Problems in Wireless and P2P communication

A Project Report

submitted by

PANANJADY MARTIN ASHWIN

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **Problems in Wireless and P2P Networks**, submitted by **Pananjady Martin Ashwin**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Andrew Thangaraj
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 5th May 2014

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my guide, Dr. Andrew Thangaraj for his invaluable support and guidance throughout the duration of my project. He is without doubt one of the coolest professors I know, and he has been a major inspiration in my decision to study further in this field.

I also thank Prof. Rahul Vaze of TIFR and my friend and classmate Vivek Bagaria, with whom I worked on the P2P network problem that forms a part of this report. I would also like to thank Prof. Vaze for his guidance in the other two problems addressed in this report.

I would also like to thank all those who have contributed to making my experience at IITM a memorable one, and all the friends that I have made over these four years. I have learnt more from many of them in these 4 years than I learnt in all the years before college.

I would like to thank my family for being incredibly supportive throughout and for their constant encouragement and unconditional love. I owe a lot of what I am today to the way in which my parents raised me, and to my sister's constructive criticism.

Ashwin.

ABSTRACT

We addressed 3 problems that arise in various contexts in wireless communication. The first was the theoretical problem of finding the integrality gap of a formulation of the coverage lifetime problem in sensor networks. We were motivated to work on this problem by a conjecture posed in Berman *et al.* (2004) which claimed that it was 2, and we disproved that conjecture by showing a counterexample that used projective geometries. We were also able to show certain other related results, but were not able to find the integrality gap exactly. The second problem involved a new paradigm of file exchange in P2P networks called the Give and Take Protocol, to encourage fairness and prevent free-riding. We provided randomized algorithms for this problem and showed guarantees on their performance. The third problem had to do with Online Algorithms for Basestation allocation, in which we attempted to build on previous work by accounting for queuing of users and their departures. We considered the processor sharing model in which the rate achieved by a user at a basestation is equal to the total rate suppliable by the basestation divided by the number of users being served at that basestation. We showed that for the case in which users have a fixed data demand, the queues at the basestations have very stringent stability conditions, and therefore that the model was not worth studying further from a competitive ratio perspective. We then considered an alternate model in which broadcast channels exist at the basestations, and the rate available to each user is divided by the number of distinct files being served at that basestation. For this case, we showed that certain probability distributions on the files give rise to stable queues. We also analysed the online vs. offline competitive ratios for some algorithms in this context.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
P2P	Peer-to-peer
MLCP	Maximum Lifetime Coverage Problem
DSCP	Disjoint Set Cover Problem
IG	Integrality Gap
LP	Linear Program
IP	Integer Program
GT	Give-and-Take

CHAPTER 1

INTRODUCTION

Wireless communication has thrown up a variety of open problems over the last 30 years, many of them theoretical. In this report, we aim to study 3 such problems that span 3 sub-fields within wireless communication - sensor networks, peer-to-peer networks and load balancing at data centers.

We will provide detailed introductions to each problem along with the necessary background and notation, in the chapters that follow. All three problems build on a substantial amount of existing work. Two out of three problems still do not have conclusive results, while the third throws up many new open questions.

This report has been compiled in the chronological order in which the problems were tackled. Chapter 2 addresses the maximum lifetime coverage problem in wireless networks, and deals mainly with the concept of integrality gaps of binary integer programs. It aims to analyse the integrality gap of a specific subset of packing LPs. Chapter 3 looks at a newly proposed, stringent paradigm of file exchange in P2P networks and deals with probabilistic algorithms for the problem. Chapter 4 has to do with online algorithms for basestation allocation, in which users arrive at arbitrary times but with fixed data demands. We analyse regimes of rate sharing at the data centers both in the context of stability of the resulting queues and competitive ratios of online algorithms.

CHAPTER 2

Integrality Gap of the Maximum Lifetime Coverage Problem in Wireless Sensor Networks

The introductory text of this chapter follows Bagaria *et al.* (2013) almost verbatim. The text that follows is unique to this report.

2.1 Introduction

Wireless sensor networks are deployed for a variety of applications - military, data collection, and health-care, to name a few - and most of these entail monitoring or covering a specific geographic area. Therefore, maximizing the lifetime of coverage in a wireless sensor network with battery-limited sensors is a fundamental and classical problem, well studied in literature Cardei *et al.* (2005); Cardei and Wu (2006); Berman *et al.* (2004); Ding *et al.* (2012). Typically, a large number of sensors is deployed in a given area, and consequently, many sub-collections of these sensors can cover/monitor all the intended targets. Each such sub-collection of sensors is called a *set cover*. To maximize the lifetime with the practical constraint of limited battery capacity, we need to find an activity schedule for each sensor (signifying when it must be turned *on* or *off*) that ensures that all intended targets are covered/monitored for the longest time possible.

Concisely, the maximum coverage lifetime problem (MLCP) is as follows. Given a set of sensors and a set of targets, find an activity schedule for these sensors such that (i) the total time of the schedule is maximized, (ii) all targets are constantly monitored (i.e. at any point of time, at least one of the set covers is active), and (iii) no sensor is used for longer than what its battery allows.

In literature, the MLCP has been approached using two methods. The first method involves solving the maximum disjoint set cover cover problem (DSCP) Bollobás *et al.*

(2013). The DSCP finds the maximum number of set covers such that any two set covers are pairwise disjoint. Clearly, sequentially turning on each of the disjoint set covers found by the DSCP provides a feasible solution to MLCP. This approach has been used in Cardei and Du (2005); Ahn and Park (2011); Cardei and Du (2005); Slijepcevic and Potkonjak (2001); Lai *et al.* (2007), and the DSCP has been solved heuristically. The first provably approximate solution to the DSCP was given in Bagaria *et al.* (2013), in which it was shown that the same approximation ratio extended to the MLCP as well.

The second method to solve the MLCP uses non-disjoint set covers, i.e. set covers that are not constrained to be disjoint. The optimal solution obtained using this method will exactly match the optimal solution of the MLCP, unlike the DSCP approach. This approach has been used by several papers - Cardei *et al.* (2005); Berman *et al.* (2004); Kasbekar *et al.* (2011); Zhao and Gurusamy (2008); Pyun and Cho (2009). Some of them are heuristic while others provide provable guarantees to the MLCP. It is also worth mentioning here that the special *geometric* case of the MLCP has also been studied Ding *et al.* (2012), in which each sensor can monitor a circular area around itself with a given radius.

The relationship between the MLCP and DSCP is an interesting one. It is obvious from the description above that any solution to the DSCP is also a solution to the MLCP. In fact, the optimal solution of the MLCP is always greater than equal to that of the DSCP, as we will demonstrate in Section 2.2.

Let us define $\frac{OPT(MLCP)}{OPT(DSCP)} = IG$, where $OPT(X)$ denotes the optimal solution of problem X. It was conjectured in Berman *et al.* (2004) that:

Conjecture 2.1.1 $IG \leq 2$ *for any input of targets and sensors (with identical battery capacities).*

It was also conjectured in Berman *et al.* (2004) that:

Conjecture 2.1.2 $IG \leq 1.5$ *for inputs of targets and sensors (with identical battery capacities) in which the sensor coverage regions are known to be convex.*

It is worthwhile to mention here that both the DSCP and MLCP are NP-complete,

Cardei and Du (2005); Cardei *et al.* (2005), and so evaluating optimal solutions of either is not a trivial task. Approximate solutions cannot directly help us with the conjectures, although they will provide upper bounds on integrality gaps as we will see in the following sections.

2.1.1 Impact of a proof of conjectures 2.1.1 and 2.1.2

If the conjectures are true, and the DSCP and MLCP indeed have a constant (even non 2) integrality gap, then it justifies approaching the MLCP through the DSCP from the point of view of an accurate solution. Already, the approach has many functional advantages, as enumerated in Bagaria *et al.* (2013), and showing that the two problems are similar in their solutions would mean that the DSCP approach to the MLCP would always be preferred.

More generally, this would influence the general class of LPs to which the MLCP belongs, known as Packing LPs or Multiple Knapsack problems. In general, these LPs have an unbounded integrality gap, but there has been some literature on constructing examples with a constant integrality gap Pritchard (2010); Anagnostopoulos *et al.* (2013).

The main result in this report however is the disproof of Conjectures 2.1.1 and 2.1.2, although we do not show that IG grows unbounded. We show in Section 2.2 that the conjectures can in fact be formulated as an integrality gap problem, and provide some insight and devise methods to analyse that integrality gap.

2.2 Preliminaries

We define a universe of targets $\mathcal{U} = \{1, 2, 3, \ldots, n\}$, where n is the number of targets. We will hereafter refer to each target as an *element*. Each sensor i can cover a subset of targets $S_i \subseteq \mathcal{U}$, and so the sensors are defined by the multiset $\mathcal{S} = \{S_1, S_2, \ldots\}$. Hereon, we use S_i to denote a sensor and call each S_i a *subset*. Let each sensor S_i have a battery capacity b_i. Since we are interested in monitoring all targets, we define a *set*

cover $C \subseteq \mathcal{S}$ to be a collection of sensors such that sensors in C cover the universe, i.e., $\bigcup_{S_i \in C} S_i = \mathcal{U}$. The idea is to switch on set covers sequentially so as to prolong the time for which all elements can be monitored (which we call the network lifetime), while ensuring that each sensor is used only for as long as its battery will allow. The formal definition of the MLCP is as follows:

Problem 1 (MLCP) *Let* $\mathcal{C} = \{ C_1, \ldots, C_m \}$ *be the collection of all set covers from* $\mathcal{S}$ *and* t_i *be the time for which set cover* C_i *is switched on. Then the MLCP is to*

$$\begin{aligned} &\textit{Maximize :} \qquad \sum_{j=1}^{m} t_j \\ &\textit{Subject to} \qquad C_{ij} t_j \leq 1, \qquad \forall\, i, \\ &t_i \in [0,1] \;\forall\, i \\ &C_{ij} = \begin{cases} 0 \textit{ if sensor } S_i \textit{ is not in set cover } C_j, \\ 1 \textit{ if sensor } S_i \textit{ is in set cover } C_j. \end{cases} \end{aligned}$$

Lemma 2.2.1 *Without loss of generality, we can consider all sensors to have unit battery capacity, i.e.* $b_i = 1, \forall\, i$.

Proof If $b_i = B, \forall\, i$, then the solution to the MLCP solved using $b_i = 1, \forall\, i$, need only be multiplied by a factor of B to get the required solution. Otherwise, we take the greatest common divisor $b_{com} = gcd\big(b_1, b_2, \ldots \big)$, create b_i / b_{com} copies of each sensor $S_i, \forall\, i$, and consider each copy to be a separate sensor with battery capacity b_{com}. Thus, the problem can be reduced to $b_j = B = b_{com}, \forall\, j \in \mathbf{J}$, where $\mathbf{J}$ is an index set and $|\mathbf{J}| > |\mathcal{S}|$. ■

Problem 2 (DSCP) *Given a universe* $\mathcal{U}$ *and a set of subsets* $\mathcal{S}$*, find as many set covers as possible such that all set covers are pairwise disjoint. In other words, the DSCP may*

be formally stated as:

$$\begin{aligned} &\textit{Maximize :} && \sum_{j=1}^{m} t_j \\ &\textit{Subject to} && C_{ij}t_j \leq 1, \qquad \forall\, i, \\ &t_i \in \{0,1\}\ \forall\, i \end{aligned}$$

The DSCP necessitates that any subset can be present in a maximum of one set cover. If the number of disjoint set covers is k, then using each of the k disjoint set covers for one time unit, clearly, we have an MLCP solution of k with $b_i = 1, \forall\, i$. Thus, solving the DSCP provides a feasible solution to the MLCP.

However, the solution of the MLCP differs from that of the DSCP as shown in Ding *et al.* (2012), because the optimal solution to the MLCP may not always involve disjoint set covers. For example, let $\mathcal{U} = \{1, 2, 3\}$ and $\mathcal{S} = \{S_1, S_2, S_3\}$, where $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$ and $S_3 = \{3, 1\}$. Clearly, the maximum number of disjoint set covers (and therefore network lifetime) is 1, while if we operate the sensors as follows: $\{S_1, S_2\}$ for 0.5 units of time, $\{S_2, S_3\}$ for 0.5 units of time, and $\{S_1, S_3\}$ for 0.5 units of time, the lifetime is 1.5 time units. This is shown pictorially in figure 2.1.

Ironically, however, Bagaria *et al.* (2013) showed that the best possible algorithm to solve the DSCP is also one of the best possible algorithms to solve the MLCP by showing matching upper and lower bounds on the approximation ratio.

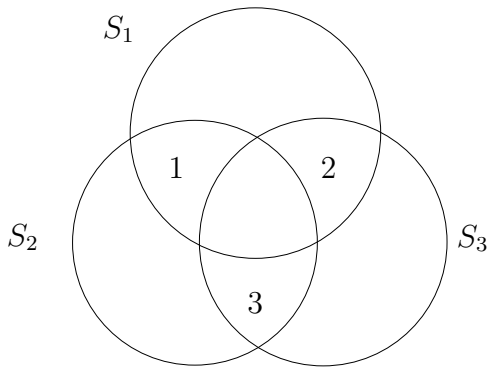

Figure 2.1: Case where the solution to MLCP is greater than DSCP

Note that Problem 1 is effectively the LP relaxation of Problem 2. We are interested in the ratio $IG = \frac{OPT(MLCP)}{OPT(DSCP)}$, which is the integrality gap between problems 1 and 2. Problem 1 also belongs to a class of LPs called packing LPs, which for an arbitrary constraint matrix C have an infinite integrality gap. In our case however, C is constructed in a very particular form, and so this case may have other properties with respect to its integrality gap. This is what we aim to analyse.

2.2.1 Terminology

(i) n : Number of elements in the universe $|\mathcal{U}|$. **(ii)** $|\mathcal{S}|$: Number of subsets. Note how $\mathcal{S}$ has been defined as a multiset. This is because subset repetitions are possible, since multiple sensors may cover the same targets. $|\mathcal{S}|$ is therefore the *total* number of subsets, not the number of distinct subsets. **(iii)** R : Maximum size of a subset S_i = $\max_i |S_i|$ **(iv)** F_i : The *frequency* F_i of any element $i \in \mathcal{U}$ is defined as the number of subsets $S_j \in |\mathcal{S}|$ that it appears in. $F_i = \#\{S_j : i \in S_j\}$. **(v)** $F_{min} : \min_i F_i$. **(vi)** $F_{max} : \max_i F_i$.

2.2.2 Known, relevant results

There are a few observations to be made about the DSCP (refer Problem 2 for formal definition) with respect to the terminology we defined in Section 2.2.1. The solution to the DSCP cannot be more than F_{min}, since the element with frequency F_{min} can only be present in a maximum of F_{min} disjoint set covers. We therefore have a trivial upper bound of F_{min} on the solution to the DSCP.

Note that the MLCP solution is also upper-bounded by F_{min}, since the element with the least frequency has only F_{min} sensors covering it, and at least one of these sensors must be on at all times to ensure coverage.

In Bagaria *et al.* (2013), a deterministic algorithm was shown which given any universe and set of sensors (all with unit battery capacity) returns a lifetime of $\frac{F_{min}}{/} c \ln n$ (where $1 < c < 2$), by finding an equal number of disjoint set covers. This was effectively an $\mathrm{O}(\ln n)$ approximation algorithm to both the MLCP and DSCP, since both are

upper-bounded by F_{min}.

Another interesting observation is that this also gives us an upper-bound on IG as being $2 \ln n$, since the optimal solution of the DSCP is at least $\frac{F_{min}}{l} 2 \ln n$ and that of the MLCP is at most F_{min}.

We will now move on to our results themselves - disproofs of Conjectures 2.1.1 and 2.1.1.

2.3 Counterexample for Conjecture 2.1.1 - The Fano Plane

A few examples should tell the reader that the disproof is quite hard to find. Our disproof is a counter-intuitive projective geometry $PG(2, 2)$ - the Fano Plane. A diagrammatic representation is shown in figure 2.2. Consider each point in the figure - 7 in all - to be our universe elements, and each line - again 7 in all (including the circle) - to be the sensors. Note that if you consider all the sensors that pass through a point, you end up with a set cover, and so there are 7 minimal set covers[1].

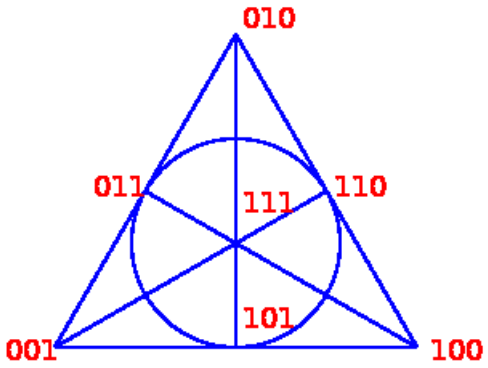

Figure 2.2: The Fano Plane. Points are labelled in keeping with the group theoretic construction, which will be explained in Section 2.5.1.

Also note that once a set cover has been formed, the other 4 sensors cannot form a set cover on their own. So this example has only **1** disjoint set cover. In other words,

[1]A minimal set cover is defined as one in which the removal of any sensor destroys the set cover property. It should be obvious that is is sufficient to consider just set covers of this form.

$OPT(DSCP) = 1$. On the other hand, consider the operation of each of the 7 set covers for time $1/3$. Note that no sensor is overused - each sensor forms a part of exactly 3 set covers, and so is operated for 1 time unit totally. Each of these set covers can be switched on sequentially, and so $OPT(MLCP) = 7 \times \frac{1}{3} = 7/3$ time units. So for the Fano Plane,

$$\frac{OPT(MLCP)}{OPT(DSCP)} = \frac{7}{3} > 2 \tag{2.1}$$

and hence, we have disproved Conjecture 2.1.1.

Note that this is still a weak disproof of the result. Ideally, a disproof ought to have consisted of an example in which the integrality gap grew as a function of n or $|S|$ without a constant upper bound, and that is still an open problem.

2.4 Counterexample for Conjecture 2.1.2 - The Polygon

Note that the disproof is Section 2.3 was for a case of the problem in which sensors could cover non-convex areas, and Conjecture 2.1.2 applies to those cases in which sensor coverage areas are convex. The conjectured bound is 1.5, which is even tighter that that of Conjecture 2.1.1. Conjecture 2.1.2 basically claims that the example shown figure 2.1 achieves the highest possible integrality gap for convex sensing regions.

In this section, we provide an example in which the integrality gap approaches $2-\epsilon$. This example is much more simple, and uses a very intuitive construction. Consider a polygon with an odd number of vertices/edges. Let each vertex represent a universe element and each edge represent a sensor. Note that each sensor covers two elements and each element is covered by two sensors.

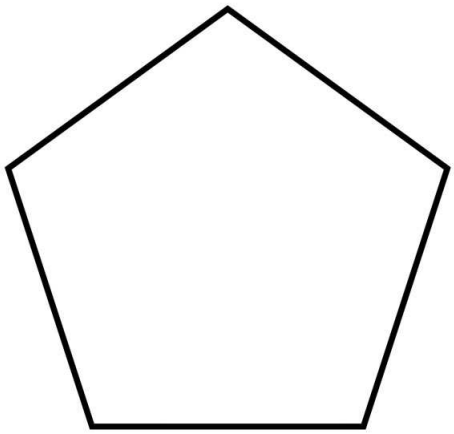

Figure 2.3: The Pentagon. The integrality gap is $5/3$ for this construction.

We will use the pentagon in Figure 2.3 as an example. Label the edges from 1 to 5 in some cyclic order. Note that there exist 5 minimal set covers - 135, 241, 352, 413 and 524. This is basically obtained by what we call the `SKIP` algorithm: starting at some edge and picking alternate edges in a particular direction. We could have started at one of 5 edges, and so there must be 5 minimal set covers. Note that each sensor in the pentagon appears in 3 of those set covers, so operating each set cover for time $1/3$ will give us an MLCP solution of $5/3$. The DSCP solution is of course 1. $IG = 5/3$, which disproves conjecture 2.1.2.

We now extend this analysis to all polygons with an odd number of vertices. Note that the DSCP is always 1 because any two minimal set covers will always have some nontrivial intersection. There will exist n minimal set covers (similar to the polygon example), and each sensor will be part of $(n+1)/2$ of these set covers. That is reasonably simple to show, since in the `SKIP` algorithm, any sensor will be picked up in alternate set covers. This means that each set cover can be operated for time $2/(n+1)$ without violating any battery capacity constraints.

So the MLCP solution will be $n \times \frac{2}{n+1} = \frac{2n}{n+1} = 2 - \frac{2}{n+1}$. Denoting $2/(n+1)$ by ϵ, which goes to 0 as $n \to \infty$, we can say that $IG = 2 - \epsilon$ for some arbitrary $\epsilon > 0$.

We emphasize here again that this is a weak disproof, and that a stringer result would be to show that IG grows without bound for some family of examples.

We will now describe some methods that we tried which did not give us positive or conclusively negative results, but whose insights may help in tackling this problem in the future.

2.5 Other Insights and Methods

2.5.1 The general projective geometries

The counterexample for the conjecture is provided by the Fano plane, which is a projective geometry of the form $PG(2, 2)$. It is therefore natural to ask what happens when one considers a projective geometry $PG(n, q)$?

We analyze this problem for $PG(2, q)$ first. Note that here, there are $q^2 + q + 1$ lines and $q^2 + q + 1$ points. Each line passes through $q + 1$ points. It is easy to show by the group theoretic construction that all lines that pass through a particular point form a set cover, and each line is part of $q + 1$ set covers. So a lower bound on the MLCP solution is therefore $\frac{q^2+q+1}{q+1}$. It is possible to show that this upper bound is tight, by considering the polytope formed by the LP feasible region. It turns out that the solution corresponding to $q^2 + q + 1$ is a corner point.

The solution of the DSCP, on the other hand, is not easily computable. This is unlike the usual case in which it is hard to get a handle on the MLCP solution. But for all the simulated cases[2], the IG was within 2 as q increased. IG, in fact, seemed to get smaller.

2.5.2 An unsuccessful attempt to prove constant integrality gap

In this section, we describe a method by which we tried to prove the constant integrality gap of the LP through a dual formulation of the MLCP optimization problem.

[2]A library to group-theoretically construct $PG(n, q)$ and its m-flats is up at `http://github.com/ashwinpm/pgnq`.

Dual Formulation of Problem 1

The dual formulation of Problem 1 would look thus:

Problem 3

$$\textit{Minimize :} \qquad \sum_{j=1}^{|\mathcal{S}|} y_j$$

$$y_j \in [0,1] \forall j = 0, 1, \ldots, |\mathcal{S}|$$

$$\textit{Subject to} \qquad D_{ij} t_j \geq 1, \qquad \forall\, i,$$

$$D_{ij} = \begin{cases} 0 \textit{ if cover } C_i \textit{ does not contain subset } S_j, \\ 1 \textit{ if cover } C_i \textit{ contains subset } S_j. \end{cases} \qquad i = 0, 1, \ldots, m; j = 0, 1, \ldots, |\mathcal{S}|$$

Note that although the physical meaning of each y_i is unclear, this formulation is what gives us bounds such as $OPT(MLCP) \leq F_{min}$, which is obtained by setting y_i to 1 if subset S_i contains the element with frequency F_{min}.

IP interpretation of the Dual

In the IP formulation of the dual, each y_i must now take binary values. This formulation is effectively a minimum weight set cover problem, in which each set cover can be considered to be an element of a universe $\mathcal{C}_U$ and each sensor is a newly defined collection of set covers - a sensor “contains" a set cover if it is present in that cover. The problem is then to find the minimum number of sensors which are together present in all set covers. Let us denote the optimal solution to this problem by $OPT(IP_{dual})$.

Result to be proved

Let us denote the optimal solution to Problem3 by $OPT(Dual)$. We are interested in proving the following result: $OPT(MLCP) \leq 2OPT(DSCP)$. Now, note that $OPT(MLCP) = OPT(Dual) < OPT(IP_{dual})$. So it is sufficient to prove that

$OPT(IP_{dual}) \leq cOPT(DSCP)$, where c is some constant, to show that $OPT(MLCP) \leq cOPT(DSCP)$.

This now gives us a quantity that is easier to handle combinatorially. We now need to prove that,

Claim 2.5.1 *Given all set covers, we can find ck sensors which, between them, are present in all set covers, where k is the number of disjoint set covers and c is some constant.*

Note that a proof of claim 2.5.1 will provide a proof (within a constant factor) to Conjecture 2.1.1, but a disproof of 2.5.1 will not carry as a disproof of Conjecture 2.1.1. This is because claim 2.5.1 actually involves two integrality gaps together - the primal IG$\times$ dual IG.

We disproved claim 2.5.1 by considering a random bipartite graph with sensors and elements. We were able to show that c in claim 2.5.1 actually grows unbounded in n.

While this was an unsuccessful attempt, the analysis of the dual may help any future work on the problem.

CHAPTER 3

Enforcing Fair Exchange in P2P networks

This chapter is more or less the text of the paper Maximizing Utility Among Selfish Users in Social Groups Pananjady *et al.* (2014), which won the Best Paper Award in the Networks track of the National Conference on Communications (NCC) 2014 held at IIT Kanpur.

3.1 Introduction

We consider a peer-to-peer (P2P) network scenario, where each node (or user) desires a certain fixed set of files residing on a central server, e.g. a full high-definition movie Knoke and Yang (2008). For each node, the cost associated with downloading all the files from the server is prohibitively large. However, the cost of exchanging files across the nodes is very low. The high cost of file acquisition from the server could be due to large delay, license fee, power etc., while the low cost of P2P exchange could be due to nodes sharing bandwidth, physical proximity between the nodes, mutual cooperation, etc. Popular applications based on this concept include web caching, content distribution networks, P2P networks, etc. Device-to-device (D2D) communication in cellular wireless networks is another example, where mobile phones cooperate with one another to facilitate communication.

Clearly, to keep cost low, it is intuitive for each node to download a small fraction of files from the server, and then obtain the remaining files through exchanges with other nodes. Free riding -the phenomenon in which a node does not download anything and still obtains all the files from other nodes - is possible in P2P networks, and can lead to attacks like whitewashing, collusion, fake services, Sybil attack Karakaya *et al.* (2009); Fox (2001); Dinger and Hartenstein (2006), etc. To avoid free riding Feldman *et al.* (2006); Feldman and Chuang (2005); Locher *et al.* (2006); Rahman *et al.* (2010);

Nishida and Nguyen (2010); Karakaya *et al.* (2009); Mol *et al.* (2008), the file exchange among nodes follows a *give-and-take* criterion. Two nodes can exchange files if both have some file(s) to offer each other.

The problem we consider in this paper is to find an algorithm for scheduling file exchanges between nodes such that at the end of algorithm, when no more exchanges are possible due to the condition imposed by the give-and-take criterion, the number of nodes that receive all files is maximized. Prior work Aggarwal *et al.* (2013) has considered maximizing the aggregate number of files received by all nodes. Depending on the utility of the user, e.g. watching a movie, maximizing the number of nodes that obtain all the files could be more important than maximizing the total number of files across the network.

Finding an optimal algorithm for scheduling file exchanges with the give-and-take model is challenging, since at each step there are many pairs of nodes that satisfy the give-and-take criterion and the choice of exchange at each step determines the final outcome. As a consequence, the number of feasible schedules is exponential. We therefore concern ourselves with providing approximation algorithms for the problem that have polynomial time complexity. An algorithm to solve a maximization problem is said to have an approximation ratio $\rho > 1$ if it always returns a solution greater than $\frac{1}{\rho}$ times the optimal solution. We call such an algorithm a ρ approximation algorithm, or simply a ρ algorithm.

We deal with the following file acquisition paradigm. Let each user download each file from the server independently with probability p. Let there be a total of n files and m users. Note that the download cost for each user is directly proportional to p, since the cost follows a binomial distribution. Users can either choose p before entering the system (a-priori) or after (a-posteriori), in which case it could depend on the number of users or files. In this paper, we propose a deterministic, iterative tree-splitting algorithm with polynomial complexity. The algorithm involves recursively dividing the users into two equally-sized groups, thereby forming a tree. The divisions are defined keeping the give-and-take criterion in mind. Exchanges are then effected from the leaves upward.

Our contributions for various regimes of m and n are listed below. All of these hold

with very high probability as $n \to \infty$:

(i) For $m = \mathrm{O}(\log n)$, our algorithm has an approximation ratio $\rho = 2$, provided p can be chosen a-posteriori such that $\exp(-2\log n/m) < 1 - p < \exp(-\log n/m)$.

(ii) When $m = \mathrm{O}(n)$, we present an algorithm that has an approximation ratio of $\rho = 2$ when p is chosen a-priori (users can choose to incur a vanishingly small cost to themselves by minimizing p), and an approximation ratio of $\rho = 1 + \epsilon_1$, $(\epsilon_1 > 0)$ when p can be chosen a-posteriori (the cost per user depends on m, n, and on the choice of ϵ_1 in the algorithm).

(iii) When $m = \mathrm{O}(n^z)$ for some $z > 1$, we present an algorithm that has an approximation ratio of $\rho = 1 + z$ when p is chosen a-priori (small cost), and an approximation ratio of $\rho = \frac{1+z}{2} + \epsilon_2$, $(\epsilon_2 > 0)$ when p can be chosen a-posteriori. The cost in the latter case is again dependent on m, n and ϵ_2.

3.2 Preliminaries

3.2.1 Notation

(i) T: set of files on the server (the complete universe); $|T| = n$. **(ii)** $U = \{u_1, u_2, \ldots, u_m\}$: set of all users. **(iii)** C_i^t: set of files that user u_i possesses at time step t. C_i^0 represents the files that user u_i possesses at the start, i.e. before any exchanges. **(iv)** $F = \bigcup_i C_i^0 = \{1, 2, 3, \ldots, f\}$: set of all files in the achievable universe. **(v)** $u_i \leftrightarrow u_j$: exchange between users u_i and u_j. **(vi)** F_G: set of files that a group of users G possesses. $F_G = \bigcup_{i:u_i \in G} C_i^t$ at time t. For example, $F_U = F$.

3.2.2 Problem Definition

We assume that a single central server contains n files represented by the set T. Let m users represented by the set $U = \{u_1, u_2, \ldots, u_m\}$ initially obtain some files from this server, but at high cost to themselves. Let the files obtained by a user u_i be represented by C_i^0. Note that in practice, file acquisition is a distributed process, with each user

picking files without being directed to do so by any central controller. The exchanges in the P2P network, on the other hand, can be centrally controlled, and the controller's objective is to ensure that the users acquire all files in the achievable universe. To that end, we propose the following file acquisition paradigm.

Definition 1 (Random Sampling) *Each user picks up a file with probability p (a file is not picked up with probability $q = 1 - p$). The pickup of files is i.i.d. across files and users.*

The parameters of the paradigm, i.e. the value of p (or equivalently, q) can either be decided a-priori - when the users are oblivious to the number of other users in the social group and so decide the value of p before entering the system - or a-posteriori - when the users appropriately choose the value of p depending on m and n. We will consider both cases.

Let $F = \bigcup_i C_i^0 = \{1, 2, 3 \ldots, f\}$ be the *achievable universe*. The primary objective for each user is to obtain all f files in the achievable universe F, since these represent all the files obtainable through exchanges. To this end, the users are interested in disseminating the files among themselves at low cost. However, since each user is selfish, file transfer between users can only occur via the give-and-take protocol, which we explain using the following definitions:

Definition 2 (GT Criterion Aggarwal *et al.* (2013)) *Two users u_i and u_j satisfy the GT criterion at time t if $C_i^t \not\subseteq C_j^t$ and $C_j^t \not\subseteq C_i^t$.*

Definition 3 (Exchange: $u_i \leftrightarrow u_j$) *Two users can only exchange files if they satisfy the GT criterion. After an exchange between users u_i and u_j (at time t), $C_i^{t+1} = C_j^{t+1} = C_i^t \bigcup C_j^t$.*

Def. 2 & 3 form the cornerstone of the give-and-take protocol.

We call a sequence of exchanges between users a *schedule*, and represent it by $\mathcal{S}$. The set of all schedules is denoted by $\mathcal{X}$. It is clear that any schedule will lead to a situation in which no further exchanges are possible. At this stage, we represent the set of *satisfied* users who have obtained all f files by U_{sat}.

Formally, we are interested in the following problem:

$$\begin{aligned} &\max |U_{sat}| \\ &\text{Subject to: } \mathcal{S} \in \mathcal{X} \end{aligned} \tag{3.1}$$

As mentioned in Section 3.1, there are an exponential number of schedules in $\mathcal{X}$, and our interest lies in approximating (3.1) in polynomial time.

3.3 Recursive Algorithm

To solve (3.1), we present a recursive algorithm, which we call the TreeSplit algorithm, which repeatedly divides the users into two groups of equal size and then effects exchanges appropriately. It is illustrated in Fig. 3.1.

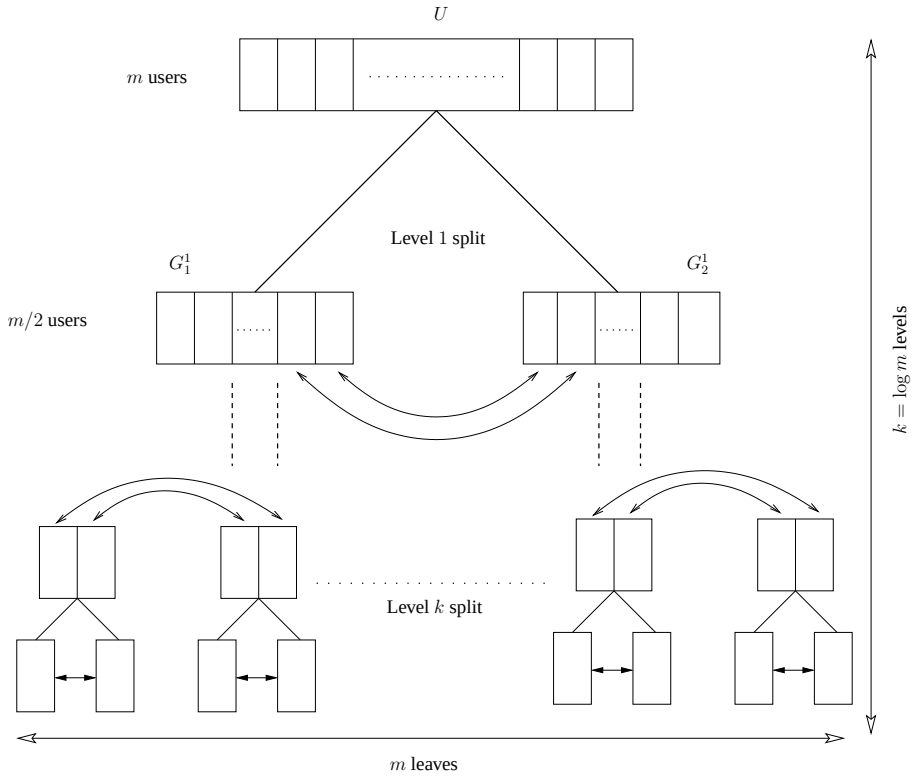

Figure 3.1: TreeSplit of users. $\leftrightarrow$ represents exchange, solid lines arising out of a group represent the division of that group which obeys the splitting condition.

Recall from Section 3.2.1 that the set of files contained by a group of users G is

represented by F_G. Let there be $m = 2^k$ users for some k. Divide the m users into two groups G_1^1 and G_2^1 of $m/2$ users each, satisfying the *splitting condition*, defined as $F_{G_1^1} \not\subseteq F_{G_2^1}$ and $F_{G_2^1} \not\subseteq F_{G_1^1}$. We call this division a *level 1* split, which is indicated by the superscript of groups G_i^1. If we can design a schedule to ensure a scenario in which all users in G_1^1 contain $F_{G_1^1}$, and all users in G_2^1 contain the files $F_{G_2^1}$, we can then effect pairwise exchanges between users in G_1^1, i.e., $u_{iG_1^1}, i = \{1, 2 \ldots, m/2\}$, and users in G_2^1, i.e., $u_{iG_2^1}, i = \{1, 2 \ldots, m/2\}$. In other words,

$$u_{iG_1} \leftrightarrow u_{iG_2} \; \forall \; i \in \{1, 2, \ldots, m/2\}$$

ensuring that all m users obtain all files in F, maximizing (3.1).

Thus, essentially, we have reduced the problem to equivalent problems on two smaller groups of users G_1^1 and G_2^1, separately. Taking this recursion forward, we need to see what happens at the base case (at a level k split), when a group of two users G_j^{k-1} is divided into the groups G_i^k and G_{i+1}^k of one user each, for some i, j. If the division of G_j^{k-1} obeys the splitting condition, then $F_{G_i^k} \not\subseteq F_{G_{i+1}^k}$ and $F_{G_{i+1}^k} \not\subseteq F_{G_i^k}$. Since both G_i^k and G_{i+1}^k consist of one user each, the splitting condition forces the GT criterion to be satisfied between the two users G_i^k and G_{i+1}^k. So we can effect an exchange between users G_i^k and G_{i+1}^k and ensure that G_j^{k-1} contains two users who possess all the files in $F_{G_j^{k-1}}$. Pairwise exchanges as we move up the tree will therefore solve problem (3.1), and the result is summarised in the following Theorem.

Theorem 3.3.1 *For $m = 2^k$ for some k, we can use the TreeSplit algorithm to schedule exchanges such that all users obtain the entire achievable universe F, provided that divisions obeying the splitting condition can be enforced at all levels.*

Notice that the TreeSplit algorithm is valid only if m is a power of 2. We now look at what happens for a general m.

We define the *UniquePick* algorithm, which we apply on the m users as follows: Initially, let $U_{suff} = U$. We now *prune* U_{suff} as follows: Pick any user in U_{suff}. If he contains a *unique* file that no other user in U_{suff} contains, keep him in U_{suff}; otherwise throw him out. Do this in any arbitrary order until all users have been checked for

uniqueness.

Proposition 3.3.2 *The UniquePick algorithm ensures that U_{suff} finally contains all files in the achievable universe, i.e. $F_{U_{suff}} = F$.*

Proof The UniquePick algorithm removes only those users who do not have any unique files. Therefore, even if a user u_i is removed from U_{suff}, every file $j \in C_i^0$ is present in $U_{suff} \backslash u_i$. Since $U_{suff} = U$ initially, and no file of F is removed from all users in U_{suff} up to the end of the Unique-Pick algorithm, U_{suff} must finally contain all files in F. ■

Theorem 3.3.3 *For any m, provided that the splitting condition can be enforced in all divisions, the TreeSplit algorithm has a 2-approximation algorithm to* (3.1).

Proof We consider the power of 2 closest to but less than m, and call this 2^y ($y = \lfloor \log_2 m \rfloor$). Note that $2^y/m > 1/2$, by definition. At the end of the UniquePick algorithm, either **(i)** $|U_{suff}| \leq 2^y$ or **(ii)** $|U_{suff}| > 2^y$. In the case of condition **(i)**, we add some users from $U \backslash U_{suff}$ to U_{suff} so that $|U_{suff}| = 2^y$. We know from Prop. 3.3.2 that 2^y users contain F, so we can now use Thm. 3.3.1 to ensure that 2^y users obtain F provided the splitting condition can be enforced in all divisions. So, at least half the users obtain the achievable universe (since $2^y/m > 1/2$), giving us an approximation ratio of 2.

In the case of condition **(ii)**, we know that more than 2^y users are such that each contains a unique file, and $F_{U_{suff}} = F$ (by Prop. 3.3.2). We can therefore use the Polygon Algorithm from Aggarwal *et al.* (2013) to schedule exchanges and ensure that all users in U_{suff} obtain the universe. Again, more than 2^y users obtain F, giving us an approximation ratio of 2. ■

One important assumption we made in the TreeSplit algorithm was that the splitting condition can be enforced in every division; we did not explain how the division itself is done. In the sections that follow, we will consider different regimes of m and n for which the splitting condition is satisfied with high probability.

3.3.1 Approximation ratio of 2 when $m = \mathrm{O}(\log n)$, $n \to \infty$

When $m = \mathrm{O}(\log n)$, we apply the TreeSplit algorithm to the users. We **randomly divide** groups as we go down the tree. We show that all divisions done randomly satisfy the splitting condition with high probability.

Recall the Random Sampling paradigm, in which a file is not chosen by a user with probability q. Let $m = 2^k$. If all divisions satisfy the splitting condition in this case, then the approximation ratio is 2 by Thm. 3.3.3. The TreeSplit algorithm is applied on these m users. During the algorithm, a total of $1+2+2^2+\ldots+2^{k-1} = 2^k - 1 = m-1$ groups are divided (2^j groups at level j), as is obvious from Fig. 3.1. We represent these $m-1$ groups by the set $\mathcal{K}$. We would like the splitting condition to be obeyed whenever any group $M \in \mathcal{K}$ is divided. To this end, we define an event - that the splitting condition is violated during the division of a group $M \in \mathcal{K}$ - and denote it by A_M. We state the following Lemma.

Lemma 3.3.4 *The probability of occurrence of A_M when the group M under consideration has some d users is*

$$\mathcal{P}(A_M) = 2\sum_{\ell=0}^{n} \binom{n}{\ell} (1-q^{d/2})^{\ell} (q^{d/2})^{n-\ell} (1-q^{d/2})^{\ell}. \tag{3.2}$$

Proof We know that group M has d users. Now consider a division of these d users into two groups M_1 and M_2 of $d/2$ users each. If M_1 and M_2 are such that $F_{M_1} \subseteq F_{M_2}$ or $F_{M_2} \subseteq F_{M_1}$, then the splitting condition is violated and event A_M occurs. Let us assume that $F_{M_1} \subseteq F_{M_2}$. Note that $(1-q^{d/2})^{\ell}(q^{d/2})^{n-\ell}$ is the probability that group M_1 has only some ℓ files and $\binom{n}{\ell}$ is the number of ways of choosing these ℓ files from n files in T. Also, $(1-q^{d/2})^{\ell}$ is the probability that group M_2 also contains those ℓ files (making F_{M_2} a superset). Summing over all possible ℓ, and multiplying by 2 to factor in the equivalent case in which $F_{M_2} \subseteq F_{M_1}$, we get the expression in (3.2). ■

Note that (3.2) can also be written as:

$$\mathcal{P}(A_M) = 2((1-q^{d/2})^2 + q^{d/2})^n = 2(1+q^d-q^{d/2})^n. \tag{3.3}$$

As stated earlier, we want to ensure that the splitting condition is met for all $M \in \mathcal{K}$. When the splitting condition is not met for any group in $\mathcal{K}$, we say that an *error* has occurred. Note that the probability of error is the union of the probabilities of all A_M's, and can be written as follows:

$$\begin{aligned}\mathcal{P}(error) = \mathcal{P}(\bigcup_{M\in\mathcal{K}} A_M) &\leq \sum_{M\in\mathcal{K}} \mathcal{P}(A_M), \\ &= 2\sum_{M\in\mathcal{K}} (1 + q^{|M|} - q^{|M|/2})^n, \end{aligned} \tag{3.4}$$

where $|M|$ represents the number of users in group M. Since the number of terms in the RHS of 3.4 is $|\mathcal{K}| = m - 1 < m$ and $\max_{M\in\mathcal{K}}(1 + q^{|M|} - q^{|M|/2})^n$ occurs at $|M| = m$ for a sufficiently large m, we get

$$\mathcal{P}(error) \leq 2m(1 + q^m - q^{m/2})^n. \tag{3.5}$$

We now enforce a constraint on file acquisition. Note that until now, we have only been concerned with all users obtaining the achievable universe F. But in practice, each user desires the complete universe T. Before discussing how each user can obtain T, we define a *file cover*.

Definition 4 (File Cover) *A group of users G is said to be a file cover if $F_G = T$, the universe of available files.*

Lemma 3.3.5 *A group of m users is a file cover if $nq^m \to 0$ as $n \to \infty$.*

Proof We consider a suitably defined bad event, that the group of m users U is not a file cover ($F_U \neq T$). We call this event D, and note that

$$\mathcal{P}(D) = 1 - (1 - q^m)^n \leq nq^m \tag{3.6}$$

It is clear from (3.6) that $\mathcal{P}(D) \to 0$ as $n \to \infty$ if $nq^m \to 0$, so the set U is a file cover with high probability. ■

Lemma 3.3.6 *If $m = c\log n$, the TreeSplit algorithm has an approximation ratio of 2*

to (3.1) *provided that the value of* q *(or* p*) can be chosen a-posteriori such that* $-2/c < \log q < -1/c$*.*

Proof Note that for $m = c \log n$, using (3.5) and the identity that $1 - x < \exp(-x)$,

$$\begin{aligned} \mathcal{P}(error) &< 2c \log n \exp\left(n(q^{c \log n} - q^{\frac{c \log n}{2}})\right), \\ &= 2c \log n \exp\left((n^{\frac{c \log q}{2}} - 1) \cdot n^{(1 + \frac{c \log q}{2})}\right). \end{aligned} \tag{3.7}$$

The RHS of (3.7) goes to zero as $n \to \infty$ when $\log q > -2/c$, so all random divisions in the TreeSplit algorithm obey the splitting condition if $\log q > -2/c$. Furthermore, from (3.6), we see that for this case,

$$\mathcal{P}(D) \leq nq^{c \log n} \tag{3.8}$$

$\mathcal{P}(D) \to 0$ as $n \to \infty$ provided $\log q < -1/c$. Therefore, for $-2/c < \log q < -1/c$, the TreeSplit algorithm ensures that all users obtain the complete universe T if m is a power of 2, which by Thm. 3.3.3, is a 2-approximation for general m. ■

We now look at a more general regime, in which the number of users grows linearly with the number of files.

3.3.2 Approximation ratio of $1 + \epsilon$ for $m = \mathrm{O}(n)$

For $m = \mathrm{O}(n)$, we propose the *Partition+TreeSplit* algorithm. There are two steps to the algorithm. In step **(i)**, the users are randomly partitioned into a set of groups $\mathcal{G} = \{G_1, G_2, \ldots\}$ where each group $G_i \in \mathcal{G}$ has $w \log n$ users. We would like for all groups $G_i \in \mathcal{G}$ to be file covers. In step **(ii)**, the TreeSplit algorithm with random divisions is applied to some $v \log n$ users in every group $G_i \in \mathcal{G}$, where $v \log n$ is a power of 2 closest to but less than $w \log n$. Let $m = \alpha n$.

Lemma 3.3.7 *If* w*,* v *and* q *are such that* $-2/v < \log q < -2/w$ *and* $v \log n$ *is a power of* 2 *closest to but less than* $w \log n$*, then the Partition+TreeSplit algorithm ensures that at least* $\frac{\alpha n}{w \log n} v \log n$ *users obtain the complete universe* T*.*

Proof In the Partition+TreeSplit algorithm, we would like to avoid errors at both steps of the algorithm. An error E_1 in step **(i)** occurs if any group $G_i \in \mathcal{G}$ is not a file cover. An error E_2 in step **(ii)** occurs if any division in the TreeSplit algorithm does not obey the splitting condition. We are therefore interested in analyzing the probability $\mathcal{P}(E_1 \cup E_2)$, which can be written as:

$$\sum_{G_i \in \mathcal{G}} \mathcal{P}(G_i \text{ is not a file cover}) + \sum_{G_i \in \mathcal{G}} \mathcal{P}(\text{Divisions among } v \log n \text{ users in } G_i \text{ do not obey the splitting condition}). \tag{3.9}$$

Each $G_i \in \mathcal{G}$ has $w \log n = \mathrm{O}(\log n)$ users, for which we have already computed the probability that G_i is not a file cover in (3.8). Similarly, since the term in the second summation involves $v \log n = \mathrm{O}(\log n)$ users, we have computed the probability of an error in division in (3.7). Also note that each summation in (3.9) has $\alpha n / w \log n$ terms, since there are $\alpha n / w \log n$ groups in $\mathcal{G}$. Using (3.7) and (3.8), we see that

$$\mathcal{P}(E_1 \cup E_2) < \frac{\alpha n}{w \log n} \left(n q^{w \log n} + 2v \log n e^{(n^{\frac{v \log q}{2}} - 1) \cdot n^{(1 + \frac{v \log q}{2})}} \right). \tag{3.10}$$

We see that the first term of (3.10) goes to zero when $\log q < -2/w$ and the second goes to zero when $\log q > -2/v$, so the probability of error in the Partition+TreeSplit algorithm goes to zero for q such that $-2/v < \log q < -2/w$.

We now look at how many users can obtain the universe via the Partition+TreeSplit algorithm. We know from (3.10) that if q is such that $\log q < -2/w$, the random partition of m users into the set of groups $\mathcal{G}$ ensures that all groups in $\mathcal{G}$ are file covers with high probability. Each group has $w \log n$ users and $|\mathcal{G}|$ is equal to $\alpha n / w \log n$. Let us now try to prune each group $G_i \in \mathcal{G}$ through the UniquePick algorithm to obtain H_i. H_i can be of 2 types: **(a)** $|H_i| > v \log n$ or **(b)** $|H_i| \leq v \log n$.

If H_i is of type **(a)**, we know by the definition of the UniquePick algorithm that it contains more than $v \log n$ users who contain unique files. So we can apply the Polygon algorithm of Aggarwal *et al.* (2013) to ensure that more than $v \log n$ users in H_i obtain the complete universe T. If H_i is of type **(b)**, add users from $G_i \backslash H_i$ so that $|H_i| = v \log n$. Now, $|H_i| = v \log n$, which is a power of 2 and contains the complete universe. If v and q satisfy $\log q > -2/v$, the TreeSplit algorithm can be applied to

H_i with random divisions to ensure that all $v \log n$ users obtain T. So for both types of groups, at least $v \log n$ users from each group can obtain the complete universe T. Since there are $\alpha n / w \log n$ such groups, we see that at least $\frac{\alpha n}{w \log n} v \log n$ users obtain the complete universe T ■

We now consider two cases in Sections 3.3.2 and 3.3.2.

Case i: q chosen a-priori

Lemma 3.3.8 *When $m = \alpha n$ and q is chosen a-priori, the Partition+TreeSplit algorithm has an approximation ratio of 2.*

Proof We choose w such that $\log q < -2/w$. By Lemma 3.3.7, we know that if we can choose a v such that $v \log n$ is a power of 2 and $\log q > -2/v$, then by the Partition+TreeSplit algorithm, at least $\frac{\alpha n}{w \log n} v \log n$ users obtain the complete universe T. So we choose v such that $v \log n$ is the power of 2 closest to but less than $w \log n$. We also ensure while choosing w that $w \log n$ itself is not a power of 2. Note that $v/w > 1/2$, by definition. Therefore, the fraction of all users who obtain the achievable universe is $\frac{\alpha n}{w \log n} \frac{v \log n}{m} = v/w > 1/2$. ■

Case ii: q can be chosen a-posteriori

Lemma 3.3.9 *When $m = \alpha n$ and q can be chosen a-posteriori, the Partition+TreeSplit algorithm has an approximation ratio of $1 + \epsilon_1$ for some arbitrarily small $\epsilon_1 > 0$.*

Proof We can now choose all of w, v and q as we please such that $-2/v < \log q < -2/w$, and Lemma 3.3.7 assures us that at least $\frac{\alpha n}{w \log n} v \log n$ users obtain the complete universe T. We first choose v such that $v \log n$ is a power of 2. We then choose w arbitrarily close to but greater than v ($w = v + \delta$, say). We then choose q such that $-2/v < \log q < -2/w$, and this choice is possible because of our choice of v and w. Operating with these values of v, w and q, we see that the fraction of users who finally obtain the complete universe T is $\frac{\alpha n}{w \log n} \frac{v \log n}{m} = v/w \approx 1$. ■

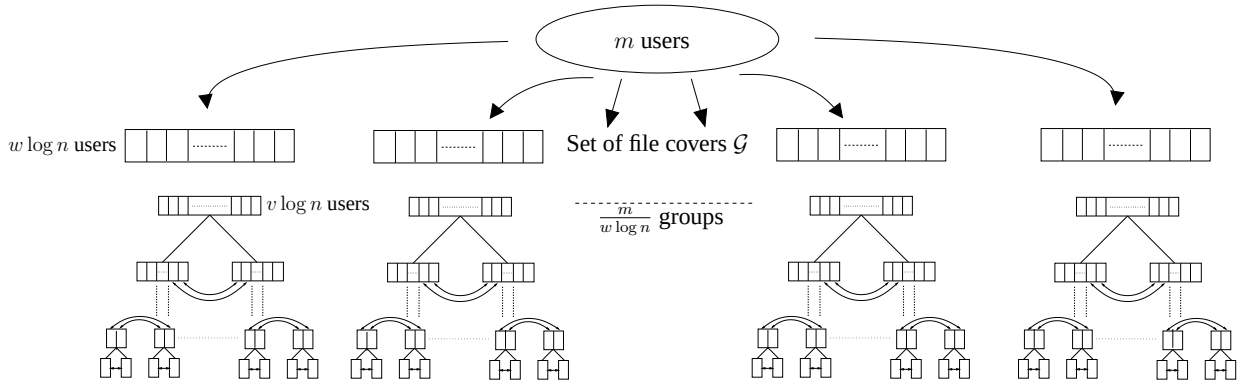

Figure 3.2: Partition+TreeSplit algorithm when $m = \mathrm{O}(n)$ or $\mathrm{O}(n^z)$. The users are first randomly partitioned into a set of groups $\mathcal{G}$. Each group has size $w \log n$, and is a file cover with high probability. $v \log n$ users from each group are then chosen by the UniquePick algorithm and the TreeSplit algorithm is applied on them.

3.3.3 Approximation ratio of $\frac{1+z}{2} + \epsilon$ for $m = \mathrm{O}(n^z)$, $z > 1$

The algorithm for this case is still the Partition+TreeSplit algorithm. We recall that it involves the following: randomly partition the users into groups of size $w \log n$, and then perform the TreeSplit algorithm with random divisions on $v \log n$ users in each group. Consequently, we follow an analysis similar to that of Section 3.3.2.

Lemma 3.3.10 *If $m = \alpha n^z$ and w, v and q are such that $-2/v < \log q < -(1+z)/w$ and $v \log n$ is a power of 2 closest to but less than $w \log n$, then the Partition+TreeSplit algorithm ensures that at least $\frac{\alpha n^z}{w \log n} v \log n$ users obtain the complete universe T.*

Proof As in Section 3.3.2, we want to avoid errors E_1 and E_2 in step **(i)** and step **(ii)** of the algorithm, respectively. In this regime, however, $m = \alpha n^z$. So,

$$\mathcal{P}(E_1 \cup E_2) < \frac{\alpha n^z}{w \log n} \left(n q^{w \log n} + 2v \log n e^{(n^{\frac{v \log q}{2}} - 1) \cdot n^{(1 + \frac{v \log q}{2})}} \right). \tag{3.11}$$

Splitting the RHS of (3.11) into two terms, we see that the first term goes to zero for $\log q < -(1+z)/w$ and that the second term goes to zero for $\log q > -2/v$. So the probability of error goes to zero as $n \to \infty$ when $-2/v < \log q < -(1+z)/w$.

As for the number of users who finally obtain the complete universe T, we follow a logic similar to that of the proof of Lemma 3.3.7. It has been omitted for the sake of brevity. ■

We now consider the two cases as before.

Lemma 3.3.11 *When $m = \alpha n^z$ for $z > 1$, and q is chosen a-priori, the Partition+TreeSplit algorithm has an approximation ratio of $1 + z$.*

Proof We choose w such that $\log q < -(1 + z)/w$. We then choose some v such that $v < 2w/1 + z$ and $v \log n$ is a power of 2. Note that we can always choose v such that $1/1 + z < v/w < 2/1 + z$. We are also assured that $\log q > -2/v$. Following the arguments that we made in Section 3.3.2 but now with Lemma 3.3.10, we can say that at least $\alpha n^z \cdot \frac{v}{w} > \alpha n^z \cdot \frac{1}{1+z}$ users obtain the universe. ■

Lemma 3.3.12 *When $m = \alpha n^z$ and q can be chosen a-posteriori, the Partition+TreeSplit algorithm has an approximation ratio of $\frac{1+z}{2} + \epsilon_2$ for an arbitrarily small $\epsilon_2 > 0$.*

Proof We start by choosing v such that $v \log n$ is a power of 2. We then set $w = \frac{1+z}{2}v + \delta$. Now, we choose q such that $-2/v < \log q < -(1 + z)/w$, which we know is possible. So by Lemma 3.3.10, we can now ensure that at least $\alpha n^z \cdot \frac{v}{w}$ users obtain all files in the universe. Since $v/w \approx 2/1 + z$, the fraction of users who obtain all files is $\frac{1+z}{2} + \epsilon_2$. ■

In the last 3 sections, we have shown that the TreeSplit and Partition+TreeSplit algorithms are effective for a variety of regimes of m and n. In the next section, we show that for all m up to the order of $e^{o(n)}$, it is highly probable that there exists a schedule which ensures that at least half the users obtain the complete universe T, thereby showing how beneficial the Random Sampling acquisition paradigm is.

3.3.4 Existence of a schedule such that at least half the users obtain the universe for any $m = \mathrm{O}\left(e^{o(n)}\right)$

We show the existence of a schedule for any $m = \mathrm{O}\left(e^{o(n)}\right)$ by considering the splitting condition in the TreeSplit algorithm. We consider the division of a particular group G of d users, and define E_G as the event that all possible divisions of G into two equal groups do not obey the splitting condition.

Without loss of generality, let $G = \{u_1, u_2, \ldots, u_d\}$. The total number of ways to divide d users in G into 2 groups with $d/2$ users each is $\binom{d}{d/2}$. Let the two groups of users obtained from the i^{th} division be denoted by G_{1i} and G_{2i}, where $1 \leq i \leq \binom{d}{d/2}$. Let $\mathcal{G}_1 \triangleq \{G_{1i}, 1 \leq i \leq \binom{d}{d/2}\}$ and similarly $\mathcal{G}_2 \triangleq \{G_{2i}, 1 \leq i \leq \binom{d}{d/2}\}$.

Without loss of generality, we assume $F_{G_{2i}} \subseteq F_{G_{1i}} \ \forall \ i$. Note that $F_G = F_{G_{2i}} \cup F_{G_{1i}} = F_{G_{1i}} \ \forall \ i$. Let $F_G = \{1, 2, \ldots, a, a+1, \ldots, a+b\}$, again without loss of generality.

Lemma 3.3.13 *Let $\mathcal{I} \triangleq \bigcap_{M \in \mathcal{G}_1} M$. If E_G occurs, $|\mathcal{I}| = 1$.*

Proof We prove this by contradiction. Case **(i)**: Suppose $|\mathcal{I}| > 1$; then at least two users are constrained to always reside together after division. That does not tally with the fact that we are dividing G in $\binom{d}{d/2}$ ways, which is a contradiction. Case **(ii)**: Suppose $|\mathcal{I}| = 0$, it would imply that $\exists \ G_{1i_1}, G_{1i_2} \in \mathcal{G}_1, s.t \ G_{1i_1} \cap G_{1i_2} = \phi$; in that case, G could have been divided into G_{1i_1} and G_{1i_2} while obeying the splitting condition. So E_G cannot occur.

According to Lemma 3.3.13, there is one user who is responsible for the event E_G, whom we call the *culprit user*. Without loss of generality, let this user be u_1. Let $C^0_{u_1} = \{1, 2, \ldots, a\}$ and define $\mathcal{B} \triangleq F_G \backslash C^0_{u_1} = \{a+1, \ldots, a+b\}$.

Lemma 3.3.14 *If E_G occurs, any group G' of $d/2-1$ users from $\{u_2, u_3, \ldots, u_d\}$ must satisfy $\mathcal{B} \subseteq F_{G'}$.*

Proof Consider any group $G' \subset G$ consisting $d/2 - 1$ users $\{u_{g_1}, u_{g_2}, \ldots, u_{g_{d/2-1}}\}$. These $d/2 - 1$ users along with user u_1 will form a group $G_{1i} \in \mathcal{G}_1$ for some i, since that is one of the possible divisions. Also note that $F_G = F_{G'} \bigcup C^0_{u1}$. Therefore $\{a + 1, a+2, \ldots, a+b\} \subseteq F_{G'}$.

Lemma 3.3.15 *At least $d/2$ users from $\{u_2, u_3, \ldots, u_d\}$ must possess each file $f_i \in \mathcal{B}$ for E_G to occur, and the probability that this happens is less than $(2p)^{\frac{db}{2}}$, where $b = |\mathcal{B}|$.*

Proof We prove the first part of the Lemma by contradiction. Suppose less than $d/2$ users from $\{u_2, u_3, \ldots, u_d\}$ possess some file $f_0 \in \mathcal{B}$. This implies that at least $d/2 - 1$ users from $\{u_2, u_3, \ldots, u_d\}$ do not possess f_0. Without loss of generality, let users $\{u_2, u_3, \ldots, u_{d/2}\}$ not possess f_0. These $d/2 - 1$ users will not satisfy Lemma 3.3.14, which is a contradiction. Thus, each file $f_i \in \mathcal{B}$ must be possessed by at least $d/2$ users from $\{u_2, u_3, \ldots, u_d\}$.

We now come to the second part of the Lemma. Probability that at least $d/2$ users from $\{u_2, u_3, \ldots, u_d\}$ possess a file f_0 is $\binom{d-1}{d/2}p^{d/2}$. Since each file $f_0 \in \mathcal{B}$ is chosen independently, probability that at least $d/2$ users from $\{u_2, u_3, \ldots, u_d\}$ possess all b $(= |\mathcal{B}|)$ files is $\left(\binom{d-1}{d/2}p^{d/2}\right)^b < (2p)^{\frac{db}{2}}$.

Lemma 3.3.16 *$\mathcal{P}(E_G)$ is less than $\left(p + q\big(q' + p'(2p)^{d/2}\big)\right)^n$, where $q' = q^{d-1}$ and $p' = 1 - q'$.*

Proof: Let us denote the event in which the culprit user u_1 has a files and F_G has $a + b$ files by $E_{a,b}$. We know from Lemma 3.3.15 that $\mathcal{P}(E_G|E_{a,b}) < (2p)^{\frac{db}{2}}$. But

$$\mathcal{P}(E_G) = \sum_{a=0}^{n}\sum_{b=0}^{n-a}\mathcal{P}(E_G|E_{a,b})\mathcal{P}(E_{a,b}). \tag{3.12}$$

Note that the probability that $d-1$ users do not possess a file is q^{d-1}, denoted by q', and

$$\mathcal{P}(E_{a,b}) = \binom{n}{a}p^a q^{n-a} \times \binom{n-a}{b}(p')^b(q')^{n-a-b}, \tag{3.13}$$

Therefore, from (3.12) and (3.13),

$$\begin{aligned}
\mathcal{P}(E_G) &< \sum_{a=0}^{a=n}\sum_{b=0}^{b=n-a}\binom{n}{a}p^a q^{n-a}\binom{n-a}{b}(p')^b(q')^{n-a-b}(2p)^{db/2}, \\
&= \sum_{a=0}^{a=n}\binom{n}{a}p^a q^{n-a}\big(q' + p'(2p)^{d/2}\big)^{n-a}, \\
&= \Big(p + q\big(q' + p'(2p)^{d/2}\big)\Big)^n. \qquad \blacksquare
\end{aligned} \tag{3.14}$$

Theorem 3.3.17 *For $m = \mathrm{O}\left(\exp(n^{1-\epsilon_3})\right)$ ($\epsilon_3 > 0$) and $p < 1/2$, there exist divisions obeying the splitting condition for all groups in the TreeSplit algorithm as $n \to \infty$.*

Proof Let $\mathcal{K}$ represent the set of $m-1$ groups which are divided in the TreeSplit algorithm. We say that an *error* occurs if $\exists\ G' \in \mathcal{K}$, for which all divisions do not satisfy splitting condition.

$$\mathcal{P}(error) \leq \sum_{G' \in \mathcal{K}} \mathcal{P}(E_{G'}) \leq |\mathcal{K}| \Big(\max_{G' \in \mathcal{K}} \{\mathcal{P}(E_{G'})\} \Big) \tag{3.15}$$

From equation (3.14), it is easy to see that $\max\{\mathcal{P}(E_{G'})\}$ occurs G' has minimum number of users, i.e. $d = 2$, and is equal to $\Big(p + q\big(q + 2p^2\big)\Big)^n$. Also, $|\mathcal{K}| = m - 1 < m$. Therefore, from equation (3.15), we obtain

$$\mathcal{P}(error) \leq \mathrm{O}\left(\exp(n^{1-\epsilon})\right) \left(p + q(1 - p + 2p^2)\right)^n \tag{3.16}$$

For $p < 1/2$, the RHS of equation (3.16) tends to zero as $n \to \infty$. This implies that $\forall\ G' \in \mathcal{K}$ at least one division exists which satisfies the splitting condition.

So the TreeSplit algorithm with some method of division exists with high probability even for $m = \mathrm{O}\left(e^{o(n)}\right)$, and provides a schedule through which at least half the total number of users obtain the complete universe.

3.4 Simulations

We ran simulations to verify our result from Section 3.3.4. We considered large m when compared to n. We divided users as in the TreeSplit algorithm, and noted an error whenever any one group in the tree was such that all possible divisions violated the splitting condition. Shown in Fig. 3.3 is a plot of the lowest value of n for a given m for which the percentage of error cases was less than 1%. Notice that for $p = 0.10$, even if there are 2^{15} users in the system and only 160 files, 99% of cases are such that all divisions obey the splitting condition, showing that it is very probable that there exists a schedule through which 2^{14} users obtain all 160 files even though each user only starts out with around $160 \times 0.10 = 16$ files.

3.5 Conclusions and Future Work

We presented a random file acquisition paradigm and showed constant approximation algorithms for a variety of regimes under the give-and-take file-exchange protocol. We also showed that it is beneficial for users to acquire files using the Random Sampling file acquisition paradigm (with small p, to keep cost low) by showing that there exists a schedule that ensures at least half the users obtain the universe with high probability. Future work in this area could involve the study of other file-acquisition paradigms and comparisons of their performance with respect to the give-and-take protocol. It could also involve an interesting game theoretic question thrown up by the GT criterion, as to what are the minimum number of files each node should download from the server to ensure that it finally gets all the files that it desires.

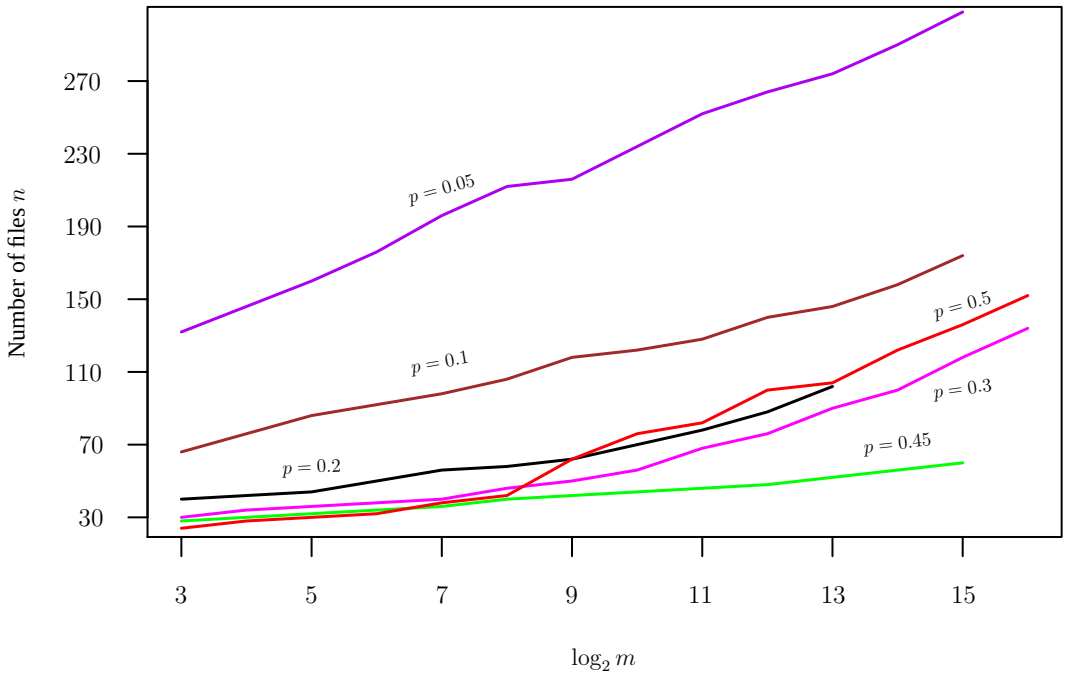

Figure 3.3: Plot for different values of p of the minimum value of n for which 99% of cases are such that there exist divisions for all groups, with $\log_2 m$.

CHAPTER 4

Online algorithms for basestation allocation of users with fixed data demands and arbitrary arrival times

We have had the least success with this problem among the three we have considered. Most of these results are exploratory in nature.

4.1 Introduction

In this chapter, we consider the problem of assigning users to basestations, online, in order to satisfy some objective function. This extends the work of Thangaraj and Vaze (2013) in the context of basestation allocation.

The problem setup of Thangaraj and Vaze (2013) is as follows. There are n users and m basestations. Every user-basestation pair i, j has a particular weight w_{ij} associated with it, which signifies the rate of data download that the user i can achieve when connected to basestation j. After some assignment of users to basestations, if a basestation j has a set of users U_j connected to it, where $|U_j| = d_j$, then the rate achieved by each user i in U_j is equal to w_{ij}/d_j. This represents the *processor sharing model*. In Thangaraj and Vaze (2013), the objective function that was considered was sum-rate maximization, i.e., they were interested in maximizing the sum of rates achieved by all users after they have been allocated to basestations, keeping in mind the processor sharing model. They wanted to characterize the *online* case, in which users arrive in a particular order, and they must be assigned to basestations irrevocably without any knowledge of future arrivals.

The main difference between the work in Thangaraj and Vaze (2013) and previous load balancing work Azar *et al.* (1993); Azar (1998); Caragiannis *et al.* (2006) was in the processor sharing model (load balancing does not have sharing of resources at the

basestation) and the corresponding objective function. As in most analysis of online algorithms, Thangaraj and Vaze (2013) was concerned with providing competitive ratio analyses. The competitive ratio of an online algorithm is a measure of its performance. The competitive ratio (in a maximization problem) can be defined for both the worst case and the average case. In the former case, it represents the maximum ratio of the objective functions of the optimal offline and online algorithms for any incoming order of the users. The average case, on the other hand, takes a mean of all ratios - one for each incoming order of users.

A competitive ratio that grows with the parameters of the problem (m, n in this case) is considered undesirable; an algorithm is *competitive* if its competitive ratio can be shown to be a constant. As it turns out, for the problem in Thangaraj and Vaze (2013), the worst case competitive ratio scales with n for any online algorithm, so they analysed the problem for the average case competitive ratio. They provided an online algorithm based on the k-secretary problem that had an average competitive ratio of 8, for arbitrary weights of users to basestations.

The objective function of Thangaraj and Vaze (2013), however is impractical, because it assumes that the users, once assigned, will continue in the system forever (hence the maximization of sum rate). In practice, however, users arrive and exit the system, and it makes more practical sense to address this question with perhaps another, more relevant objective function.

In this report, we consider models that try and incorporate this very notion, and analyse their feasibility. We then move on to offline versus online algorithms and propose some such algorithms. As pointed out before, the nature of this work is mainly exploratory.

4.2 Preliminaries

The notation is the same as stated in Section 4.1. There are m basestations, represented by the set $B = \{b_1, b_2, \ldots, b_n\}$; n users represented by the set $U = \{u_1, u_2, \ldots, u_n\}$, and a weight matrix W with entries w_{ij}. To incorporate the queuing, we first explore a

trivial extension of the problem considered in Thangaraj and Vaze (2013).

4.2.1 A trivial extension of Thangaraj and Vaze (2013)

- The users arrive in a queue and depart when they have been served by the system. Define each arrival or departure as an *event*.
- Time is slotted as follows. One time slot is set to the smallest duration between two events. Since we are interested in an online algorithm and the time of occurrence of the events is not known a-priori, we set one time slot to an arbitrarily small duration. This slot duration could also be an additional input to the online algorithm - an estimation of the smallest duration between events under heaviest traffic. By definition, events can only occur either at the beginning or end of a slot, and not during a slot. Also, a maximum of one user can arrive at the beginning of any slot.
- Users spend some *service time* in the system. The arrival time (slot number) of user u_i is represented by A_i and his service time by S_i.
- Our objective is to assign users irrevocably to basestations in an online fashion so as to maximize the average rate supported by the system. If P_ℓ is the total rate output of the system (all users at all basestations) in slot ℓ, and we consider some L slots in all, our objective function is represented by $\max \sum_{\ell=1}^{L} P_\ell / L$.

Two Basestation Model with simplistic queuing

- Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$.
- Let $\mathcal{B} = \{b_1, b_2\}$.
- Let $w_{i1} = w_{i2} \; \forall \; i \in \{1, 2, \ldots, n\}$.
- Let $A_i = i$ and let $S_i = A_i + k$ for all users u_i. So a user arrives in every slot and stays for k slots. Note that this implies that there are a total of n slots to be averaged over.

Offline Algorithm

We know that the good algorithms in the 2 basestation setting assign the “best" users to one basestation and all the others to the other basestation. In the same spirit, we propose the following optimal offline algorithm to our problem.

Let $\mathcal{U}_g = \{u_i : u_i \in \mathcal{U}$ and $\forall u_j, u_k \in \mathcal{U}_g$, $|j - k| > k$, $\sum_{u_i \in \mathcal{U}_g} w_{i1}$ is maximum.$\}$. Let $\mathcal{U}_b = \mathcal{U} \setminus \mathcal{U}_g$.

The algorithm itself is simple: assign all users in $\mathcal{U}_g$ to b_1 and all users in $\mathcal{U}_b$ to b_2. Note that the users in b_2 will contribute a very small fraction of the total utility. If the average rate supported by the offline algorithm is represented by R_{off}, we see that:

$$R_{off} \geq \sum_{u_i \in \mathcal{U}_g} w_{i1} \tag{4.1}$$

Online Algorithm

For this, let us split the users into contiguous *blocks* of k users each. $U_1 = \{u_1, u_2, \ldots, u_k\}$, $U_2 = \{u_{k+1}, u_{k+2}, \ldots, u_{2k}\}$ and so on. We now pick the best user in each block by the secretary algorithm and assign him to b_1. All the other users are assigned to b_2. Note the following about this algorithm:

- There are a maximum of two users assigned to b_1 at any time.
- $A = \sum_j \max_{u_i \in U_j} w_{i1}$ is an upper bound on R_{off}. The secretary algorithm's competitive ratio will be with respect to A, and because $A \geq R_{off}$, the ratio can be extended to our algorithm.
- The competitive ratio of the overall online algorithm over permutations of blocks and then of users within each block is therefore given by $2 \cdot (n/k) \cdot c_{sec}$, where c_{sec} is the competitive ratio of the secretary algorithm.

Note that this trivial extension that incorporates queuing provides a competitive ratio that scales with n, and so the corresponding online algorithm is not competitive. Therefore, we propose another practically feasible model.

4.2.2 A New Model

We now consider the following model:

Definition 5 *Users come in as before, in a queue. Arrival times are arbitrary. Each user u_i now has a constant data demand d_i. The user is irrevocably assigned to a basestation when he arrives, he achieves a certain rate at the basestation (which varies as other users enter and leave that basestation), and leaves when he has fulfilled his data demand. Our objective is allocate users so as to* minimize the maximum time *spent by any user in the system.*

Assumption 4.2.1 *Inter-arrival times follow an exponential distribution. So given that a user has arrived at time $t = 0$, the time of arrival of the next user follows the probability distribution $p(t) = \lambda e^{-\lambda t}$. Inter-arrival times are independent for all pairs of consecutive users.*

Our objective with Definition 5 and Assumption 4.2.1 is to create a stable queuing model that is amenable to analysis. However, we show in the next section that because of our processor sharing constraint, the stability conditions become way too stringent, and so analyzing an infinite queue is useless.

4.3 Unstable queues!

Consider the problem posed by definition 5 and Assumption 4.2.1. Let the users all have a constant data demand $d = 1$. Also, let $w_{ij} = w \; \forall \; i, j$. This is the trivial case of identical users and identical basestations. Note that an online vs. offline analysis is trivial in this case, since the online algorithm knows exactly what the offline algorithm does (since users are identical). The competitive ratio is therefore 1, and will be achieved by a simple round-robin algorithm, which assigns user u_i to basestation $b_{i \mod m}$. We now analyze how the queue at a particular basestation evolves due to this algorithm.

Let us limit ourselves without loss of generality to b_1. Let us denote the users at b_1 by u_{i_1}, u_{i_2} and so on. Let the arrival time of user u_{i_k} be t_k and let his departure time be T_k.

The departure time of the first user T_1 is now a random variable, and can be expressed using the equation:

$$w\frac{t_2 - t_1}{1} + w\frac{t_3 - t_2}{2} + w\frac{t_4 - t_3}{3} + \ldots + w\frac{t_k - t_{k-1}}{|s|_{T_1}} \geq 1 \tag{4.2}$$

where $|s|_{T_k}$ represents the number of users in the system (at that basestation) when user k departs. The above equation is simply one which is written based on the constraint on data demand.

In expectation, equation (4.2) becomes

$$w\lambda m\Big(\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{T_1/\lambda m}\Big)\geq 1 \tag{4.3}$$

The LHS goes as a $\log$, and it is easy to show write a similar expression for a general user u_{i_p}, with some gain terms due to the fact that the first few users leave. We will gloss over the details, but is easy to see that if some $O(n^\epsilon)$th user exists in the system before u_{i_1} leaves, the queue will become unstable, and the objective function we want to analyse will go to infinity.

So basically, the processor sharing model does not allow for stable queues. In the next section, will discuss another processor sharing model in which the stability of the queue can be preserved.

4.4 Another Variant to ensure queue stability

Let us now consider a different model by which service at the basestation happens.

Definition 6 *We will assume a broadcast channel at the basestation, which broadcasts the files that users need on the channel. Note that if all users at the basestation require the same file, then the basestation will be able to serve them all at their respective weights (without division). In general, the rate achievable by user u_i when connected to basestation b_j is $w_{ij}/|f|_j$, where $|f|_j$ represents the number of distinct files being demanded at b_j. Users still have a constant data demand as in Definition 5, only the processor sharing model has changed.*

Note that all basestations can serve the files that are being demanded by all users. It is trivial to see that this case reduces to the previous case when all users come in demanding unique files. But what if the demand of files followed some distribution, by which some files were more popular than others? Wouldn't this ensure that the number of distinct files being served at a basestation reduced? We will see that in this section.

Let the probability distribution on the demand of files be one in which picking up file i is represented by $p(i)$ and $1-p(i)=q(i)$. Note that users sample from this

distribution independently. Let us denote using the indicator random variable I_i^k the event that file i was picked up by k users. Let there be a total of F files. We can now write an expression for the random variable X^k, the number of distinct files at a basestation when there are k users assigned to it as

$$X^k = \sum_{j=1}^{F} I_j^k. \tag{4.4}$$

By the linearity of expectation,

$$E[X^k] = \sum_{j=1}^{F} E[I_j^k]. \tag{4.5}$$

Now, if the assignments to basestation happen independently of the files acquired by the users (bad algorithm), then

$$E[X^k] = \sum_{j=1}^{F} 1 - q(j)^k. \tag{4.6}$$

Now as it turns out, using the file distribution $p(i) = 2^{-i}$ with an infinite number of files returns $X^k = O(\log k)$ for the independent allocation algorithm. Note that $E[X^k]$ was k in our previous processor sharing model. This new model can keep the queue stable.

Another thing to be observed is that the files that users acquire introduces another dimension to the problem concerning online algorithms, because of the ordering of these files. Even the trivial case of equal weights and data demands is not immediately obvious, and needs an intelligent online algorithm.

4.5 Some online algorithms

We propose an algorithms for the problem in Definition 5 with arbitrary user arrival times (without Assumption 4.2.1), and one for Definition 6.

Algorithm 1: Let the users arrive at arbitrary times. At the arrival of user u_i, assume

that he is allocated to basestation b_j and evaluate the objective function (minmax service time) for all the users at that basestation. Assign the user to that basestation in which the objective function is minimum. This is a sort of globally greedy algorithm. Based on the fact that we have not been able to produce a bad case, we conjecture that in the average case,

Conjecture 4.5.1 *Algorithm 1 is within a constant factor of the offline performance for the problem corresponding to definition 5.*

It also seems intuitive that a combination of Algorithm 1 and assigning users to a basestation already serving the file they demand ought to be optimal for the problem corresponding to definition 6. In keeping with the tone of the rest of this chapter however, we do not have analytical proofs to support this conjecture, and this is merely to encourage future work in a variety of possible directions.

4.6 Conclusion

We analysed the problem of online basestation allocation and introduced a few models to take care of the queuing and departure of users. We proposed online algorithms that we conjectured to be optimal for these models. We also looked at how these models influence queue stability, and found that queues are usually unstable in the case of processor sharing as defined in Thangaraj and Vaze (2013). We therefore proposed a new mode of processor sharing that keeps the queues stable for certain conditions. This new sharing model also throws up its own questions of online versus offline performance, and we proposed another algorithm for this case as well. This problem still leaves a lot to be explored in the context of competitive ratios.

REFERENCES

1. **Aggarwal, S.**, **J. Kuri**, and **R. Vaze** (2013). Social optimum in social groups with give-and-take criterion. *arXiv preprint arXiv:1308.1911*.

2. **Ahn, N.** and **S. Park** (2011). A new mathematical formulation and a heuristic for the maximum disjoint set covers problem to improve the lifetime of the wireless sensor network. *Ad Hoc & Sensor Wireless Networks*, **13**(3-4), 209–225.

3. **Anagnostopoulos, A.**, **F. Grandoni**, **S. Leonardi**, and **A. Wiese**, Constant integrality gap lp formulations of unsplittable flow on a path. *In Integer Programming and Combinatorial Optimization*. Springer, 2013, 25–36.

4. **Azar, Y.**, On-line load balancing. *In Online Algorithms*. Springer, 1998, 178–195.

5. **Azar, Y.**, **B. Kalyanasundaram**, **S. Plotkin**, **K. R. Pruhs**, and **O. Waarts**, Online load balancing of temporary tasks. *In Algorithms and Data Structures*. Springer, 1993, 119–130.

6. **Bagaria, V. K.**, **A. Pananjady**, and **R. Vaze** (2013). Optimally approximating the lifetime of wireless sensor networks. *arXiv preprint arXiv:1307.5230*.

7. **Berman, P.**, **G. Calinescu**, **C. Shah**, and **A. Zelikovsky**, Power efficient monitoring management in sensor networks. *In IEEE WCNC. 2004 Wireless Communications and Networking Conference, 2004.*, volume 4. IEEE, 2004.

8. **Bollobás, B.**, **D. Pritchard**, **T. Rothvoß**, and **A. Scott** (2013). Cover-decomposition and polychromatic numbers. *SIAM Journal on Discrete Mathematics*, **27**(1), 240–256.

9. **Caragiannis, I.**, **M. Flammini**, **C. Kaklamanis**, **P. Kanellopoulos**, and **L. Moscardelli**, Tight bounds for selfish and greedy load balancing. *In Automata, Languages and Programming*. Springer, 2006, 311–322.

10. **Cardei, M.** and **D.-Z. Du** (2005). Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, **11**(3), 333–340.

11. **Cardei, M.**, **M. T. Thai**, **Y. Li**, and **W. Wu**, Energy-efficient target coverage in wireless sensor networks. *In Proceedings of IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3. IEEE, 2005.

12. **Cardei, M.** and **J. Wu** (2006). Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Computer communications*, **29**(4), 413–420.

13. **Ding, L.**, **W. Wu**, **J. Willson**, **L. Wu**, **Z. Lu**, and **W. Lee**, Constant-approximation for target coverage problem in wireless sensor networks. *In Proceedings of IEEE INFOCOM, 2012*. IEEE, 2012.

14. **Dinger, J.** and **H. Hartenstein**, Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. *In Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*. IEEE, 2006.

15. **Feldman, M.** and **J. Chuang** (2005). Overcoming free-riding behavior in peer-to-peer systems. *ACM SIGecom Exchanges*, **5**(4), 41–50.

16. **Feldman, M.**, **C. Papadimitriou**, **J. Chuang**, and **I. Stoica** (2006). Free-riding and whitewashing in peer-to-peer systems. *Selected Areas in Communications, IEEE Journal on*, **24**(5), 1010–1019.

17. **Fox, G.** (2001). Peer-to-peer networks. *Computing in Science & Engineering*, **3**(3), 75–77.

18. **Karakaya, M.**, **I. Korpeoglu**, and **O. Ulusoy** (2009). Free riding in peer-to-peer networks. *Internet Computing, IEEE*, **13**(2), 92–98.

19. **Kasbekar, G. S.**, **Y. Bejerano**, and **S. Sarkar** (2011). Lifetime and coverage guarantees through distributed coordinate-free sensor activation. *Networking, IEEE/ACM Transactions on*, **19**(2), 470–483.

20. **Knoke, D.** and **S. Yang**, *Social network analysis*, volume 154. Sage, 2008.

21. **Lai, C.-C.**, **C.-K. Ting**, and **R.-S. Ko**, An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications. *In IEEE Congress on Evolutionary Computation, 2007. CEC 2007.*. IEEE, 2007.

22. **Locher, T.**, **P. Moor**, **S. Schmid**, and **R. Wattenhofer**, Free riding in bittorrent is cheap. *In Proc. Workshop on Hot Topics in Networks (HotNets)*. Citeseer, 2006.

23. **Mol, J. J.-D.**, **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.

24. **Nishida, H.** and **T. Nguyen** (2010). A global contribution approach to maintain fairness in p2p networks. *Parallel and Distributed Systems, IEEE Transactions on*, **21**(6), 812–826.

25. **Pananjady, A.**, **V. K. Bagaria**, and **R. Vaze**, Maximizing utility among selfish users in social groups. *In National Conference on Communications (NCC), 2010*. IEEE, 2014.

26. **Pritchard, D.** (2010). An lp with integrality gap 1+ epsilon for multidimensional knapsack. *arXiv preprint arXiv:1005.3324*.

27. **Pyun, S.-Y.** and **D.-H. Cho** (2009). Power-saving scheduling for multiple-target coverage in wireless sensor networks. *Communications Letters, IEEE*, **13**(2), 130–132.

28. **Rahman, R.**, **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*. IEEE, 2010.

29. **Slijepcevic, S.** and **M. Potkonjak**, Power efficient organization of wireless sensor networks. *In IEEE International Conference on Communications, 2001. ICC 2001.*, volume 2. IEEE, 2001.

30. **Thangaraj, A.** and **R. Vaze** (2013). Online algorithms for basestation allocation. *arXiv preprint arXiv:1308.1212*.

31. **Zhao, Q.** and **M. Gurusamy** (2008). Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, **16**(6), 1378–1391.