and remove L from E' to get an updated H' = (V', E')
        End
        R ← E' ⋃ L
        Arbitrarily remove some edges from R to make |R| = k
        F ← F ⋃ {R} (i.e., R is the set of edges assigned to a new channel)
        H ← H\R
End
F ← F ⋃ {E}
```

Lemma 4.1 *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 4.2 *The total number of vertices in the edge partition is at most $\frac{wn}{(1-1/w)}\left(\frac{m}{k}\right)^{1-1/w}$.*

Lemma 4.3 *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 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 [14] 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 $ek^{1-1/w}$.*

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 $O(\frac{m}{k}(m+n))$.*

Note that when $w = 2$, the approximation ratio is at most $4n^{1/3}$. While in [14], the algorithm only works for $w = 2$ and their approximation ratio for $w = 2$ is about $\sqrt{\frac{k}{2}}$.

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 $G = (V, E)$.

Distributed model and results obtained. 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 number of vertices has not increased much. For any vertex v, we have $1 \leq deg(v) \leq n-1$. Let $l = \lceil \frac{n-1}{\bar{d}} \rceil$. 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 $[2\bar{d}+1, 3\bar{d}]$, $\cdots$, 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 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

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

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, 15]. 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 [16]. 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 $O(n(\bar{d})^2p^3) = O(k^{3/2})$. 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 [17, 19] 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.

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] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13-30, 1963.

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

[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. ACM Symposium on Principles of Distributed Computing*, pages 33–42, 2001.

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

[20] J. Wong. Broadcast Delivery. In *Proc. of the IEEE*, 76(12), 1988.

5 Appendix

Proof of Lemma 2.1

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

Proof of Lemma 2.2

Proof Throughout, we let $\Pr[\cdot]$ and $\mathbf{E}[\cdot]$ denote probability and expectation, respectively. 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 random variable $L \doteq \max_i L_i$. 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, 15] 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 $\delta = c_2\sqrt{\ln n/\mu} = c_2\sqrt{\ln n/np}$. 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. $\Pr\left[\overline{E_i}\right] \leq (1-p^w)^k$. Using the union bound, we get

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

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 $c_2\sqrt{\ln n/np} \le 1$ as required, since $2\ln m \le k \le (\frac{n}{\ln n})^w$. 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 $\Pr[E_o \bigvee E_u] \le n^{w-c_1} + c$. This probability can be reduced by repeating the random process. This completes the proof. ∎

Proof of Lemma 2.3

Proof The set cover obtained using the greedy algorithm has size at most $O(\lg M)$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 (L_{OPT}, k) solution into a (t, k') solution (t is the constant used in the algorithm) then $k' \ge$ 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 every combination of w groups on a separate channel to make sure that every request that was satisfied by h is still satisfied. Thus, 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$\le O((L_{OPT})^w k)$. 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 a $(O((L_{OPT})^w), k)$ feasible solution. ∎

Proof of Theorem 2.4

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

Case I: $L_{OPT} \le n^{\frac{1}{w+1}}$

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 $O(n^{\frac{w-1}{w+1}})$.

Case II: $L_{OPT} > n^{\frac{1}{w+1}}$ and $k \ge n^{\frac{w}{w+1}}$

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

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

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

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

The cost of our solution is upper bounded by the first term which is equal to $O\left(n(\frac{\log n}{k})^{\frac{1}{w}}\right) = O\left(\frac{n}{k}(\log n)^{\frac{1}{w}} k^{\frac{w-1}{w}}\right) \le O\left((\log n)^{\frac{1}{w}} k^{\frac{w-1}{w}}\right) L_{OPT}$. This gives an approximation guarantee of $O\left(k^{\frac{w-1}{w}}(\log n)^{\frac{1}{w}}\right) \le O\left((n^{\frac{w}{w+1}})^{\frac{w-1}{w}}(\log n)^{\frac{1}{w}}\right) = O\left(n^{\frac{w-1}{w+1}}(\log n)^{\frac{1}{w}}\right)$. ∎

Proof of Claim 2.6

We will prove the claim by showing a way to satisfy all requests while keeping the load on each channel at most 12.

Consider the solution, S, to 3DM containing q disjoint triples. We will dedicate $4q$ channels for the triples in S. For each triple in S, satisfy each triangle $T_i, 1 \le i \le 4$, on a separate channel. For the remaining $(|C| - q)$

triples, dedicate three channels per triple. For each triple in $C \setminus S$, satisfy each triangle $T_i, 5 \leq i \leq 7$, on a separate channel. Thus all of the $(3q + 9|C|)$ big requests are satisfied and $(48q + 144|C|)$ small requests are satisfied. The number of channels used to satisfy these requests is $4q + 3(|C| - q) = q + 3|C|$. The number of small requests that are not yet satisfied is equal to $(144|C| - 48q)$. This is same as the number of dummy requests and the number of unused channels. We pair each dummy request with a small request and satisfy each pair on a separate channel. The load on all channels is 12.

Proof of Claim 2.7
Consider a channel that would satisfy any of the following set of requests: (i) two dummy requests, (ii) one dummy request and one big request, (iii) one dummy request and two small requests. The load on the channel is 20, 14, and 13, respectively for the three cases. Thus our claim is true for all the three cases. For the remainder of the argument we will assume that none of the above set of requests are satisfied on the same channel. Since each dummy request is satisfied on a separate channel, the dummy requests use up $(144|C| - 48q)$ channels. At most one small request can be satisfied on each of the channels on which a dummy request is satisfied. Hence, $(3q + 9|C|)$ big requests and at least $(144|C| + 48q)$ small requests must be satisfied on the remaining $(q + 3|C|)$ channels. We will focus our attention on this case. Let us consider satisfying these requests as follows. Every big request is satisfied exactly once and each channel satisfies exactly 3 big requests and 48 small requests, i.e., each channel satisfies one triangle in a gadget. If this were the case then there exists q gadgets from which 4 triangles were satisfied. Since each big set is satisfied exactly once in this scenario, the big sets corresponding to the elements of the triples represented by the q gadgets are mutually disjoint. This means that the q triples are mutually disjoint and is a valid solution to 3DM, a contradiction. Now consider the case in which the above scenario is not true. This happens if any of the following happens. (i) a big set is satisfied more than once, (ii) a channel satisfies less than 3 big sets, (iii) a channel satisfies less than 48 small sets. Since the number of big requests is exactly three times the number of channels, in case (i) and (ii) there must exist a channel that satisfies four big requests and thus having a load of at least 16. For case (iii), since the number of small requests is 48 times the number of channels, there must exist a channel satisfying 49 small requests and thus incurring a load of 13. This proves our claim.

Proof of Theorem 3.1
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 user requests, 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, 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 $|\text{cut}(V_b^1, V_b^2)| \leq W$, 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 user 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 $L = \frac{n^3}{2} + \lceil \frac{W}{2} \rceil$ completes the reduction from GBP instance to CHA instance.

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

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 $\lceil \frac{W}{2} \rceil$ edges in E_c and $E_c^2 = E_c \setminus E_c^1$. For each edge in E_c^1, we include in CH_1, the endpoint of the edge in CH_2. For each edge in E_c^2, we include in CH_2, the endpoint of the edge in CH_1. Thus $L_2 \leq L_1 = \frac{n^3}{2} + \lceil \frac{W}{2} \rceil = L$.

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

Note 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 $\lceil \frac{W}{2} \rceil$ vertices that do not belong to any of the n^2-sized cliques on that channel and on the other channel there are $\lfloor \frac{W}{2} \rfloor$ such vertices. Each such vertex corresponds to an edge in $\text{cut}(V_b^1, V_b^2)$. Hence, $|\text{cut}(V_b^1, V_b^2)| \leq W$. ■

Proof of Lemma 3.2

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 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 ρ is the performance guarantee of DkS. Thus, after the first round, the number of uncovered edges $m_2 \leq m_1(1 - \frac{1}{\rho k})$. In general, after l rounds, $m_{l+1} \leq m_l(1 - \frac{1}{\rho k})$. 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

$$\begin{aligned}
m_{k'} &\leq m_{k'-1}(1 - 1/\rho k) \\
1 &\leq m(1 - 1/\rho k)^{k'-1} \\
\frac{1}{m} &\leq e^{-\frac{k'-1}{\rho k}} \\
e^{\frac{k'-1}{\rho k}} &\leq m \\
k' &\leq \rho k \ln m + 1
\end{aligned}$$

Using Lemma 2.1, we convert the (L_{OPT}, k') solution to an $(\lceil \frac{k'}{k} \rceil L_{OPT}, k) = (\lceil \frac{\rho k \ln m + 1}{k} \rceil L_{OPT}, k)$ 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 a $O(\rho \ln m) = O(\rho \ln n)$-approximate solution. ■

Proof of Theorem 3.3

Proof The best known approximation guarantee for DkS is $\rho \leq n^a, a < 1/3$ [10]. Combining this fact with Lemma 3.2 completes the proof. ■

Proof of Lemma 4.1

Proof For each iteration of the outer **while** loop we only need to calculate the number p of iterations of the inner **while** loop being executed, because each iteration of the inner loop removes exactly one vertex from V', hence 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 at most $\frac{wm_i'}{n_i'}$ edges will be removed from E', it is easy to verify that $m_{i+1}' \ge m' \prod_{j=0}^{i}(1-\frac{w}{n_j'})$, It is from the algorithm that 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 $m_p' \ge m' \prod_{j=0}^{p-1}(1-\frac{w}{n_j'})$ is meaningful. Thus we need $m' \prod_{j=0}^{p-1}(1-\frac{w}{n'-j}) \le k$. 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 $0 \le i \le w-1$. Hence we have $\prod_{i=0}^{w-1}(n'-p-i) \ge (\frac{n'-p}{w})^w$. Also, since $\prod_{i=0}^{w-1}(n'-i) \le (n')^w$, we get $n'-p+1 \le w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n'+1 \simeq w\left(\frac{k}{m'}\right)^{\frac{1}{w}} n'$. ■

Proof of Lemma 4.2

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

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

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

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

■

Proof of Lemma 4.3

Proof Suppose the optimal solution partitions the set of edges into t parts, and that part i contains m_i edges, $1 \le i \le t$. It is easy to verify that the number of vertices in part i is at least $(\frac{w}{e})m_i^{\frac{1}{w}}$. Therefore the total number of vertices is $S \ge \frac{w}{e}\sum_{i=1}^{t} m_i^{\frac{1}{w}}$. Obviously, $0 < m_i \le k$ and $\sum_{i=1}^{t} m_i = m$. Because function $f(x) = x^{\frac{1}{w}}$ is a concave function for $x \ge 0$ (since $f''(x) < 0$), hence we have $x^{\frac{1}{w}} > \frac{x}{k^{1-1/w}}$ for $0 < x \le k$. 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 $\max\{n, \frac{w/e}{k^{1-1/w}}m\}$. Also, because $\max\{a,b\} \ge a^p b^q$ 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 $q=1-\frac{1}{w}$, we obtain the claimed result. ■

Proof of Lemma 4.5

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 $ek^{1-1/w}$. ■

Proof of Theorem 4.6

Proof The approximation ratio is at most $r = \min\{ek^{1-1/w}, \frac{w^2}{w-1}\left(\frac{en}{wk^{1/w}}\right)^{1-1/w}\}$. If $k \le \left(\frac{w^2}{e(w-1)}\right)^{\frac{w^2}{w^2-1}}(\frac{en}{w})^{\frac{w}{w+1}}$, then $r \le ek^{1-1/w} \le w(\frac{e}{w-1})^{\frac{w}{w+1}} n^{\frac{w-1}{w+1}}$. Else, $r \le \frac{w^2}{w-1}\left(\frac{en}{wk^{1/w}}\right)^{1-1/w} \le$

$w(\frac{e}{w-1})^{\frac{w}{w+1}} n^{\frac{w-1}{w+1}}$. $f(x) = (\frac{e}{x-1})^{\frac{x}{x+1}}$ is a decreasing function when $x \geq 2$ (since $f'(x) < 0$). Hence, $(\frac{e}{w-1})^{\frac{w}{w+1}} \leq e^{2/3} < 2$. Thus the approximation ratio is at most $r < 2w \cdot n^{\frac{w-1}{w+1}}$. Furthermore, the running-time bound is easily verified. ■