

Social optimum in Social Groups with Give-and-Take criterion

Saurabh Aggarwal, Joy Kuri

Rahul Vaze

Department of Electronic Systems Engineering,
Indian Institute of Science, Bangalore, India,
Email: saggarwal@cedt.iisc.ernet.in, kuri@cedt.iisc.ernet.in

School of Technology and Computer Science,
Tata Institute of Fundamental Research, Mumbai, India
Email: vaze@tcs.tifr.res.in

Abstract—We consider a “Social Group” of networked nodes, seeking a “universe” of segments. Each node has subset of the universe, and access to an expensive resource for downloading data. Alternatively, nodes can also acquire the universe by exchanging segments among themselves, at low cost, using a local network interface. While local exchanges ensure minimum cost, “free riders” in the group can exploit the system. To prohibit free riding, we propose the “Give-and-Take” criterion, where exchange is allowed if each node has segments unavailable with the other. Under this criterion, we consider the problem of maximizing the aggregate cardinality of the nodes’ segment sets. First, we present a randomized algorithm, whose analysis yields a lower bound on the expected aggregate cardinality, as well as an approximation ratio of $1/4$ under some conditions. Four other algorithms are presented and analyzed. We identify conditions under which some of these algorithms are optimal

I. INTRODUCTION AND RELATED WORK

We consider a “Social Group” [1] of networked nodes, where each node is looking for a common set of data files/segments (henceforth refereed to as the “universe”) [2], [3]. The group has access to a common resource for downloading segments which is high cost, for example cost can be delay, license fee, transmission power etc. On the other hand, the nodes can exchange data among themselves at much lower cost possibly because of physical proximity, mutual self-help cooperation etc.. Initially, each node has a subset of the *universe*, and it is interested in acquiring all other pieces of the universe through the low cost local area network rather than the high cost common resource to reduce the overall download cost. Various popular applications based on this concept include web caching, content distribution networks, peer-to-peer networks, etc., with the common theme of leveraging low-cost data exchange instead of consuming expensive resources. More recently, device-to-device (D2D) communication in cellular wireless networks is another example of such a social group, where mobile phones can either directly talk to other mobile phones without going through the base station or help other mobiles in their communication.

Principle assumption in resource sharing is cooperation and willing participation of all nodes. In particular, a node should be willing to let other nodes obtain data from it. Typically, *selfish nodes* are reluctant to provide data/help to others; on the other hand, they are keen to grab as much as possible from

other nodes. In a peer-to-peer networking context, such selfish nodes are termed *free riders*. In a scenario with a large social group of nodes, it is very likely that nodes do not “know” one another, and hence do not trust each other. This lack of trust acts as a deterrent to sharing, to the detriment of the whole community.

For rational nodes, free riding is the dominant strategy. However, if every node engages in free riding, there will be a complete breakdown of sharing. To discourage free riding, various policies and methods have been proposed in the context to peer-to-peer networks [4]–[10]. Policy counterparts of these can be used to counter the free riding problem in social groups also. Often, however, these policies suggest and incentivize good behavior, but do not enforce it. Selfish users manage to find weaknesses that can be exploited. Various types of attacks (such as whitewashing, collusion, fake services, Sybil attack [9], [11], [12]) are possible in P2P networks [9]. Similar attacks can be launched by nodes in social groups as well.

To prohibit free riding, we propose the *Give-and-Take (GT) criterion* for segment exchange. Essentially, the idea is that a node A can download segments from another node B if and only if A offers at least one new segment to B . Thus, a notion of *fairness* is built into the *GT criterion*— A cannot get anything from B unless she offers B something in return. Each side has to contribute at least one segment that the other side does not have. In this paper, we study the special case of “Full Exchange,” in which nodes exhibit altruistic behaviour — A is willing to provide *all* segments it possesses even if she gets just *one* segment from B . Because of this, after an exchange, both A and B will possess the *union* of their individual segment sets before exchange.

Incorporating the *GT criterion* in our model, we study the problem of scheduling exchanges complying with the *GT criterion*, so that the aggregate number of data segments available with all the nodes can be maximized. The motivation is *reduced cost of downloading*—if the aggregate number of data segments acquired through low cost local exchanges is maximized, then the total number of segments to be downloaded using the high-cost resource is least.

Given data segment sets at each node, there may be several pairs of nodes that satisfy the *GT criterion* and different

choices for data exchange lead to different final aggregate cardinalities. Identifying the optimal schedule of data exchange that maximizes the aggregate cardinality of data segments available at all nodes is a combinatorial problem with exponential complexity (\mathcal{NP} -hard). The complexity of the problem can be accessed from a simple example illustrated in Fig. 1, where there are only 4 nodes and the universe size is 5, and there are exponentially many data exchange schedules. Finding the one with optimal final aggregate cardinality is a challenging task.

This paper makes the following contributions.

- We propose the novel *GT criterion*, which prohibits free riding in social groups. This, we believe, is a new concept that will help understand the fundamental principles of data sharing in local networks under fair exchange models.
- We propose a Randomized algorithm, that works in phases, where in each phase it randomly picks a pair of nodes for exchange. If the chosen pair satisfies the *GT criterion*, then the exchange happens, otherwise the two nodes are kept aside and a new pair is chosen randomly again. The phase ends when all pairs are exhausted. We show that the randomized algorithm is optimal for maximizing the aggregate cardinality when the number of nodes is very large. We also show that the aggregate cardinality obtained by the randomized algorithm is at least $1/4$ times of the optimal aggregate cardinality under some conditions, thus providing a 4 approximation.
- We propose a Greedy-Links algorithm that, at each step, pairs those nodes so that the number of possible exchanges in the next step is maximized. We show that this algorithm is optimal for small number of nodes, e.g. 4.
- We propose shifted round-robin algorithm called the Polygon algorithm, that in each phase exchanges segments between neighboring nodes (certain predefined order of nodes), and then for the next phase, repeats the process after circularly left shifting the order of nodes. The polygon algorithm is shown to be optimal when each node has at least one unique segment.
- We propose two more algorithms, the first (called the Greedy-Incremental Algorithm) exchanges nodes that maximize the aggregate cardinality in next phase, while the second (called the Rarest First Algorithm) chooses that pair for exchange such that subset of segments with the minimum number of nodes is maximized.
- By means of extensive simulation results, we show that the Greedy links algorithm performs the best, since it tries to maximize the potential for nodes to exchange their pieces. Rarest first algorithm comes a close second to the Greedy links algorithm, while the Randomized algorithm on an average outperforms Greedy incremental algorithm and Polygon algorithm.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a set of nodes, $\mathcal{M} = \{1, 2, \dots, m\}$ and a universe $\mathcal{N} = \{1, 2, \dots, n\}$ of segments. Each node $i \in \mathcal{M}$ has an initial collection of segments $O_i \subset \mathcal{N}$.

Give-and-Take (GT) criterion: Two nodes $i, j \in \mathcal{M}$ with segment sets X_i and X_j , respectively, can exchange segments if and only if $X_i \cap X_j^c \neq \emptyset$ and $X_i^c \cap X_j \neq \emptyset$, i.e. node j has at least one segment which is unavailable with node i and vice versa. After exchange, both nodes have the segment set $X_i \cup X_j$.

We consider the set of nodes \mathcal{M} as the vertices in an undirected graph G , where an edge or *link* exists between two vertices $i, j \in \mathcal{M}$ if and only if they satisfy the *GT criterion*. We denote the link between nodes i and j by the unordered 2-tuple (i, j) . By activating/pairing an edge/link we mean that the two nodes on that edge exchange their segments.

Given an initial collection of segment sets O_i , a schedule S is a repeated activation of links (exchange of nodes) in graph G , in a given order. Note that since one link activation changes the graph G , we denote the dynamic graph G as $G_S(r)$, where subscript S stands for the schedule of node exchanges/link activations over graph G and r stands for the r^{th} step of schedule S . Thus, $G_S(r)$ is a graph with vertex set \mathcal{M} , and where an edge between two vertices $i, j \in \mathcal{M}$ exists if they satisfy the *GT criterion* at the r^{th} iteration of schedule S .

For example, in $G_B(3)$ of Fig. 1(b), i.e. in 3^{rd} step of schedule B , links exist between node 1 and 4, node 3 and 4 and node 2 and 4. If link $(2, 4)$ is activated, then node 2 gets segment 5 from node 4, and node 4 gets segments 2, 3, 4 from node 2. Thus, for $G_B(4)$ the graph is completely disconnected since no two nodes satisfy the *GT criterion*.

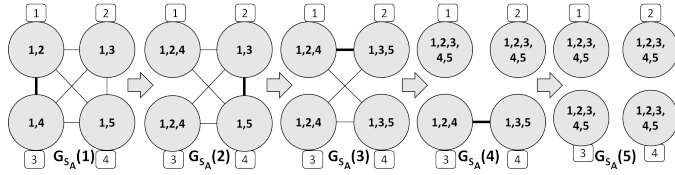
We represent schedule S as a row vector of links in the order of activation and $|S|$ denotes the number of link activations in the schedule S . We denote the set of available links after r activations of a schedule S by $\mathcal{L}(S, r)$.

Let $O_i(S, r)$ denote the set of segments with node $i \in \mathcal{M}$ after the first r activations of schedule S . In this context, $O_i(S, 0) = O_i$ denotes the initial collection of segments with node $i \in \mathcal{M}$. Similarly, $\mathcal{L}([\cdot], 0)$ denotes the set of links that exist between the initial segment sets.

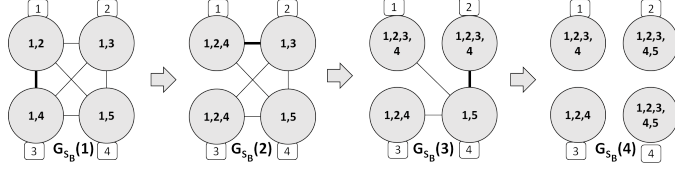
A *Maximal Schedule* is a schedule which cannot be extended by any other link activations, i.e. at the end of the maximal schedule the *GT criterion* is not satisfied between any pair of nodes. Let \mathcal{X} denote the set of all *Maximal Schedules*.

For any schedule S , let $[S, (i, j)]$ denotes the schedule which follows all link activations in S , followed by activation of the link between node i and j such that $(i, j) \in \mathcal{L}(S, |S|)$.

In Fig. 1(b), schedule S_B is being followed for link activation. We activate link $(1, 3)$ and both the nodes 1 and 3 will have segment sets $O_1(S_B, 1) = O_3(S_B, 1) = O_1 \cup O_3$. We can activate either of links in $\mathcal{L}(S_B, 1) = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4)\}$. Now, activating link $(1, 2)$ followed by link $(2, 4)$, leads us to graph $G_{S_B}(4)$ where no links exist between any of the nodes. Hence, S_B is a maximal schedule which will be represented as $S_B = [(1, 3), (1, 2), (2, 4)]$.



(a) Evolution of nodes' segment set under schedule S_A ; Aggregate cardinality under schedule S_A is 20



(b) Evolution of nodes' segment set under schedule S_B ; Aggregate cardinality under schedule S_B is 17

Fig. 1. Evolution of nodes' segment set under two different schedules. Nodes are numbered in the rounded squares. Node set is $\mathcal{M} = \{1, 2, 3, 4\}$ and Universe is $\mathcal{N} = \{1, 2, 3, 4, 5\}$.

Assumptions

- A1: We assume that the cost of exchange between nodes is zero or negligible as compared to the cost of downloading the segment from outside the set \mathcal{M} similar to [3].
A2: $O_i \neq \emptyset$, and $O_i \neq \mathcal{N}$, $\forall i \in \mathcal{M}$.

Computing the Social optimum

The aggregate cardinality on completion of a maximal schedule S is denoted by $\alpha(S) = \sum_{i \in \mathcal{M}} |O_i(S, |S|)|$. Our objective is to find a schedule of link activations S^* , such that the total number of segments available with the nodes, i.e., the aggregate cardinality on completion of S^* , is *maximized*. Formally, we want to find

$$\begin{aligned} \max \quad & \alpha(S). \\ \text{Subject to: } & S \in \mathcal{X} \end{aligned} \quad (1)$$

Note that a trivial upper bound to $\alpha(S) \leq mn$ for even- m and $\alpha(S) \leq mn - 1$ for odd- m , that will be useful later in the paper.

To understand the implications of solving (1), note that to begin with, the number of “missing” pieces in the network is $(mn - \sum_{i \in \mathcal{M}} |O_i|)$. After the completion of optimal schedule S^* , at most $(mn - \alpha(S^*))$ segments will need to be fetched using the expensive resource, thus solving (1) minimizes the social cost.¹

The optimal schedule (solution) to (1) is the social optimum, however, solving (1) has exponential complexity, that makes it computationally infeasible. Even for simple examples, as in Fig. 1, there are many possible schedules, and different schedules lead to different aggregate cardinalities, e.g. with schedule S_A every node can get the universe while with schedule S_B only two nodes can get the universe. In the rest of

the paper, we propose several polynomial-time algorithms to solve (1) and bound their performance analytically and through extensive simulations.

Next, we derive an important property of the the *GT criterion* based segment exchanges in Lemma 1.

Lemma 1. *For any maximal schedule S and $O_i \subsetneq \bigcup_{j \in \mathcal{M}} O_j \forall i, j \in \mathcal{M}$, at least two nodes will have $\bigcup_{i \in \mathcal{M}} O_i$ towards the end of schedule S .*

Proof: We reorder the nodes in ascending order of cardinalities at the end of S and index them as i_1, \dots, i_m : $|O_{i_1}(S, |S|)| \leq |O_{i_2}(S, |S|)| \leq \dots \leq |O_{i_m}(S, |S|)| \leq |O_{i_m}(S, |S|)|$. By the *GT criterion*, if a link does not exist between the nodes i and j , then $O_i \subseteq O_j$ or $O_j \subseteq O_i$. Therefore,

$$O_{i_1}(S, |S|) \subseteq O_{i_2}(S, |S|) \subseteq \dots \subseteq O_{i_{m-1}}(S, |S|) \subseteq O_{i_m}(S, |S|).$$

Since the cardinalities of nodes i_{m-1} and i_m are the largest and second largest, and each link activation produces two nodes with same subset of segments, therefore $O_{i_{m-1}}(S, |S|) = O_{i_m}(S, |S|)$. Since there is no link between i_{m-1}, i_m and any other node, the segment sets of nodes i_{m-1} and i_m are supersets of segments available with all other nodes, therefore

$$O_{i_{m-1}}(S, |S|) = O_{i_m}(S, |S|) = \bigcup_{i \in \mathcal{M}} O_i(S, |S|) = \bigcup_{i \in \mathcal{M}} O_i.$$

Hence, at least two nodes will have $\bigcup_{i \in \mathcal{M}} O_i$ at the end of any maximal schedule S . ■

Corollary 1. *For any $S \in \mathcal{X}$,*

$$\alpha(S) \geq 2 \left| \bigcup_{i \in \mathcal{M}} O_i \right| + \sum_{i \in \mathcal{M} \setminus \{i_{m-1}, i_m\}} |O_i|,$$

where i_{m-1} and i_m are two nodes having the maximum cardinalities at the end of schedule S .

Hence, Lemma 1, shows that for any maximal schedule at least two nodes will get the whole *realized* universe, where by realized we mean the union of segments available with all nodes. In Lemma 2, we derive the probability that the realized universe is equal to the universe, when each node gets k -segments uniformly randomly drawn from universe \mathcal{N} .

Lemma 2. *Let each of the m nodes select k segments uniformly at random from a universe \mathcal{N} . Then*

$$\mathbb{P} \left(\bigcup_{j=1}^m O_j = \mathcal{N} \right) = \begin{cases} p^{m,n,k} & \text{if } mk \geq n, \\ 0 & \text{otherwise.} \end{cases}$$

Proof: Evidently, if $mk < n$, then $\mathbb{P} \left(\bigcup_{j \in \mathcal{M}} O_j = \mathcal{N} \right) = 0$. The interesting case is when $mk \geq n$, for which $\mathbb{P} \left(\bigcup_{j \in \mathcal{M}} O_j = \mathcal{N} \right) = \frac{\text{No of favourable cases}}{\text{Total number of possibilities}}$. The number of k element subsets of $\mathcal{N} = \binom{n}{k}$. Total number of possibilities = $\binom{n}{k}^m$.

¹There is a possibility for nodes to download few segments over the high cost network after S^* and again use the low cost network for exchanges among themselves. This process might be repeated recursively.

$p^{m,n,k}$ is defined here for the sake of convenience; it is used in Lemma 2.

$$p^{m,n,k} = \binom{n}{k}^{-m} \sum_{\substack{\sum_{j=1}^{m-1} k_j = mk - n \\ (k_1, k_2 \dots k_{m-1})}} \prod_{j=1}^m \binom{(j-1)k - \sum_{i=1}^{j-2} k_i}{k_{j-1}} \binom{n - ((j-1)k - \sum_{i=1}^{j-2} k_i)}{k - k_{j-1}}. \quad (2)$$

Calculating the number of favourable cases: If $mk > n$, some elements must appear in the segment sets of several nodes. The total number of repeated elements is given by $(mk - n)$. Let us assume $|O_2 \cap O_1| = k_1$, $|O_3 \cap (O_1 \cup O_2)| = k_2$, ..., $|O_m \cap (\bigcup_{i=1}^{m-1} O_i)| = k_{m-1}$. Therefore, $\sum_{j=1}^{m-1} k_j = mk - n$. The number of ways of choosing $O_1 = \binom{n}{k}$. O_2 has k_1 elements in common with O_1 , so k_1 elements should be chosen from O_1 (consisting of k elements) and the remaining $(k - k_1)$ elements should be chosen from the remaining $(n - k)$ elements. This gives the number of ways of choosing O_2 as $\binom{k}{k_1} \binom{n-k}{k-k_1}$.

Similarly, O_3 has k_2 elements in common with O_1 and O_2 , so k_2 elements should be chosen from $O_1 \cup O_2$ (consisting of $(2k - k_1)$ elements) and the remaining $(k - k_2)$ elements from $(n - (2k - k_1))$ elements. Thus, the number of ways of choosing O_3 is $\binom{2k-k_1}{k_2} \binom{n-(2k-k_1)}{k-k_2}$.

In the same way, the number of ways of choosing O_j is

$$\binom{(j-1)k - \sum_{i=1}^{j-2} k_i}{k_{j-1}} \binom{n - ((j-1)k - \sum_{i=1}^{j-2} k_i)}{k - k_{j-1}}.$$

Hence, the total number of ways of choosing $O_1, O_2 \dots O_j \dots O_m$ for a given $(k_1, k_2 \dots k_{j-1} \dots k_{m-1})$ is

$$\prod_{j=1}^m \binom{(j-1)k - \sum_{i=1}^{j-2} k_i}{k_{j-1}} \binom{n - ((j-1)k - \sum_{i=1}^{j-2} k_i)}{k - k_{j-1}}.$$

To get the total number of favorable cases, we need to sum over all possible values of $(k_1, k_2 \dots k_{j-1} \dots k_{m-1})$ for which $\sum_{j=1}^{m-1} k_j = (mk - n)$. Therefore,

$$\mathbb{P} \left(\bigcup_{j=1}^m O_j = \mathcal{N} \right) = \begin{cases} p^{m,n,k} & \text{if } mk \geq n, \\ 0 & \text{Otherwise.} \end{cases}$$

Corollary 2. *The probability that at least one node will have the universe \mathcal{N} at the end of any schedule $S \in \mathcal{X}$ is $p^{m,n,k}$.*

Proof: This follows from Lemma 1 and Lemma 2. ■

III. ALGORITHMS

A. Randomized Algorithm

We first present a randomized algorithm to solve (1). The randomized algorithm works in phases. In each phase, two nodes are uniformly randomly chosen for exchange. If they have an edge between them in the corresponding graph, i.e. satisfy the *GT criterion*, then they exchange their segments, otherwise they are kept aside without any exchange. Once a

pair is chosen in a phase, with or without actual exchange, it takes no further part in that phase. The phase ends when choice of all pairs is exhausted, and then the new phase starts all over again, forgetting what happened in earlier phases. The algorithm terminates when there are no more links available in the graph after the end of any phase.

Algorithm 1 Randomized Algorithm

```

1:  $r := 0, p := 1$  and  $S_{rand} := [\cdot]$ 
2: while  $\mathcal{L}(S_{rand}, r) \neq \emptyset$  do
3:    $\mathcal{M}_{pick} := \mathcal{M}$ 
4:   while  $\mathcal{M}_{pick} \neq \emptyset$  do                                % phase p begins
5:     Select nodes  $i$  and  $j$  at random from  $\mathcal{M}_{pick}$ 
6:     if  $(i, j) \in \mathcal{L}(S_{rand}, r)$  then
7:       Activate link between nodes  $i$  and  $j$ 
8:        $S_{rand} \leftarrow [S_{rand}, (i, j)]$ 
9:        $r \leftarrow r + 1$ 
10:    end if
11:     $\mathcal{M}_{pick} \leftarrow \mathcal{M}_{pick} \setminus \{i, j\}$ 
12:  end while                                                % phase p ends
13:   $p \leftarrow p + 1$ 
14: end while

```

Theorem 1. *If initial segment sets (O_i 's) are chosen uniformly at random from \mathcal{N} , with $|O_i| = k$, $1 \leq k \leq n - 1$, $\forall i \in \mathcal{M}$,*

- 1) *Then the randomized algorithm is asymptotically optimal in m , i.e. for large m , $E(\alpha(S_{rand})) = nm$, where nm is the upper bound on (1).*
- 2) *The randomized algorithm is a 4 approximation to (1), if $k \in \left[\min \left(\frac{n}{\log_2 m}, \frac{n}{4} \right), n - 1 \right]$.*

Proof: We define a *phase* as a round of link activations in steps 4-12 of Algorithm 1. For notational convenience, let $s_p^i = |O_{(i,p)}|$, where $O_{(i,p)}$ denotes the segment set with node i at the beginning of phase p . At the start of phase 1, each node has $s_1^i = k$ segments, that are uniformly randomly picked from the universe. Since the algorithm is randomized, we focus on a particular node i for analysis, and let in phase p , assume that node i and j_p are paired, where j_p is uniformly randomly chosen among the other $m - 1$ nodes.

The segment set at i^{th} node at the beginning of phase p is $O_{(i,p)} = \{e_1, e_2, \dots, e_{|O_{(i,p)}|}\}$. For every phase p , and any segment $e \in O_{(j_p,p)}$, we define the random variables $X_e(j_p, p)$:

$$X_e(j_p, p) = \begin{cases} 1 & e \notin O_{(i,p)}, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{M}_0 = \{i\}$ and \mathcal{M}_p denote the subset of nodes that have been influenced by node i by the end of phase $p - 1$. The set of influenced nodes is inductively defined to be all nodes that have had any pairings with node i until phase $p - 1$. For example, if node i and node 2 are paired in phase 1, and nodes 2 and 4 are paired in phase 2, then $\mathcal{M}_3 = \{i, 2, 4\}$. Note that $|\mathcal{M}_p| \leq 2^{p-1}$ since in any phase only two nodes are paired with each other and therefore influence of i grows only two-folds in each phase.

For phase 1, node i 's segment set cardinality increase *after exchange*, $(s_2^i - s_1^i)$, will equal the number of segments that are available with node j_1 but not with node i , i.e.

$$\begin{aligned} E(s_2^i - s_1^i | j_1 \notin \mathcal{M}_0) &= E\left(\sum_{e \in O_{(j_1, 1)}} X_e \mid j_1 \notin \mathcal{M}_0\right), \\ &\stackrel{(a)}{=} |O_{(j_1, 1)}| \mathbb{P}(X_e = 1), \\ &\stackrel{(b)}{=} k \left(1 - \frac{k}{n}\right). \end{aligned}$$

where (a) follows from linearity of expectation and since $\mathcal{M}_0 = \{i\}$ implying that $\{j_1 \notin \mathcal{M}_0\}$ is always true, and (b) follows from $\mathbb{P}(X_e = 1) = (1 - \frac{k}{n})$ for every e . Moving on to phase 2, similarly, $E(s_3^i - s_2^i | j_2 \notin \mathcal{M}_1, |O_{(i, 2)}| = c_2)$

$$\begin{aligned} &= E\left(\sum_{e \in O_{(j_2, 2)}} X_e \mid j_2 \notin \mathcal{M}_1, |O_{(i, 2)}| = c_2\right), \\ &= E(s_2^{j_2}) \left(1 - \frac{c_2}{n}\right). \end{aligned}$$

Then $E(s_3^i - s_2^i | j_2 \notin \mathcal{M}_1)$,

$$\begin{aligned} &= \sum_{c_2} E(s_3^i - s_2^i | j_2 \notin \mathcal{M}_1, |O_{(i, 2)}| = c_2) \mathbb{P}(|O_{(i, 2)}| = c_2) \\ &= E(s_2^{j_2}) \left(1 - \frac{E(s_2^i)}{n}\right), \end{aligned} \quad (3)$$

where (3) follows from linearity of expectation, and more importantly from the fact that for $j_2 \notin \mathcal{M}_1$, $O_{(i, 2)}$ and $O_{(j_2, 2)}$ are independent, since initially all segments at all nodes were drawn uniformly randomly and node j_2 has had no exchange with any node that had an exchange with node i , consequently, s_2^i and $s_2^{j_2}$ are independent. Also note that since the algorithm is randomized, $s_2^{j_2}$ and s_2^i are identically distributed, hence $E(s_2^{j_2}) = E(s_2^i)$, and from (3)

$$E(s_3^i - s_2^i | j_2 \notin \mathcal{M}_1) = E(s_2^i) \left(1 - \frac{E(s_2^i)}{n}\right). \quad (4)$$

Generalizing (4) for any phase p ,

$$E(s_{p+1}^i - s_p^i | j_p \notin \mathcal{M}_{p-1}) = E(s_p^i) \left(1 - \frac{E(s_p^i)}{n}\right), \quad (5)$$

$$\text{Since } |\mathcal{M}_p| \leq 2^{p-1}, \quad \mathbb{P}(j_p \notin \mathcal{M}_{p-1}) \geq$$

$$\frac{\max(m - 2^{p-1}, 0)}{m - 1}. \text{ Therefore, from (4)}$$

$$\begin{aligned} E(s_{p+1}^i - s_p^i) &\geq E(s_{p+1}^i - s_p^i | j_p \notin \mathcal{M}_{p-1}) \mathbb{P}(j_p \notin \mathcal{M}_{p-1}), \\ &\geq E(s_p^i) \left(1 - \frac{E(s_p^i)}{n}\right) \left(\frac{\max(m - 2^{p-1}, 0)}{m - 1}\right). \end{aligned}$$

Therefore, the expected cardinality of node i at the end of phase p is given by $E(s_{p+1}^i)$

$$\geq E(s_p^i) + E(s_p^i) \left(1 - \frac{E(s_p^i)}{n}\right) \left(\frac{\max(m - 2^{p-1}, 0)}{m - 1}\right) \quad (6)$$

$\forall p \geq 2$. Expected aggregate cardinality at the end of phase p is given by $mE(s_{p+1}^i)$.

For any m , we can find a natural number a such that $2^{a-1} < m \leq 2^a$. Then, for iterations $p = 2$ to $a - 1$, $\left(\frac{\max(m - 2^{p-1}, 0)}{m - 1}\right) > \frac{1}{2}$. Thus, $E(s_{p+1}^i)$

$$\begin{aligned} &\geq E(s_p^i) + E(s_p^i) \left(1 - \frac{E(s_p^i)}{n}\right) \left(\frac{\max(m - 2^{p-1}, 0)}{m - 1}\right), \\ &\geq E(s_p^i) + \frac{E(s_p^i)}{2} \left(1 - \frac{E(s_p^i)}{n}\right), \quad p = 1, \dots, a - 1. \end{aligned} \quad (7)$$

Noting that $a - 1 = \lfloor \log_2 m \rfloor$, we have $E(s_{\lfloor \log_2 m \rfloor + 1}^i)$

$$\geq E(s_{\lfloor \log_2 m \rfloor}^i) + \frac{E(s_{\lfloor \log_2 m \rfloor}^i)}{2} \left(1 - \frac{E(s_{\lfloor \log_2 m \rfloor}^i)}{n}\right)$$

Now the sequence $E(s_{\lfloor \log_2 m \rfloor}^i)$ is monotonically nondecreasing and bounded above by n ; hence, the sequence converges to n . Adding the cardinality of all nodes, we get $\alpha(S_{rand}) \geq nm$. Thus, asymptotically in m , i.e. for large number of nodes, the randomized algorithm achieves the optimal solution to (1), since nm is an upper bound on the aggregate cardinality of (1). This proves part (1) of the Theorem.

For part (2), we note that function $\frac{x}{2}(1 - \frac{x}{n})$ is a concave function for $x = k, k + 1, \dots, n$, with maximum at $x = n/2$, and increasing from $x = k, \dots, n/2$. Hence from (7), either $E(s_{p+1}^i) > n/4$ or $E(s_p^i) \left(1 - \frac{E(s_p^i)}{n}\right) > k$ for phases $p = 1, \dots, a - 1$. In the former case, adding the cardinality across m nodes, we have $\alpha(S_{rand}) \geq \frac{mn}{4}$. In the latter case, $E(s_a^i) > \frac{ka}{2}(1 - \frac{k}{n})$ by adding for a phases, and $\alpha(S_{rand}) \geq \frac{mka}{2}(1 - \frac{k}{n})$. We only consider $k < \frac{n}{2}$, since otherwise we already have the 4-approximation, for which case $(1 - \frac{k}{n}) > \frac{1}{2}$. Since $a = \log_2 m$, if $k \log_2 m > n$, then $\frac{mka}{2}(1 - \frac{k}{n}) = \frac{mk \log_2 m}{2}(1 - \frac{k}{n}) > \frac{mn}{4}$ as required. The proof for $k > \frac{n}{4}$ is immediate. ■

B. Greedy-Links Algorithm

The Greedy-Links algorithm works by examining the impact of a choice (i, j) on the *number of links in the graph after i and j are paired*. At each decision point, the algorithm chooses that pair for which the number of links in the resulting graph is

maximized. Ties are broken arbitrarily. The idea is motivated by the observation that if the number of links in the graph is large, then it is likely that the maximal schedule will be longer, and closer to an optimal one. The resulting maximal schedule is denoted by S_{Glink} . Algorithm 2 shows the pseudo-code. Time complexity for greedy-links algorithm is $\theta(m^6)$.

Algorithm 2 Greedy-Links Algorithm

```

1:  $r := 0$  and  $S_{Glink} := [\cdot]$ 
2: while  $\mathcal{L}(S_{Glink}, r) \neq \emptyset$  do
3:    $w_{(i,j)} := |\mathcal{L}([S_{Glink}, (i,j)], r+1)| \quad \forall (i,j) \in \mathcal{L}(S_{Glink}, r)$ 
4:    $(i,j) = \arg \max_{(i,j) \in \mathcal{L}(S_{Glink}, r)} w_{(i,j)} \quad \% \text{ Ties are broken arbitrarily}$ 
5:   Activate link between node  $i$  and  $j$ 
6:    $S_{Glink} \leftarrow [S_{Glink}, (i,j)]$ 
7:    $r \leftarrow r + 1$ 
8: end while

```

We next show that Greedy Links algorithm can be shown to be optimal under certain cases. Even though the results stated next are for very restrictive scenarios, however, we conjecture that Greedy Links algorithm is close to optimal for all cases.

Proposition 1. *For $m = 4$ nodes, if each node has a segment set O_i consisting of exactly k segments. Then, the Greedy Links algorithm is optimal.*

Proof: The proof follows by examining all possible equivalent graphs with 4 nodes having identical number of k segments, and then checking for optimality by brute force. We skip the details due to space constraints. ■

Proposition 2. *For $m = 4$ nodes such that optimal aggregate cardinality is $4n$, i.e. all nodes can get all segments of the universe with an optimal algorithm. Then the aggregate cardinality achieved by the Greedy links algorithm is also $4n$, i.e., if $\alpha(S^*) = 4n$ then $\alpha(S_{Glink}) = 4n$.*

Proof: Without loss of generality, we only consider the case when none of the 4 nodes have all the segments of the universe to begin with.

Claim 1: For the optimal schedule S^* , $|\mathcal{L}(S^*, \ell)| \geq 4$, where for the first time after ℓ exchanges any two nodes can exchange to get all the segments of the universe.

Consider the optimal schedule S^* . From the hypothesis of the claim, the length of S^* is $\ell + 2$, $|S^*| = \ell + 2$, since for the optimal schedule all nodes can get all segments of the universe and that can happen in two exchanges after ℓ . For S^* , after the $\ell + 1^{th}$ exchange, the number of links available is $|\mathcal{L}(S^*, \ell + 1)| = 1$, since two nodes have necessarily obtained all segments of the universe and cannot have links to any other nodes, and the only link present is between the remaining two nodes. Without any loss of generality, let us assume that $(1, 2)$ link is still active after $\ell + 1$ exchanges, $\mathcal{L}(S^*, \ell + 1) = \{1, 2\}$.

Hence, link $(3, 4)$ was activated in the $\ell + 1^{th}$ exchange in S^* , such that $O_3(S^*, \ell) \cup O_4(S^*, \ell) = \mathcal{N}$. Also, $O_1(S^*, \ell) \cup$

$O_2(S^*, \ell) = \mathcal{N}$ as S^* ensures that all nodes get all the segments of the universe. Also note that $(\ell + 1)^{th}$ activation yields first pair of nodes having universe.

Therefore, after ℓ exchanges, node 1 has an edge to node 2, and an edge to at least either of nodes 3 or 4, since $O_1(S^*, \ell) \subset \mathcal{N}$, $O_3(S^*, \ell) \subset \mathcal{N}$, $O_4(S^*, \ell) \subset \mathcal{N}$ and $O_3(S^*, \ell) \cup O_4(S^*, \ell) = \mathcal{N}$. Similarly, node 2 will have an edge either to node 3 or 4. Therefore, $|\mathcal{L}(S^*, \ell)| \geq 4$.

Claim 2: For Greedy links algorithm schedule S_{Glink} , if all nodes do not get all the segments of the universe, then $|\mathcal{L}(S_{Glink}, \ell)| \leq 3$, where for the first time after ℓ exchanges S_{Glink} algorithm activates two nodes such that they get all the segments of the universe.

Subclaim 2.1: Under the hypothesis of Claim 2, the length of the S_{Glink} schedule is $\ell + 1$, i.e. $|S_{Glink}| = \ell + 1$.

The proof of Subclaim 2.1 is immediate since otherwise S_{Glink} algorithm (by definition) would not have activated two nodes such that they get all the segments of the universe, and from the fact all nodes do not get all the segments of the universe.

At the $\ell + 1^{th}$ exchange, let nodes 1 and 2 exchange their pieces. Since $|S_{Glink}| = \ell + 1$, this will be the last possible exchange. Therefore, after the ℓ^{th} exchange, if node 3 has edge to both node 1 and 2, then node 4 cannot have an edge to either node 1 or 2, since otherwise the S_{Glink} algorithm will not be at its last possible exchange. Hence, after the ℓ^{th} exchange, the set of edges are either $(1, 2), (1, 3), (1, 4)$ or $(1, 2), (2, 3), (2, 4)$, proving Claim 2.

The proof of the Theorem follows by a contrapositive argument to Claim 2, i.e. if we show that $|\mathcal{L}(S_{Glink}, \ell)| \geq 4$, where for the first time after ℓ exchanges S_{Glink} algorithm activates two nodes such that they get all the segments of the universe, then with S_{Glink} algorithm all nodes get all the segments of the universe.

The claim that $|\mathcal{L}(S_{Glink}, \ell)| \geq 4$, follows from the fact the the optimal algorithm does so (*Claim 1*) and the fact that Greedy links S_{Glink} algorithm keeps maximum number of links alive, and using brute force we can verify that S_{Glink} will also have at least 4 edges when for the first time a pair of nodes exchange to get all the segments of the universe. ■

C. Polygon Algorithm

The polygon algorithm is defined for a set of nodes $\mathcal{M}^U \subseteq \mathcal{M}$, such that each node $i \in \mathcal{M}^U$ has at least one unique segment with itself, i.e., $O_i \setminus \bigcup_{j \in \mathcal{M}^U \setminus \{i\}} O_j \neq \emptyset \quad \forall i \in \mathcal{M}^U$.

Let us consider a row vector P which is a permutation of nodes in \mathcal{M}^U . We define the *left circular shift* of the row vector P : $P = [i_1, i_2, i_3, \dots, i_{|\mathcal{M}^U|}]$, $LeftCircShift(P) = [i_2, i_3, i_4, \dots, i_{|\mathcal{M}^U|}, i_1]$.

The proposed polygon algorithm in each round, picks two neighboring nodes in P starting from left, and activates the link between them and repeats this process until there are no pairs left. In the next round, the above procedure is repeated with $LeftCircShift(P)$. Note that this algorithm critically depends on the choice of starting permutation P . The starting permutation used in round 1 is the one that has the *smallest*

number of unique segments placed at the right most location. For the case when, $\mathcal{M}^U \subset \mathcal{M}$, for maximizing the aggregate cardinality with the polygon algorithm, we need to find the subset \mathcal{M}^U with the largest cardinality. This, however, is a combinatorial problem and we use a greedy algorithm for this purpose that has linear complexity. The complexity of polygon algorithm is $\theta(m^3)$. Algorithm 3a provides the pseudo-code.

Algorithm 3a Polygon algorithm

```

1: Find  $\mathcal{M}^U$ , Algorithm 3b,
2: while  $|\mathcal{M}^U| \geq 2$  do
3:   Let  $M$  be any permutation of  $\mathcal{M}^U$  % the GT
   criterion is satisfied for any pair of nodes in  $\mathcal{M}^U$ 
4:    $l := 0, r := 0, S_{poly} := [\cdot]$ 
5:   while  $l \leq \left\lfloor \frac{|\mathcal{M}^U| - 1}{2} \right\rfloor$  do
6:     for  $p = 2 : 2 : |\mathcal{M}^U| - 1$  do %  $p$  is a even number
7:        $i \leftarrow (p - 1)^{th}$  element in  $M, j \leftarrow p^{th}$  element
       in  $M$ 
8:       Activating link between  $i$  and  $j$ 
9:        $S_{poly} \leftarrow [S_{poly}, (i, j)], r \leftarrow r + 1$ 
10:    end for
11:     $M \leftarrow LeftCircShift(M)$ 
12:     $l \leftarrow l + 1$ 
13:  end while
14:  Find  $\mathcal{M}^U$ 
15: end while

```

Algorithm 3b Find \mathcal{M}^U

```

1:  $\mathcal{M}_{pick} := \mathcal{M}$  and  $\mathcal{M}^U := \emptyset$ 
2: while  $\mathcal{M}_{pick} \neq \emptyset$  do
3:   Choose a element  $i$  from  $\mathcal{M}_{pick}$ 
4:   if  $O_i \setminus \cup_{j \in \mathcal{M}^U} O_j \neq \emptyset$  and  $\min_{j \in \mathcal{M}^U} |O_j \setminus O_i| \neq 0$  then
5:      $\mathcal{M}^U \leftarrow \mathcal{M}^U \cup \{i\}$ 
6:   end if
7:    $\mathcal{M}_{pick} \leftarrow \mathcal{M}_{pick} \setminus \{i\}$ 
8: end while

```

For the special case of $\mathcal{M}^U = \mathcal{M}$, i.e. each node has an unique segment, we show that the polygon algorithm is optimal for solving (1).

Proposition 3. *If $\mathcal{M}^U = \mathcal{M}$, then the polygon algorithm is optimal for solving (1).*

Proof: For lack of space, the proof is best described with a figure. In Fig. 2, we consider $m = 7$ nodes and each node i contains at least one unique segment denoted by i . Each node can contain more segments, however, it is sufficient to just consider the unique segments. The figure is self-explanatory given the polygon algorithm, and in this odd- m case achieves aggregate cardinality $mn - 1$, where except one node every other node gets the universe, while for even- m case every node gets the universe and achieves the aggregate cardinality of mn , that is maximum possible for (1). If all segments of

the universe are not seen by at least one node, then replace n by $|\cup_{i \in \mathcal{M}} O_i|$, and the algorithm is still optimal. ■

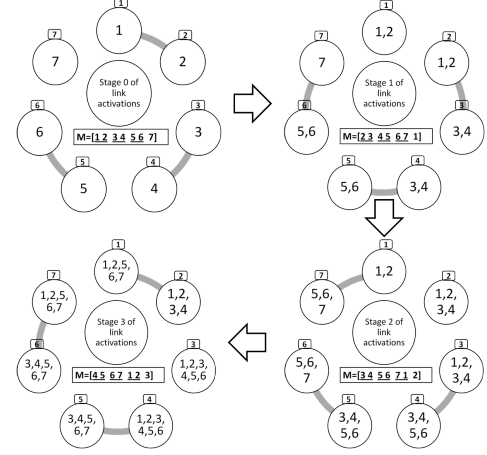


Fig. 2. Demonstration of Polygon algorithm.

D. Greedy-Incremental Algorithm

The Greedy-Incremental algorithm works at each decision epoch, chooses that pair of nodes for exchange that has maximum sum of newly added segments to each node after the exchange. To each link $(i, j) \in \mathcal{L}(S_{Ginc}, r)$, we assign a weight $w(i, j)$ (in step 3 of Algorithm 4) given by the increment of aggregate cardinality if (i, j) is activated. The link with the maximum weight is selected for activation. Ties are broken randomly. Greedy-Incremental algorithm has complexity of $\theta(m^4)$.

Algorithm 4 Greedy-Incremental Algorithm

```

1:  $r := 0$  and  $S_{Ginc} := [\cdot]$ 
2: while  $\mathcal{L}(S_{Ginc}, r) \neq \emptyset$  do
3:    $w(i, j) := 2|O_i(S_{Ginc}, r) \cup O_j(S_{Ginc}, r)| - |O_i(S_{Ginc}, r)| - |O_j(S_{Ginc}, r)| \quad \forall (i, j) \in \mathcal{L}(S_{Ginc}, r)$ 
4:    $(i, j) = \arg \max_{(i, j) \in \mathcal{L}(S_{Ginc}, r)} w(i, j)$  % Ties are broken arbitrarily
5:   Activate link between node  $i$  and  $j$ 
6:    $S_{Ginc} \leftarrow [S_{Ginc}, (i, j)]$ 
7:    $r \leftarrow r + 1$ 
8: end while

```

E. Rarest First Algorithm

Rarest first algorithm activates the link so that availability for the subset of segments with the minimum number of nodes is maximally increased. The steps have been described in detail in Algorithm 5. Rarest first algorithm has complexity of $\theta(nm^4)$.

In each round (Lines 2-9), universe \mathcal{N} is divided in m partitions such that p^{th} partition \mathcal{N}_p of segments is available with p nodes. The function $f(\mathcal{N}_p, (i, j))$ evaluates the increment in the availability of segments in \mathcal{N}_p if the link between nodes i and j are allowed to exchange segments

Algorithm 5 Rarest first Algorithm

```

1:  $r := 0, S_{rare} := [\cdot]$ 
2: while  $\mathcal{L}(S_{rare}, r) \neq \emptyset$  do
3:    $\mathcal{N}_p := \left\{ e \in \mathcal{N} : \sum_{j \in \mathcal{M}} \mathbb{I}_{\{e \in O_j(S_{rare}, r)\}} = p \right\} \quad \forall 1 \leq p \leq m$ 
4:    $f(\mathcal{N}_p, (i, j)) := \sum_{e \in \mathcal{N}_p} \mathbb{I}_{\{e \in O_i(S_{rare}, r) \cap O_j^c(S_{rare}, r)\}} + \mathbb{I}_{\{e \in O_j(S_{rare}, r) \cap O_i^c(S_{rare}, r)\}} \quad \forall 1 \leq p \leq m, (i, j) \in \mathcal{L}(S_{rare}, r)$ 
   % Increment in the availability of segments in  $\mathcal{N}_p$  if link between  $i$  and  $j$  is activated
5:    $\mathbf{R}_{(i,j)} = [\mathbb{I}_{\{O_i(S_{rare}, r) \cup O_j(S_{rare}, r) \neq \mathcal{N}\}} \quad f(\mathcal{N}_1, (i, j)) \quad f(\mathcal{N}_2, (i, j)) \quad \cdots \quad f(\mathcal{N}_m, (i, j))]_{1 \times (m+1)} \quad \forall (p_1, p_2) \in \mathcal{L}(S_{rare}, r)$ 
6:   Define link activation preference matrix  $\mathbf{L}_A$  with  $\mathbf{R}_{(i,j)} \forall (i, j) \in \mathcal{L}$  as row vectors.
7:   Sort rows of  $\mathbf{L}_A$  such that column 1 is descending order, then by column 2 in descending order, then by column 3 in descending order and so on.
8:    $(i, j) := \arg(\text{Row at the top of } \mathbf{L}_A)$ 
9: end while

```

where $1 \leq p \leq m$ and $(i, j) \in \mathcal{L}(S_{rare}, r)$. Then for each of the links $(i, j) \in \mathcal{L}(S_{rare}, r)$ we define a row vector $\mathbf{R}_{(i,j)}$ having $m+1$ entries. First entry indicates the whether the nodes will have the universe if link (i, j) is activated. Following entries record the increment in each of the partitions of the universe if the link (i, j) is activated in increasing order of p .

Then we arrange different rows in form of a link activation matrix \mathbb{L}_A such that column 1 is in descending order, followed by column 2 and so on. Sorting rows of \mathbb{L}_A in descending order by column 1 deters activating links which can produce nodes having universe. To break the tie among links having same value in column 1, we sort in descending order by column 2. This would give more preference to links activating whom will increase the segments which are available with only 1 node or are rarest. To break the tie further we use column 3 so that availability of the segments which are with 2 nodes can be increased. This process is continued for all columns.

After completion of this process link corresponding to the top row is activated.

IV. PERFORMANCE COMPARISON

Fig. 3 shows a performance comparison of the various algorithms. For a particular set of (number of nodes m , universe size n , initial segment set size k) values, we generate 100 segment sets uniformly at random. For each such “sample point” or “run” the optimal aggregate cardinality is computed, and each of the five algorithms simulated. The average aggregate cardinality values (over 100 sample points) are shown, along with 95% confidence intervals.

If an algorithm achieves the *optimal* aggregate cardinality corresponding to a sample point, the algorithm is said to be “successful” on that run. Fig. 3 also shows the “success rate” of an algorithm, which is the fraction of runs for which the algorithm was successful. The average shortfall from the optimal aggregate cardinality is shown as well. Lastly, Fig. 3 displays the value of $p_{m,n,k}$, calculated using (2). We recall that $p_{m,n,k}$ is the probability that after the initial choice of segment sets uniformly at random, the universe \mathcal{N} is available among the nodes.

Our results show that the Greedy Links (*GL*) algorithm performs the best—it is able to achieve an aggregate cardinality close to the optimal consistently (success rate $\geq 95\%$). It is followed closely by the Rarest First (*RF*) with similar success rates and mean shortfalls. On an average, Randomized algorithm (*R*) also performs better than Greedy Incremental (*GI*) and Polygon (*P*) algorithms.

The aggregate cardinality achieved by the *R* algorithm is very close to the optimal, but in most runs, it falls just a little bit short. For this reason, we see in Fig. 3 that the heights of the bars corresponding to the optimal and *R* algorithms match almost exactly, yet the success rate of the *R* algorithm is quite low. On the other hand, the *GL* algorithm enjoys a very high success rate, and therefore, is nearly optimal. It is can be seen that the mean shortfall corresponding to the *GL* algorithm is zero.

The *RF* algorithm starts off on a very promising note, but as the problem size gets bigger, there is a dip in performance. As (m, n, k) increases from (15, 20, 5) to (40, 50, 5), the success rate drops from 95% to 76%, while the mean shortfall increases from 0.2% to 0.8%. It is noteworthy that the shortfall of the *R* algorithm reduces as the problem size gets bigger.

Our results indicate that the *GI* algorithm does not perform well in general, even though it is a “natural” greedy algorithm—it chooses to activate the link that yields the largest immediate benefit (the largest increase in aggregate cardinality). Similarly, even though the *P* algorithm can be optimal under certain conditions, it does not do well in general.

The significance of $p_{m,n,k}$ is this: If $p_{m,n,k}$ is high, then there is a good chance that the social group will need to download *only a few* segments using the expensive resource. A high value of $p_{m,n,k}$ means that with high probability, the full universe \mathcal{N} is scattered among the nodes in the group. In such a situation, our results suggest that the *GL* or *RF* or *R* algorithm is very likely to achieve near-optimal aggregate cardinality, leaving only a few segments to be downloaded.

Remark: We note that the more complex algorithms exhibit better performance than the less complex ones.

In Table I, we focus on the Randomized algorithm, and ex-

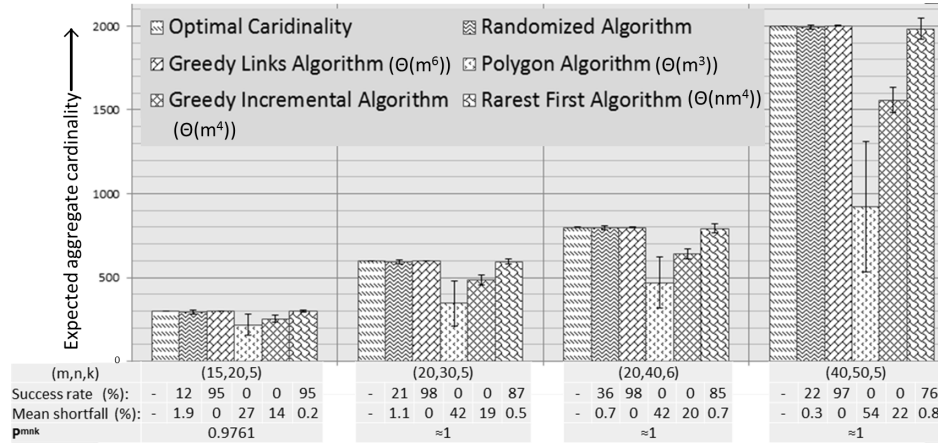


Fig. 3. Expected aggregate cardinality achieved by different algorithms, averaged over 100 randomly generated segment sets. Four different problem instances $((m, n, k))$ are shown, with corresponding $p_{m,n,k}$ values. For each object set, the success rate and mean shortfall of each algorithm are shown as well.

amine the expected aggregate cardinality achieved for different sets of (m, n, k) . It can be seen that as the number of nodes m increases, the expected aggregate cardinality approaches mn ; this is consistent with the asymptotic optimality and approximation ratio results shown in Theorem 1.

TABLE I
BOUNDS FOR RANDOMIZED ALGORITHM

m	n	k	$E(\alpha(S_{rand}))$ (Simulation)	Lower bound on mean aggregate cardinality
60	100	3	5027.0 ± 347.9	3867.4
60	100	5	5715.7 ± 219.1	4829.2
60	100	7	5919.4 ± 127.6	5318
80	200	15	15959 ± 102	15106
100	300	15	29819 ± 254	27486

V. CONCLUSIONS AND FUTURE WORK

The problems studied in this paper were motivated by the context of a social group, in which each group member was interested in obtaining a universe of segments. The approach was to study how mutual exchanges among the group members can help in maximizing the aggregate cardinality of nodes' segment sets, at low cost. To tackle the problem of free riding that arises in such situations, we proposed the novel GT criterion, and explored a number of algorithms for exchange of segments, where each exchange had to be GT-compliant. Analysis of the algorithms yielded interesting properties, and performance was benchmarked against the optimal aggregate cardinality.

The present model assumes that nodes arrive empty-handed and download segments using the expensive resource, to kick-start the process of GT-compliant mutual exchanges. After this process is over, still more segments may need to be downloaded; it will be interesting to understand the best download strategy in the second round, utilizing the state of the system at the end of the first. Further, a central scheduler with a view of all the nodes' segment sets has been assumed in this paper; we would like to understand the issues when nodes act on their own.

REFERENCES

- [1] J. Scott, *Social network analysis*. SAGE Publications Limited, 2012.
- [2] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 12, pp. 1401–1413, 2006.
- [3] E. Jaho, M. Karaliopoulos, and I. Stavrakakis, "Social similarity favors cooperation: The distributed content replication case," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 601–613, 2013.
- [4] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-riding and whitewashing in peer-to-peer systems," in *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, ser. ACM PINS 2004.
- [5] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *SIGecom Exch.*, Jul. 2005.
- [6] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free riding in bittorrent is cheap," in *In HotNets*, 2006.
- [7] R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, and H. Sips, "Improving efficiency and fairness in p2p systems with effort-based incentives," in *Communications (ICC), 2010 IEEE International Conference on*, may 2010, pp. 1–5.
- [8] H. Nishida and T. Nguyen, "A global contribution approach to maintain fairness in p2p networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 6, pp. 812–826, 2010.
- [9] M. Karakaya, I. Korpeoglu, and O. Ulusoy, "Free riding in peer-to-peer networks," *Internet Computing, IEEE*, march-april 2009.
- [10] J. J.-D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. Epema, and H. J. Sips, "Give-to-get: free-riding resilient video-on-demand in p2p systems," in *Electronic Imaging 2008*. International Society for Optics and Photonics, 2008, pp. 681 804–681 804.
- [11] J. R. Douceur, "The sybil attack," in *Peer-to-peer Systems*. Springer, 2002, pp. 251–260.
- [12] H.-H. Dinger, J., "Defending the sybil attack in p2p networks: taxonomy, challenges, and a proposal for self-registration," in *Availability, Reliability and Security, 2006. ARES 2006.*, 2006, pp. 8 pp.–.