TOFU: Semi-Truthful Online Frequency Allocation Mechanism for Wireless Networks

Ping Xu*, *Student Member, IEEE,* Xiang-Yang Li†,*, *Senior Member, IEEE,*
†Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China
*Department of Computer Science, Illinois Institute of Technology, USA
Emails: pxu3@iit.edu, xli@cs.iit.edu.

Abstract—In wireless networks, we need to allocate spectrum efficiently. One challenge is that the spectrum usage requests often come in an online fashion. The second challenge is that the secondary users in a cognitive radio network are often selfish and prefer to maximize their own benefits. In this paper, we address these two challenges by proposing TOFU, semi-truthful online frequency allocation method for wireless networks when primary users can sublease the spectrums to secondary users. In our protocol, secondary users are required to submit the spectrum bid α time-slots before its usage. Upon receiving an online spectrum request, our protocol will decide whether to grant its exclusive usage or not, within at least γ timeslots of requests' arrival. We assume that existing spectrum usage can be preempted with some compensation. For various possible known information, we analytically prove that the competitive ratios of our methods are within small constant factors of the optimum online method. Furthermore, in our mechanisms, no selfish users will gain benefits by bidding lower than its willing payment. Our extensive simulation results show that they perform almost optimum: our methods get a total profit that is more than 95% of the offline optimum when γ is about the duration of spectrum usage Δ.

Index Terms—Wireless networks, spectrum, online allocation, preemption, penalty, competitive ratio.

1 Introduction

The current fixed spectrum allocation scheme leads to significant spectrum *white spaces*. Several new spectrum management models [19] have been proposed to improve the spectrum usage. One promising technology is the opportunistic spectrum usage. Subleasing is widely regarded as another potential way to share spectrum, if we ensure that primary users' spectrum usage is not interfered with. In this work, we assume that there is a set of n secondary nodes V that will share a spectrum channel. Each secondary user v_i may wish to pay (by leasing) for the usage of the available spectrum.

- *The research of authors are partially supported by NSF CNS-0832120, National Natural Science Foundation of China under Grant No. 60828003, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), National Basic Research Program of China (973 Program) under grant No. 2010CB328100. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of author(s) and do not necessarily reflect the views of the funding agencies (NSF, and NSFC). XiangYang Li is the contact author.*
- *The preliminary version of this paper appeared in ISAAC 2009 [22].*

Previous studies on spectrum assignment (*e.g.*, [16], [28], [29]) often assume that the information of all spectrum requests is known before allocation is made. However, the spectrum usage requests often arrive online and the central authority (typically a primary user) needs to *quickly* decide whether the requests are granted or not. Here we assume that upon receiving an online spectrum request, our protocol need decide whether to grant its exclusive usage or not, within at least γ timeslots. The rejection typically could not be revoked. Two different approaches of granting the request could be used: *preemptive* and *non-preemptive*. In this work, we consider preemptive requests where the central authority could potentially terminate the current running request(s) to satisfy this new request to potentially make more profits. Assume that requests terminated cannot be restarted or resumed later (so it is also called "abortion"). Terminating a running request $\mathbf{e}_i$, the central authority has to pay a penalty that is $\beta > 0$ times of the amount paid for the remaining unserved timeslots.

Without any known constraint on requests, we show that the competitive ratio of *any* online spectrum allocation method can be arbitrarily bad when preemption is not allowed. This is because any online method has to accept the first coming request (it is possible that no other requests come), therefore miss the following conflicting requests with an arbitrarily large value. In this work, we focus on the case when we know that the maximum time requirement is Δ time slots. We always assume that every request arrives at the beginning of a time slot and the number of time slots requested is an integer $t \in [1, \Delta]$. To the best of our knowledge, we are the first to study online spectrum allocation with cancelation and preemption penalty.

The main contributions of this paper are as follows. We find that the best competitive ratio achievable depends on β and there are three regimes here. We design several efficient online spectrum allocation methods, called TOFU, that perform almost optimum in all cases. When $0 \leq \beta < 1$, our protocol has a constant competitive ratio depending on β. When $\beta > 1$, we show that the competitive ratio of any online method is at most $O(\frac{\gamma+1}{\Delta})$. We then show that our protocol TOFU achieves the bound $\Omega(\frac{\gamma+1}{\Delta})$. When $\beta = 1$, we show that the competitive ratio of any online algorithm is at most $O((\frac{\gamma+1}{\Delta})^{1/2})$. We then show that our online method TOFU has competitive ratios that match the bounds asymptotically, *i.e.*, $\Theta((\frac{\gamma+1}{\Delta})^{1/2})$. Our extensive simulations show that our methods have competitive

ratios almost 95% in most cases, even compared with the optimum offline spectrum allocation method that knows all future inputs. We also extend our protocol to a more general case in which the secondary users are distritbuted arbitrarily and the required service region are heterogeneous disks. We show that the same asymptotic lower bounds and upper bounds apply to this general model.

We also design efficient auction mechanism for spectrum allocation when secondary users are selfish. In such a mechanism, each user will be charged an amount, say $\mathbf{p}_i$, if its request $\mathbf{e}_i$ is granted. We show that in our online mechanism, to maximize its profit, no secondary user will bid lower than its actual valuation. However, a user may have a chance to improve its profit by bidding higher than its actual valuation. We call this property as *semi-truthful*. Recall that a protocol is called *truthful* in the literature if no user can lie about its bid to improve its profit. Our methods still achieve similar results when the requested region of each user is a sectoral cell.

The rest of the paper is organized as follows. In Section 2, we define in detail the problems to be studied. In Section 3, we present upper bounds on the competitive ratios of any online allocation methods. Then we present our solutions in Section 4 and analytically prove the performance bounds of our methods. We extend our methods to networks with general conflict graphs in Section 5. We present our mechanisms to deal with selfish users in Section 6. Our simulation studies are reported in Section 7. We review the related work in Section 8 and conclude the paper in Section 9.

2 System Model, Problem Formulation

2.1 Network and System Models

We consider a wireless network consisting of some primary users who hold the right of some spectrum channels for subleasing. There is a central authority who decides the spectrum assignment on behalf of these primary users. In other words, we assume that each primary user will trust the central authority and is satisfied with what s/he will get from the auction based on the mechanism designed. If each primary user has an asking price for its spectrums, we need to design mechanisms (such as double auction in which buyers enter competitive bidders and sellers enter competitive offers simultaneously) that also take into account the selfish behavior of primary users. We leave this as a future work, and note that Wang *et al.* [20] proposed some truthful double-auction mechanisms with known distributions on bids. The wireless network also consists of some secondary users $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$. Those secondary users may lease the spectrum usage in some region for a time period at any time. In this paper, we assume a simple scenario where there is only one channel available. We leave it as a future work to design allocation and auction schemes when multiple channels are available and a secondary user may request for a number of channels at same time.

Secondary users may reside at different geometry locations. Here we assume that each request is associated with a disk region. Whether a secondary user's request of channels conflict with the request of another secondary user depends on their locations, in addition to the required time period. This location-dependent conflict will be modeled by a conflict graph $H = (\mathcal{V}, E)$, where two nodes v_i and v_j form an edge (v_i, v_j) in H if and only if they cannot use same channel simultaneously. We will first study the networks with a complete conflict graph, and then extend our methods to cases when requested disks are of arbitrary sizes. We will show that our methods have asymptotically optimum performance guarantees in both cases.

2.2 Problem Formulation

Assume that secondary users $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ will ask for spectrum usage on the fly. A user could ask for spectrum usage at different time-slots. Let $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \cdots$, be the sequence of all requests. Each request $\mathbf{e}_i = (v, b_i, a_i, s_i, t_i)$ is claimed by a secondary user v at time a_i, who bids b_i for the usage of the channel from time s_i to time $s_i + t_i$. Here we also omit the region requirement since we first address networks with complete conflict graph. For most of our discussions we will omit the user v when it is clear from the context. In other words, $\mathbf{e}_i = (b_i, a_i, s_i, t_i)$ denotes the ith request. Obviously, $a_i \leq s_i$ for all requests, which means only spectrum usage in future can be requested. Assume time is slotted, and time requirement t_i must be an integer. We also assume that every user will ask for the spectrum usage for at most Δ timeslots ($t_i \leq \Delta$ for all i). We call Δ the *time bound*.

Since the objective for spectrum auction/lease is to improve spectrum utilization and improve the revenue from the spectrum auction/lease, the central authority (*i.e.*, spectrum auctioneer) can post a requirement that all requests must be submitted in advance of a certain time slots. In this paper, we assume that for every request $\mathbf{e}_i$, $s_i - a_i \geq \alpha$ for a value α given by the auctioneer. Hereafter, we call α the *advance factor*. Each request is claimed at least α time slots before its required usage. Intuitively, a larger advance factor α will give more advantage to the central authority. We later will show that the asymptotic performances of our methods do not depend on α as long as $\alpha \geq \gamma$, the delay factor.

The advance factor puts some constraints on the secondary users on when they should submit their bids. To be fair, our system will also put some condition on the central authority about when the central authority should make a decision on each request. In this work, we always require that the central authority should make a decision on a request within γ time slots of its arrival time. We call γ *delay factor* in the spectrum allocation problem. Obviously, we need $\gamma \leq \alpha$ to make the system meaningful. When $\gamma = 0$, central authority has to make decision *immediately* on request $\mathbf{e}_i$ at time a_i. When $\gamma = \alpha$, central authority could make decision as late as possible. If a request has been rejected, it will never be reconsidered and accepted later. Figure 1 illustrates our model of advance and delay factor. For each request, its arrival time a_i must be before time t_1 in the figure, and central authority must make a decision before time t_2 in the figure.

Since central authority are not able to estimate accurate price for spectrum usage, our model lets the central authority accept a reservation in advance, and lets the secondary user get a reasonable guarantee. Crucially, to improve the spectrum

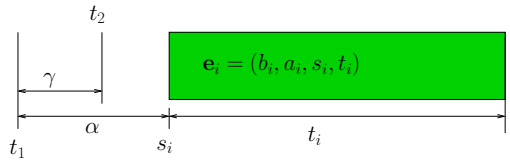


Fig. 1. Advance factor α, and delay factor γ.

usage and revenue, we also let the central authority terminate a reserved/running request at a later time for new request with higher priority at any time. Cancelation and preemption is necessary to take advantage of a spike in demand and rising prices for spectrum channels and not be forced to sell the spectrum below the market because of an a priori contract. When current spectrum usage is terminated, penalty should be paid to compensate the preemption. Here we assume that the penalty $\mu(b, \ell, t)$ is a linear function of the unfinished time $\ell \leq t$ of that request $\mathbf{e}(v, b, a, s, t)$, *i.e.*, $\mu(b, \ell, t) = \beta \frac{\ell}{t} b$ for a constant $\beta \geq 0$. Notice that the cancelation is modeled as $\ell = t$ since the spectrum usage was not started at all. Here the constant $\beta \geq 0$ is the *penalty factor*. We also summarize the symbols used in this paper in Table 1.

TABLE 1
Symbols used in this paper.

Symb.	Meaning	Symb.	Meaning
$\mathbf{e}_i$	request	b_i	bid value
a_i	arrival time	s_i	start time
t_i	required service time	ℓ_i	unfinished service time
Δ	time bound	α	advance factor
γ	delay factor	β	penalty factor

As we stated before, all requests arrive on the fly, thus any information about the future, *e.g.*, the distribution of future bids, the arrival times, the start times and the required time durations are all unknown. The objective of the spectrum allocation is to find an allocation which maximizes the total net profit, *i.e.*, the total payment collected minus the total penalties caused by preemption and cancelation.

To evaluate the performance of an online algorithm, we usually compare it with an optimal *offline* algorithm. The following lemma indicates the hardness of the offline version of our problem.

Lemma 1: Even knowing all requests, it is NP-hard to find an optimal spectrum allocation that maximizes the total bids of admitted requests.

Proof: Observe that the maximum weighted independent set (MWIS) problem in geometry graph is a special case of the spectrum allocation problem: we assume that all spectrum requests are for the same time duration, and the graph for MWIS problem is the conflict graph H, and the weight of a node is the bid value of the spectrum request by this node. Recall that MWIS in geometry graph itself is an NP-complete problem [15] . Thus, the lemma follows. □

For all online spectrum allocation problems studied here, we will design online algorithms with constant competitive ratios if advance factor is chosen carefully.

The performance of an online algorithm $\mathcal{A}$ for spectrum allocation is measured by its *competitive ratio* $\varrho(\mathcal{A}) = \min_{\mathcal{I}} \frac{\mathcal{A}(\mathcal{I})}{\mathrm{OPT}(\mathcal{I})}$, where $\mathcal{I}$ is *any* possible sequence of requests arrival, $\mathcal{A}(\mathcal{I})$ is the profit produced by *online algorithm* $\mathcal{A}$ on input $\mathcal{I}$, $\mathrm{OPT}(\mathcal{I})$ is the profit produced by optimum *offline algorithm* OPT on input $\mathcal{I}$ when the sequence $\mathcal{I}$ is known in advance by OPT. In this work, when we study the competitive ratio, we assume all users are truthful. Given a sequence of spectrum requests, at any time instant t, an online spectrum allocation method only knows all spectrum requests that arrived from time $t-\gamma$ to timeslot t. On the other hand, for the same sequence of spectrum requests, although the optimum offline method also has to make decision for a request within γ timeslots of its arrival, it does know *all future* requests it will get, thus makes a smarter decision.

As we will see later that the performances of our methods and the lower bounds on the performance of any online algorithm vary when the penalty factor β, the advance factor α and the delay factor γ are different. For certain parameters β, α and γ, we call it (β, α, γ) problem in this paper.

3 Upper Bounds on All Online Methods

In this section, we present upper bounds on the competitive ratios of online allocation methods in cases depending on whether β is less than 1, equal to 1, or larger than 1. These bounds will be proved to be almost tight later.

3.1 Upper Bound for $(\beta < 1, \gamma, \alpha)$ Problem

We first show upper bounds on the competitive ratios when $\beta < 1$. Note that in this case we assume that every preempted job must be executed for at least one timeslot.

The main approach of our proofs in this section is *adversary argument*: an adversary carefully constructs sequences of requests to make the performance of online algorithms as bad as possible. The trick is that the adversary can choose the future requests based on the current decision of online algorithms. Then no matter what decision is made, certain performance can not be achieved.

Theorem 2: Let $\theta = \frac{1}{c\beta}$. No online algorithm has a competitive ratio $> \theta$ for $(\beta < 1, \gamma, \alpha)$ problem when $\gamma \leq \frac{\beta}{(c+1)(2+\beta)}\Delta$ for $c > 1/\beta$.

3.2 Upper Bound for $(\beta = 1, \gamma, \alpha)$ Problem

When the penalty factor $\beta = 1$, there are two different cases: $\gamma = o(\Delta)$ or $\gamma = \Omega(\Delta)$.

1) When $\gamma = o(\Delta)$, *i.e.*, $\lim \frac{\gamma}{\Delta} = 0$, we first show that no online algorithm can achieve competitive ratio more than $\sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(1, \alpha, \gamma)$ problem. Then we improve this bound to $\sqrt{2(\gamma+1)}\Delta^{-\frac{1}{2}}$.
2) When $\gamma = \Omega(\Delta)$, we show that $(1, \alpha, \gamma)$ problem cannot have a competitive ratio $1 - \epsilon$ for an arbitrary $\epsilon > 0$.

Theorem 3: [23] No online algorithm has a competitive ratio $> \sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = o(\Delta)$.

Theorem 4: [23] No online algorithm has competitive ratio $> \frac{1}{c}$ for $(\beta = 1, \gamma, \alpha)$ problem, where $\sqrt{2(\gamma+1)}\Delta^{-\frac{1}{2}} < \frac{1}{c} \leq \sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = o(\Delta)$.

Theorem 5: [23] No online algorithm has a competitive ratio $\geq 1-\epsilon$ for an arbitrary small $\epsilon > 0$, for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = \Omega(\Delta)$.

3.3 Upper Bound for $(\beta > 1, \gamma, \alpha)$ Problem

For $(\beta > 1, \gamma, \alpha)$ problem, we show that upper bound of competitive ratios is $O(\frac{\gamma+1}{\Delta})$ when $\gamma = o(\Delta)$.

Theorem 6: [23] No online algorithm has competitive ratio more than $\frac{2\beta}{(\beta-1)^2}\frac{\gamma+1}{\Delta}$ for $(\beta > 1, \gamma, \alpha)$ problem, when $\gamma = o(\Delta)$.

A surprising result from the analysis in this section is that, the performance upper bounds do not depend on the *advance factor* α. In other words, no matter how many time slots the secondary users claim their requests in advance, the theoretical upper bounds will not be improved asymptotically if the *delay factor* γ does not change.

4 Efficient Online Allocation Methods

In this section, we present several methods for efficient online spectrum allocation. We then analytically study the performances of our methods, and prove that they have asymptotically best competitive ratios. All results in this section assume that the conflict graph of the network is a complete graph.

4.1 Method $\mathcal{K}$ for $(\beta < 1, \gamma, \alpha)$ Problem

When $\beta < 1$, whenever we admit a request e_i at time t, even we preempt it at next timeslot $t+1$, we still receive a profit $(1-\beta)b_i$. Thus, we will adopt the following strategy $\mathcal{K}$: at any time t, let e_i be the request from the set of all currently available requests, denoted as $\mathcal{R}_a(t)$, with the largest bid b_i among all requests whose starting time is $t-\gamma+\alpha$. We admit request e_i at time t. It is easy to prove the following Lemma.

Lemma 7: Strategy $\mathcal{K}$ is at least $(1-\beta)$-competitive.

4.2 Method $\mathcal{G}$ for $(\beta = 1, \gamma, \alpha)$ Problem

We then consider the case $\beta = 1$, *i.e.*, the preemption penalty is exactly the remaining portion of the bid. Since we don't know anything about the future, to maximize the total profits, we shall accept request with large bid or large bid per time slot (called *density*) intuitively. The main difficulty is the tradeoff between requests with large bid and requests with large *density*. Our results on upper bounds of competitive ratio in Section 3 illustrate some intuitions. In this and following subsection, we design online algorithms whose competitive ratios match corresponding upper bounds.

Consider current time t, let $\mathcal{R}_a(t)$ be all spectrum requests submitted before time t, which are not processed. Based on the delay requirement, we know that all requests in $\mathcal{R}_a(t)$ must be submitted during the time-slots $[t-\gamma, t)$, and their starting times must be in the time-interval $[t-\gamma+\alpha, t+\alpha]$. Among all spectrum requests in $\mathcal{R}_a(t)$, let $\mathcal{R}(t) \subseteq \mathcal{R}_a(t)$ be all spectrum requests whose starting times are in the time-interval $[t, t+\gamma]$. Recall that we always have $\gamma \leq \alpha$. Our online algorithms will only make decisions at time t using the information about requests $\mathcal{R}(t)$, although it knows a superset of requests $\mathcal{R}_a(t)$. See Figure 2 for illustration, where

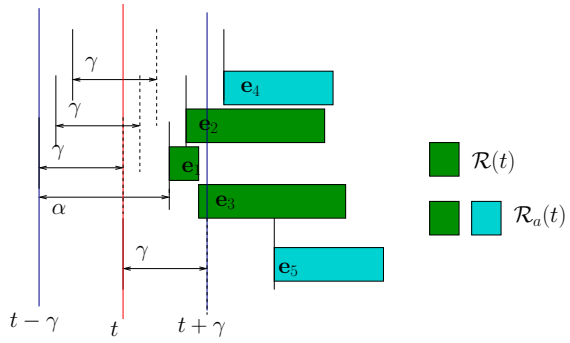


Fig. 2. All requests $\mathcal{R}_a(t)$ and a subset of requests $\mathcal{R}(t)$ with starting time in $[t, t+\gamma]$ (denoted by green color) .

$\mathcal{R}_a(t) = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5\}$ and $\mathcal{R}(t) = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. We will show that our method can achieve a competitive ratio that is already asymptotically optimum. In other words, knowing the information α will not enable us to get a method with a better competitive ratio asymptotically.

4.2.1 Candidate Requests Set

For requests $\mathcal{R}(t)$, we will find some subsets, called *candidate requests set*, using dynamic programming to optimize some objective functions. Our method will then make decisions on whether to admit these subsets of requests under some conditions involving the currently running spectrum usages.

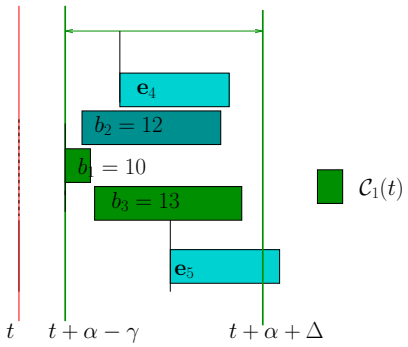


Fig. 3. Strong Candidate Set $\mathcal{C}_1(t)$ at time t (color green).

Definition 1: **Candidate Requests Set**:

1) A *strong candidate requests set* at time t, denoted as $\mathcal{C}_1(t)$, is a subset of requests from $\mathcal{R}(t)$ that has the largest total bids if $\mathcal{C}_1(t)$ were allowed to run from time $t-\gamma+\alpha$ to time at most $t+\alpha+\Delta$. See Figure 3 for illustration where $\mathcal{C}_1(t) = \{\mathbf{e}_1, \mathbf{e}_3\} \subseteq \mathcal{R}(t)$ since the total profit during time interval $[t-\gamma+\alpha, t+\alpha+\Delta]$ is maximized. We abuse the notation little bit here by also letting $\mathcal{C}_1(t)$ denote the profit made by $\mathcal{C}_1(t)$.
   Also let $\mathcal{P}(\mathcal{C}_1(t), t')$ denote the profit made from $\mathcal{C}_1(t)$ if $\mathcal{C}_1(t)$ are admitted and then possibly being preempted at a time-slot $t' \in [t-\gamma+\alpha, t+\alpha+\Delta]$.
2) A *weak candidate requests set* at time t, denoted as $\mathcal{C}_2(t)$, is a subset of requests from $\mathcal{R}(t)$ that has the largest total bids if $\mathcal{C}_2(t)$ were allowed to run during time interval

$[t-\gamma+\alpha, t+\alpha]$ (assume they are preempted at time $t+\alpha+1$). We abuse the notation little bit here by also letting $\mathcal{C}_2(t)$ denote the profit made by $\mathcal{C}_2(t)$.

Briefly speaking, $\mathcal{C}_1(t)$ maximizes the total bids at time t, and $\mathcal{C}_2(t)$ maximizes the total bid per time slot in the predictable future (since the requests are claimed in advance). Observe that the starting times of candidate sets are in the interval $[t, t+\alpha]$. Since $\gamma \leq \alpha$, at time t, we should know all requests whose starting times are from t to $t+\gamma$. So we could find the *candidate requests set* $\mathcal{C}_1(t)$ and $\mathcal{C}_2(t)$ by dynamic programming as following.

Consider requests $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n$ in the increasing order of their starting times, *i.e.*, $t-\gamma+\alpha \leq s_1 \leq s_2 \leq \cdots \leq s_n \leq t+\alpha$. When we compute the strong candidate set $\mathcal{C}_1(t)$, we will *not* consider any requests that are currently running or have been rejected before. Additionally, for the scheduling of requests in $\mathcal{C}_1(t)$, we may use one request to preempt another request in $\mathcal{C}_1(t)$. Let $\mathcal{Q}(j, s_i)$ denote the maximum possible profit made during time interval $[t, t+\gamma+\Delta]$ from $\mathcal{R}(t)$ when $\mathbf{e}_j$ is the last accepted request before or at time s_i. If the starting time s_j of $\mathbf{e}_j$ satisfies $s_j > s_i$, $\mathcal{Q}(i, s_j) = 0$ since it is not a feasible solution. Then

$$\mathcal{C}_1(t) = \max_{k \in [1,n]} \mathcal{Q}(k, s_n).$$

For each i and j such that $i \geq j$, we have

$$\mathcal{Q}(j, s_i) = \max_{k \in [1,n]} \{\mathcal{Q}(k, s_j) + b_j - \mu(\mathbf{e}_k, s_j)\}.$$

Here $\mu(\mathbf{e}_k, s_j)$ is the paid penalty when request $\mathbf{e}_k$ is preempted at time s_j.

Algorithm 1 Find $\mathcal{C}_1(t)$ by Dynamic Programming

Input:$\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_{n(t)}$ such that $s_1 \leq s_2 \leq \cdots \leq s_{n(t)}$.
Output: *strong candidate set* $\mathcal{C}_1(t)$.

```
for i = 1 to n(t) do
    for j = 1 to i do
        Q(j, s_i) = max_{k∈[1,n]}{Q(k, s_j) + b_j − μ(e_k, s_j)}
C_1(t) = max_{k∈[1,n]}{Q(k, s_{n(t)})}
```

Algorithm 1 summarizes our method finding $\mathcal{C}_1(t)$. For each i and j, the algorithm takes $O(n(t))$ time to compute $\mathcal{Q}(j, s_i)$. Therefore, it takes $O(n(t)^3)$ time to find $\mathcal{C}_1(t)$ where $n(t)$ is the total number of spectrum requests in $\mathcal{R}(t)$. Our method can be easily extended to find $\mathcal{C}_2(t)$ with same running time.

4.2.2 Our Greedy Algorithm $\mathcal{G}$

We then propose an online algorithm $\mathcal{G}$. At each time t, the input of our algorithm $\mathcal{G}$ is $\mathcal{R}_a(t)$. Algorithm $\mathcal{G}$ should decide whether a request that arrived at time slot $t-\gamma$ will be accepted or rejected. The basic idea of our method is as follows.

Assume that time 0 is the reference point for time. First, before time γ, we do not need to make decisions on any arrived requests. At time γ, we need to decide whether to accept/reject requests that arrive at time 0. If channel is not occupied, we will simply accept the request from $\mathcal{C}_1(\gamma)$ with the earliest starting time being α, if there is any. We call $\mathcal{C}_1(\gamma)$ as *virtual meta* job requests currently being accepted.

For any time $t > \gamma$, our method will choose either a request from the strong candidate request $\mathcal{C}_1(t)$, or a request from weak candidate request $\mathcal{C}_2(t)$, or continue to run the request (which could be empty) chosen previously. Our method deal with three complementary cases about the channel status at time $t-\gamma+\alpha$, separately.

Case 1: Non-occupied Channel: If the channel will be *empty* at time $t-\gamma+\alpha$, we find $\mathcal{C}_1(t)$ with the maximum overall profit in $\mathcal{R}(t)$. We accept the request in $\mathcal{C}_1(t)$ with starting time $t-\gamma+\alpha$, and we say that the channel is being used by candidate requests set $\mathcal{C}_1(t)$. In other words, here we treat $\mathcal{C}_1(t)$ as a large *virtual meta* spectrum request. Note that at current time slot t we only admit the first spectrum request from $\mathcal{C}_1(t)$ while leave the admission decisions on other requests in $\mathcal{C}_1(t)$ *pending*. Whether these pending "admitted" requests will be actually admitted depending on future coming requests. If future coming requests are better, we will preempt this virtual meta request $\mathcal{C}_1(t)$[1]. Thus, those preempted pending admissions from $\mathcal{C}_1(t)$ will not be processed at all.

Case 2: Occupied by Weak Candidate Requests: If the channel will be used by a *weak candidate requests set* $\mathcal{C}_2(t)$ at time $t-\gamma+\alpha$ and this candidate requests set $\mathcal{C}_2(t)$ *weakly preempted* (exact definitions will be given later) some other candidate requests set before, all requests from $\mathcal{R}_a(t)$ submitted at time $t-\gamma$ will *not* be admitted.

Case 3: Occupied by Strong Candidate Requests: The last case is that the channel will be used by a *strong candidate requests set* $\mathcal{C}_1(t)$ at time $t-\gamma+\alpha$. Assume the request to be run at time $t-\gamma+\alpha$ is $\mathbf{e}_j$ from some virtual candidate requests set $\mathcal{C}_1(t_1)$. There are three possible steps in this case.

1) We first find the strong candidate requests set $\mathcal{C}_1(t)$. The first request $\mathbf{e}_i \in \mathcal{C}_1(t)$ such that $s_i = t-\gamma+\alpha$ will be accepted only if
$$\mathcal{C}_1(t) \geq c_1 \cdot \mathcal{C}_1(t_1), \tag{1}$$
where $c_1 > 1$ is a constant parameter. In other words, we use a virtual meta request $\mathcal{C}_1(t)$ to replace another virtual request $\mathcal{C}_1(t_1)$ if the potential profit from new virtual meta request is sufficiently larger. We call it a *Strong-Preemption*.
2) If strong-preemption cannot be applied, we find $\mathcal{C}_2(t)$. The request $\mathbf{e}_i \in \mathcal{C}_2(t)$ such that $s_i = t-\gamma+\alpha$ will be accepted only if
$$\mathcal{C}_2(t) + \mathcal{P}(\mathcal{C}_1(t_1), t-\gamma+\alpha) \geq c_2 \cdot \mathcal{C}(t_1), \tag{2}$$
where $c_2 > 0$ is an adjustable control parameter. In other words, we use a virtual candidate requests set $\mathcal{C}_2(t)$ to replace another virtual candidate requests set $\mathcal{C}_1(t_1)$ if the profit from this new meta request and the profit from the meta request $\mathcal{C}(t_1)$ being preempted at future time $t-\gamma+\alpha$ is at least a constant c_2 fraction of the profit by keeping running the meta request $\mathcal{C}(t_1)$. We call it a *Weak-Preemption*. In this subcase, all requests in $\mathcal{C}_2(t)$ will be accepted and the last request will be terminated at time $t+\alpha+1$ automatically.

1. Preempting $\mathcal{C}_1(t)$ at a time, say t', will be implemented by preempting the requests from $\mathcal{C}_1(t)$ that are supposed to be running at and after time t'.

3) If the weak-preemption cannot be applied also, we accept the request in the previously used candidate requests set $\mathcal{C}(t_1)$, whose starting time is $t-\gamma+\alpha$ (if there is any) or continue the request $\mathbf{e}_j$ that will run through the time slot $t-\gamma+\alpha$.

Algorithm 2 summarizes our online spectrum allocation method $\mathcal{G}$.

Algorithm 2 Online Spectrum Allocation $\mathcal{G}$

Input: A constant parameter $c_1 > 1$, an adjustable control parameter $c_2 > 0$, γ, α, Δ, $\mathcal{R}_a(t)$, $\mathcal{R}(t)$, $\mathcal{C}_1(t)$, and $\mathcal{C}_2(t)$.
Current candidate requests set $\mathcal{C}$ from time $t' < t$. Here $\mathcal{C} = \mathcal{C}_1(t')$ if $\mathcal{C}_1(t')$ strongly preempted others, or $\mathcal{C} = \mathcal{C}_2(t')$ if $\mathcal{C}_2(t')$ strongly preempted others.
Output: whether requests submitted at time $t-\gamma$ will be admitted and new *current candidate requests set* $\mathcal{C}$.

if $\mathcal{C} = \mathcal{C}_2(t')$ **then**
  if $t - t' \geq \gamma$ **then**
    $\mathcal{C} = \emptyset$;
  else
    Accept request $\mathbf{e}_i \in \mathcal{C}_2(t)$ such that $s_i = t-\gamma+\alpha$.
if $\mathcal{C} = \mathcal{C}_1(t')$ or $\emptyset$ **then**
  if $\mathcal{C}_1(t) \geq c_1 \cdot \mathcal{C}_1(t')$ **then**
    $\mathcal{C} = \mathcal{C}_1(t)$;
    Accept request $\mathbf{e}_i \in \mathcal{C}_1(t)$ such that $s_i = t-\gamma+\alpha$.
  else if $\mathcal{C}_2(t) + \mathcal{P}(\mathcal{C}_1(t'), t) \geq c_2 \cdot \mathcal{C}_1(t)$ **then**
    $\mathcal{C} = \mathcal{C}_2(t)$;
    Accept request $\mathbf{e}_i \in \mathcal{C}_2(t)$ such that $s_i = t-\gamma+\alpha$.
  else
    Accept request $\mathbf{e}_i \in \mathcal{C}_1(t')$ such that $s_i = t-\gamma+\alpha$.

4.2.3 Theoretical Performance Study

We then show that our algorithm $\mathcal{G}$ is asymptotically optimal if we choose constant c_1 and control parameter c_2 carefully. We use $\mathcal{G}(\mathcal{I})$ to denote the profit made on requests sequence $\mathcal{I}$ by our algorithm $\mathcal{G}$. Also use $\mathrm{OPT}(\mathcal{I})$ to denote the profit made by the optimal offline algorithm. To analyze the performance of our method, we first give a definition *candidate sequence*.

Definition 2: **Candidate Sequence**: A *candidate sequence* is a sequence of *candidate requests sets* $\mathcal{C}_1(t_i)$, $\mathcal{C}_1(t_{i+1})$, $\cdots$, $\mathcal{C}_1(t_{j-1})$, $\mathcal{C}_2(t_j)$ or $\mathcal{C}_1(t_i)$, $\mathcal{C}_1(t_{i+1})$, $\cdots$, $\mathcal{C}_1(t_{j-1})$, $\mathcal{C}_1(t_j)$ which satisfies all of following three conditions.

1) Strong candidate requests set $\mathcal{C}_1(t_i)$ does not preempt another candidate requests set;
2) Strong candidate requests set $\mathcal{C}_1(t_{k+1})$ strongly preempts strong candidate requests set $\mathcal{C}_1(t_k)$, for $k = i, \cdots, j-2$.
3) Weak candidate set $\mathcal{C}_2(t_j)$ weakly preempts strong candidate set $\mathcal{C}_1(t_{j-1})$; or a strong candidate set $\mathcal{C}_1(t_j)$ strongly preempts $\mathcal{C}_1(t_{j-1})$ and $\mathcal{C}_1(t_j)$ is not preempted by another requests set.

Here we use the parameters of the first and last *candidate requests set* to denote a *candidate sequence*, *e.g.* $\mathcal{S}(t_i, t_j)$. According to the definition, we can decompose the solution of algorithm $\mathcal{G}$, *i.e.*, $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m\}$, into multiple *candidate sequences*. Notice that each spectrum request $\mathbf{e}$ will appear in exactly one *candidate sequence*. We use $\mathcal{G}(\mathcal{S}(t_i, t_j))$ to denote the profit made on *candidate sequence* $\mathcal{S}(t_i, t_j)$ by algorithm $\mathcal{G}$. And we use $\mathrm{OPT}[t_i, t_j]$ to denote the profit made by optimal offline algorithm on the requests whose starting times are in interval $[t_i, t_j]$.

Lemma 8: For each *candidate sequence* $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm $\mathcal{G}$, we have

$$\mathcal{G}(\mathcal{S}(s_i, s_j)) \geq \min(c_1, c_2) \cdot \mathcal{C}_1(s_{j-1})$$

Proof: By definition of candidate sequences, there are two different cases: (1) $\mathcal{C}_1(s_j)$ strongly preempted $\mathcal{C}_1(s_{j-1})$ or (2) $\mathcal{C}_2(s_j)$ weakly preempted $\mathcal{C}_1(s_{j-1})$.

If request $\mathcal{C}_1(s_j)$ strongly preempted $\mathcal{C}_1(s_{j-1})$, $\mathcal{G}$ makes at least $\mathcal{C}_1(s_j) \geq c_1 \cdot \mathcal{C}_1(s_{j-1})$. If request $\mathcal{C}_2(s_j)$ weakly preempted $\mathcal{C}_1(s_{j-1})$, $\mathcal{G}$ makes at least $\mathcal{P}(\mathcal{C}_1(s_{i-1}), s_j) + \mathcal{C}_2(s_j) \geq c_2 \cdot \mathcal{C}_1(s_{j-1})$. So our lemma holds for either case. □

Lemma 9: [23] For each *candidate sequence* $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm $\mathcal{G}$, for each $i \leq k < j$, we have

$$\mathrm{OPT}[s_k, s_{k+1}] \leq (c_1 + c_2 + \frac{c_2}{\sqrt{(\gamma+1)/\Delta}})\mathcal{C}_1(s_k)$$

Lemma 10: For each *candidate sequence* $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm $\mathcal{G}$, we have

$$\mathrm{OPT}[s_i, s_j] \leq (c_1+1+\frac{1}{c_1-1})(c_1+c_2+\frac{c_2}{\sqrt{(\gamma+1)/\Delta}})\mathcal{C}_1(s_{j-1})$$

Proof: Obviously $\mathrm{OPT}[s_i, s_j] = \sum_{k=i}^{j-1} \mathrm{OPT}[s_k, s_{k+1}]$. From Lemma 9, we have $\mathrm{OPT}[s_k, s_{k+1}] \leq (c_1 + c_2 + \frac{c_2}{\sqrt{(\gamma+1)/\Delta}})\mathcal{C}_1(s_k)$. Thus, $\mathrm{OPT}[s_i, s_j] \leq (c_1 + c_2 + \frac{c_2}{\sqrt{(\gamma+1)/\Delta}}) \sum_{k=i}^{j-1} \mathcal{C}_1(s_k)$. Based on the condition of strong-preemption, we have $\mathcal{C}_1(s_k) \geq c_1 \cdot \mathcal{C}_1(s_{k-1})$ for all $i \leq k \leq j$. The lemma then follows. □

Theorem 11: Algorithm $\mathcal{G}$ is $\Theta(\sqrt{\gamma+1}\Delta^{-\frac{1}{2}})$-competitive.

Proof: Given a request sequence $\mathcal{I}$, we decompose the solution by algorithm $\mathcal{G}$ into multiple *candidate sequence* $\mathcal{S}(t_i, t_j)$. Then the total profit made by algorithm $\mathcal{G}$ is $\sum_{\forall \mathcal{S}(t_i,t_j)} \mathcal{S}(t_i, t_j)$. On the other hand, the total profit made by the optimal offline algorithm is $\sum_{\forall \mathcal{S}(t_i,t_j)} \mathrm{OPT}[t_i, t_j]$. From Lemma 8 and Lemma 10, the competitive ratio of Algorithm $\mathcal{G}$ is at least

$$\frac{\sum_{\forall \mathcal{S}(t_i,t_j)} \mathcal{S}(t_i, t_j)}{\sum_{\forall \mathcal{S}(t_i,t_j)} \mathrm{OPT}[t_i, t_j]} \geq \frac{\min(c_1, c_2)}{(c_1 + 1 + \frac{1}{c_1-1})(c_1 + c_2 + \frac{c_2}{\sqrt{(\gamma+1)/\Delta}})}$$

It is easy to show that the right hand side gets the maximum value $\frac{1}{4(1+\sqrt{\Delta/(\gamma+1)})}$ if $c_1 = 2$ and $c_2 = \frac{2}{1+\sqrt{\Delta/(\gamma+1)}}$. Thus the competitive ratio is at least $\frac{1}{8}\sqrt{\frac{\gamma+1}{\Delta}}$ when $\gamma \leq \Delta - 1$. □

Let $n(t)$ be the cardinality of $\mathcal{R}_a(t)$. At time t, Algorithm 2 takes $O(n(t)^3)$ time to find $\mathcal{C}_1(t)$ and $\mathcal{C}_2(t)$, then takes constant time to make decision. The time complexity of Algorithm 2 is $\sum_{\forall t} n(t)^3$. Notice that each request could appear in at most $\Delta + \gamma$ different $\mathcal{R}_a(t)$. Let n be the number of all online requests. We have $\sum_{\forall t} n(t) \leq (\Delta + \gamma)n$ and $n(t) \leq n$ for all t, which implies following theorem.

Theorem 12: Time complexity of Alg. 2 is $O((\Delta + \gamma)n^3)$.

4.3 Method $\mathcal{H}$ for $(\beta > 1, \gamma, \alpha)$ Problem

In this subsection, we propose a greedy algorithm $\mathcal{H}$ for $(\beta > 1, \gamma, \alpha)$ Problem. At time t, our method $\mathcal{H}$ should decide whether a request whose start time is $t+\alpha-\gamma$ will be accepted.

Algorithm 3 Online Spectrum Allocation $\mathcal{H}$

Input: A constant parameter $c > 1+\beta$, γ, α, Δ, $\mathcal{R}_a(t)$, $\mathcal{R}(t)$, $\mathcal{C}_1(t)$. Previous *current candidate requests set* $\mathcal{C} = \mathcal{C}_1(t')$ where $t' < t$. Here $\mathcal{C}_1(t')$ may be empty.

Output: whether requests submitted at time $t-\gamma$ will be admitted and new *current candidate requests set* $\mathcal{C}$.

1: **if** $\mathcal{C}_1(t) \geq c \cdot \mathcal{C}_1(t')$ **then**
2: $\quad \mathcal{C} = \mathcal{C}_1(t)$;
3: $\quad$ Accept request $\mathbf{e}_i \in \mathcal{C}_1(t)$ such that $a_i = t-\gamma$.
4: **else**
5: $\quad$ Accept request $\mathbf{e}_i \in \mathcal{C}_1(t')$ such that $a_i = t-\gamma$.

Then we show that algorithm $\mathcal{H}$ is asymptotically optimal. The notations are used in previous subsection.

Theorem 13: Algorithm $\mathcal{H}$ is $\frac{(c-\beta-1)}{c^2}\frac{\gamma+1}{\Delta+\gamma+1}$ competitive.

Proof: We divide the requests accepted by algorithm $\mathcal{H}$ into multiple *candidate requests sets*. Let us consider the profit made by $\mathcal{H}$ on each *candidate requests set* $\mathcal{C}_1(t)$. For each admitted $\mathcal{C}_1(t)$, we redistribute $(1+\beta)\mathcal{C}_1(t')$ of the money from $\mathcal{C}_1(t)$ to $\mathcal{C}_1(t')$ if $\mathcal{C}_1(t)$ preempts $\mathcal{C}_1(t')$.

There are three complementary cases here.

Case 1: $\mathcal{C}_1(t)$ does not preempt others and is not preempted. $\mathcal{H}$ makes $\mathcal{C}_1(t)$ profit on $\mathcal{C}_1(t)$. Let $t_s = t+\alpha-\gamma$. For the optimal offline algorithm OPT, if time $t_s + i(\gamma+1)$ (for some integer $i > 0$) is before the end time of $\mathcal{C}_1(t)$, we have $\mathrm{OPT}[t_s+i(\gamma+1), t_s+(i+1)(\gamma+1)] \leq c \cdot \mathcal{C}_1(t))$. Otherwise, the requests whose starting times are during time interval $[t_s+i(\gamma+1), t_s+(i+1)(\gamma+1)]$ should preempt $\mathcal{C}_1(t)$. We know that the total running time of $\mathcal{C}_1(t)$ is no more than $\Delta+\gamma+1$. Thus no more than $c\lceil\frac{\Delta+\gamma+1}{\gamma+1}\rceil\mathcal{C}_1(t))$ profit will be made by OPT during the running time of $\mathcal{C}_1(t))$.

Case 2: $\mathcal{C}_1(t)$ does not preempt others and is preempted in $\mathcal{H}$. $\mathcal{H}$ makes $\mathcal{C}_1(t))$ profit on $\mathcal{C}_1(t)$ since the *candidate requests* which preempted $\mathcal{C}_1(t)$ will distribute $(1+\beta)\mathcal{C}_1(t)$ profit to $\mathcal{C}_1(t)$. And at most $\beta\mathcal{C}_1(t)$ profit is paid as penalty. No more than $c\lceil\frac{\Delta+\gamma+1}{\gamma+1}\rceil\mathcal{C}_1(t))$ profit will be made by OPT during the running time of $\mathcal{C}_1(t))$.

Case 3: $\mathcal{C}_1(t)$ preempts another candidate requests set in $\mathcal{H}$. $\mathcal{H}$ makes at least $\frac{c-\beta-1}{c}\mathcal{C}_1(t)$ profit from $\mathcal{C}_1(t)$ (after considering redistributing some to the candidate requests set being preempted by $\mathcal{C}_1(t)$) since we distribute $(1+\beta)\mathcal{C}_1(t') \leq \frac{1+\beta}{c}\mathcal{C}_1(t)$ to the candidate set preempted by $\mathcal{C}_1(t)$. If $\mathcal{C}_1(t)$ is not preempted, we make at least $\frac{c-\beta-1}{c}\mathcal{C}_1(t)$. If $\mathcal{C}_1(t)$ is preempted, we will receive $(1+\beta)\mathcal{C}_1(t)$ from the future candidate requests set that preempts it. Thus, we always make at least $\mathcal{C}_1(t)$. For OPT it is easy to show that OPT will make no more than $c\lceil\frac{\Delta+\gamma+1}{\gamma+1}\rceil\mathcal{C}_1(t)$ profit.

Thus, the competitive ratio for each candidate requests set is at least $\frac{(c-\beta-1)}{c^2}\lceil\frac{\gamma+1}{\Delta+\gamma+1}\rceil$. The competitive ratio of $\mathcal{H}$ is at least $\frac{(c-\beta-1)}{c^2}\frac{\gamma+1}{\Delta+\gamma+1}$. We finish the proof. □

When $\gamma = a\Delta - 1$, it is easy to show that

Theorem 14: Method $\mathcal{H}$ is at least $\frac{a}{4(1+a)(1+\beta)}$-competitive (by choosing $c = 2(1+\beta)$), when $\gamma = a\Delta - 1$.

5 More General Networks

All studies in previous sections assumed that the conflict graph of networks, which models the location-dependent conflict, is a complete graph. In this section, we extend our algorithms for the networks where each request $\mathbf{e}_i$ asks for spectrum usage in a disk-shaped region D_i centered at (x_i, y_i) with a radius r_i. For the performance upper bounds, all Lemmas and Theorems in Section 3 still hold since complete conflict graph is a special case.

If we can find the strong (weak) candidate sets $\mathcal{C}_1(t)$ ($\mathcal{C}_2(t)$) in polynomial time as we did by using Algorithm 1 in subsection 4.2, then our method $\mathcal{G}$ (or $\mathcal{H}$) still works since all previous analysis still holds. However, the main difficulty is that, in this case, the central authority may accept multiple requests at same time, which makes it hard to get the strong/weak candidate sets in polynomial time by using dynamic programming. Therefore, we propose to approximate strong/weak candidate sets at each time t in polynomial time by using *shifting strategy* [15] as follows.

We divide the request disks into different levels according to their radii. At same level, the radii of all request disks are within a constant factor λ of each other. Let d be the diameter of smallest request disk, level i includes all request disks whose diameters are within $[d\lambda^{i-1}, d\lambda^i]$.

First we show that a PTAS can be achieved in *each* level i. In other word, here we approximate the strong/weak candidate sets when only request disks in level i are considered. Let $k > 1$ be an integer. At level i, a (p, q) partition is defined as the region is partitioned by a set of horizontal lines $y = \cdots, -2\lambda^i k + p, -\lambda^i k + p, p, \lambda^i k + p, 2\lambda^i k + p, \cdots$, and a set of vertical lines $x = \cdots, -2\lambda^i k + q, -\lambda^i k + q, q, \lambda^i k + q, 2\lambda^i k + q, \cdots$. In each partition, the region is partitioned into a number of $\lambda^i k \times \lambda^i k$ squares. We ignore the requests whose disks are hit by any lines of the partition, and find the strong/weak candidate sets in each $\lambda^i k \times \lambda^i k$ square. Together with strong/weak candidate sets in all squares, we get strong/weak candidate sets for a partition.

Lemma 15: Strong/weak candidate sets for each partition can be computed in polynomial time.

Proof: According to area argument, the number of independent requests that can be accepted at same time is no more than $C = 4\lambda^2 k^2/\pi$ which is a constant. Then we can find the strong/weak candidate sets in each $\lambda^i k \times \lambda^i k$ square in polynomial time by using dynamic programming as we did in Algorithm 1. Assume $n_{a,b}(t)$ requests located in a $\lambda^i k \times \lambda^i k$ square whose top left corner is $(a\lambda^i k + p, b\lambda^i k + q)$ in a (p, q) partition at time t. The main difference is that we need to define $\mathcal{Q}(e_1, e_2, \cdots, e_C, s)$ as the maximum possible profit from $\mathcal{R}(t)$ when $e_1, e_2, \cdots, e_C$ are the last independent requests accepted by central authority before or at time s. Then

$$\mathcal{C}_1(t) = \max \mathcal{Q}(e_1, e_2, \cdots, e_C, \max s)$$

where $\max s$ the latest start time among $\mathcal{R}(t)$. Since there are at most $\binom{n_{a,b}(t)}{C+1} = O(n_{a,b}(t)^{C+1})$ different $\mathcal{Q}$, and each

$\mathcal{Q}$ can be computed within $\binom{n_{a,b}(t)}{C} = O(n_{a,b}(t)^C)$ time, the time complexity to compute strong/weak candidate sets in a $k \times k$ square whose left top corner is $(ak+i, bk+j)$ at time t is $O(n_{a,b}(t)^{2C+1})$. Let $n(t)$ denote the total number of requests in $\mathcal{R}(t)$. The time complexity to compute strong/weak candidate sets in all $\lambda^i k \times \lambda^i k$ squares of a partition (p,q) at time t is no more than $O(n(t)^{2C+1})$. □

In each partition, some requests may be ignored. To achieve a better approximation, we shift the horizontal and vertical lines, and compute strong/weak candidate sets for k^2 partitions where $p \in [1,k]$ and $q \in [1,k]$. We set the best strong/weak candidate sets among all partitions as the strong/weak candidate sets of time t. It takes $O(k^2 n(t)^{2C+1})$ time to find strong/weak candidate sets at each time. Then the total running time is at most $O(k^2 n^{2C+2})$ where n is the number of total requests since we have to compute this at most n times.

On the other hand, the strong/weak candidate sets we find is at least a $1-\frac{2}{k}$ approximation as following lemma.

Lemma 16: At least one partition achieves at least $1-\frac{2}{k}$ approximation.

Proof: We prove the lemma by using strong candidate set. The weak candidate set case follows same proof. Let $\mathcal{C}_1^{i,j}(t)$ denote the total profit made by all requests such that: (1) appear in $\mathcal{C}_1(t)$; (2) not ignored in partition (i,j). Then

$$\sum_{\forall i,j} \mathcal{C}_1^{i,j}(t) \geq (k^2-2k)\mathcal{C}_1(t)$$

since each request is ignored in at most 2 partitions.

On the other hand, $\mathcal{C}_1^{i,j}(t)$ is smaller than the profit of the strong candidate set for partition (i,j), which is no larger than the profit of our approximated strong candidate set, called $\mathcal{C}_1'(t)$. In other words,

$$k^2\mathcal{C}_1'(t) \geq \sum_{\forall i,j} \mathcal{C}_1^{i,j}(t) \geq (k^2-2k)\mathcal{C}_1(t)$$

which implies $\mathcal{C}_1'(t) \geq (1-\frac{2}{k})\mathcal{C}_1(t)$ and finishes the proof. □

According to Lemmas 15 and 16, shifting strategy gives a PTAS for *each* level i. The result can be extended when request disks in *all* levels are considered. The main technology is dynamic programming which is discussed in Section 3.2 of [15]. We omit the details here.

Since the strong/weak candidate sets can be approximated well, we have the following theorem.

Theorem 17: Our methods ($\mathcal{K}$, $\mathcal{G}$, $\mathcal{H}$) will achieve asymptotically optimum competitive ratios in polynomial time for general networks modeled by disk graphs. The competitive ratios will have an additional factor $1-\epsilon$ for any constant $\epsilon > 0$ comparing with previous analysis, when we set $\epsilon = \frac{2}{k}$.

6 TOFU: Dealing With Selfish Users

In this section, we show how to design a mechanism based the allocation algorithm described in Section 4 when secondary users could be selfish. For the spectrum allocation and auction problem, when each secondary user declares his request, he may lie on the bid, and time requirement. We need to design rules such that each secondary user has incentives to declare his request truthfully. Each secondary user i has its own private information $\mathbf{t}_i$, including b_i, a_i, and t_i. Let $\mathbf{a}_i = (b_i', a_i', t_i')$ be the value user i will report. For each vector of actions $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n)$, a mechanism $M = (\mathcal{A}, P)$, (including allocation algorithm $\mathcal{A}$ and pricing scheme P), computes a spectrum allocation $\mathcal{A}(\mathbf{a}) = (\mathcal{A}_1(\mathbf{a}), \mathcal{A}_2(\mathbf{a}), \cdots, \mathcal{A}_n(\mathbf{a}))$ and a payment vector $\mathbf{p}(\mathbf{a}) = (\mathbf{p}_1(\mathbf{a}), \mathbf{p}_2(\mathbf{a}), \cdots, \mathbf{p}_n(\mathbf{a}))$. Each user i will be allocated $\mathcal{A}_i(\mathbf{a})$ and be charged $\mathbf{p}_i(\mathbf{a})$.

A mechanism is said to be truthful if it satisfies both incentive compatible (IC) and individual rational (IR) properties. A mechanism is incentive compatible if every user i will maximize its utility by reporting its private type $\mathbf{t}_i$ truthfully. A mechanism is individual rational if the utility of an agent winning the auction is not less than its utility of losing the auction. We assume that no user will delay his/her request and a user will not lie about t_i and s_i. Unfortunately, in this work, we cannot design an online spectrum allocation mechanism that is always truthful. Instead, we will show that under our mechanism TOFU, regardless of the actions of other users, no user i can improve its profit by reporting a bid b_i' that is smaller than its true bid value b_i. We call this property as *semi-truthful*.

For the spectrum auction problem, the final profit (or the utility) of a user i is 0 if its request is rejected. If its request is admitted, the final profit is

$$utility(i) = b_i - \mathbf{p}_i + \mu(b_i', \ell_i', t_i).$$

where b_i is the actual valuation, $\mathbf{p}_i$ is the real payment, and $\mu(b_i', \ell_i, t_i)$ is the potential preemption penalty, where b_i' is the actual bid of user i.

It is a forklore result that the allocation method in a mechanism that can prevent lying must have the monotone property. Here a spectrum allocation method $\mathcal{A}$ is monotone if a user i is granted the spectrum usage under $\mathcal{A}$ with a bid $\mathbf{e}_i = (b_i, a_i, t_i)$, then the user i will still be granted the spectrum usage under $\mathcal{A}$ if the user increases bid b_i, and/or decreases the required time duration t_i. First, our methods (Algorithm 2, Algorithm 3) presented in previous section do have the monotone property. In our algorithm, we need to find strong candidate requests set, and weak candidate requests set. Both sets will be found by dynamic programming. It is easy to show that these dynamic programming methods are monotone. Thus, it is possible to design a mechanism using our algorithms ($\mathcal{G}$ and $\mathcal{H}$) as spectrum allocation methods.

Consider a user i, assume the bids of all other users remain the same. We first propose the following definition based on the monotone property of our methods.

Definition 3: Let $\underline{b}_i$ be the minimum bid that i has to bid to get admitted when its spectrum request is to be processed at γ timeslots later. Let $\overline{b}_i$ be the minimum bid that i has to bid to get admitted and not get preempted later.

Clearly, $\underline{b}_i \leq \overline{b}_i$. For a secondary user whose request is not granted, the value $\underline{b}_i = 0$. The values $\underline{b}_i$ and $\overline{b}_i$ clearly can be computed in polynomial time since the bid values of all other users are known. Here $\underline{b}_i$ can be computed at the time $a_i + \gamma$. To compute the value $\overline{b}_i$, we need to know whether request i will be preempted later. Since the latest job that can preempt $\mathbf{e}_i$ must have starting time no later than $s_i + \Delta$, we must process the potential jobs that can preempt $\mathbf{e}_i$ no later

than $s_i + \Delta - \alpha + \gamma$. In other words, the value $\overline{b}_i$ can be computed within time $\Delta + \gamma$ after job i arrived. Then our protocol TOFU works as follows.

Algorithm 4 TOFU Semi-Truthful Online Spectrum Auction

1: Based on the setting, we use either Algorithm 2 or Algorithm 3 to decide whether a request $\mathbf{e}_i$ will be admitted or not.
2: Our mechanism will charge a user i a payment

$$\mathbf{p}_i = \underline{b}_i$$

Theorem 18: Our mechanism TOFU is semi-truthful, *i.e.*, to maximize its profit, every secondary user will not bid a price lower than its actual value.

Proof: Consider a user i arrived at time t. Let b_i be its private willing bid and b'_i be its actual bid. When i bids some value $b'_i \in (\underline{b}_i, \overline{b}_i)$, i will get preempted with some unserved time ℓ'_i and receive a compensation $\mu(b'_i, \ell'_i, t_i) = \beta\frac{\ell'_i}{t_i}b'_i$.

Case 1: $b_i \geq \overline{b}_i$. There are 3 strategies again for i.

1) If $b'_i \geq \overline{b}_i$, then user i will be admitted and will not be preempted. It will be charged a payment $\mathbf{p}_i < \overline{b}_i$ and thus will get a profit $b_i - \mathbf{p}_i > 0$.
2) If $b'_i < \underline{b}_i$, it will not get admitted and thus get 0 utility.
3) If $b'_i \in (\underline{b}_i, \overline{b}_i)$, then i will be admitted and then be preempted with unserved time ℓ'_i. It will get a compensation $\mu(b'_i, \ell'_i, t_i) = \beta\frac{\ell'_i}{t_i}b'_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta\frac{\ell'_i}{t_i})b_i - \mathbf{p}_i + \beta\frac{\ell'_i}{t_i}b'_i = b_i - \mathbf{p}_i - \beta\frac{\ell'_i}{t_i}(b_i - b'_i)$, which is less than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i < \overline{b}_i \leq b_i$.

Thus, in all subcases, reporting untruthfully will not gain any benefit for i.

Case 2: $b_i \in (\underline{b}_i, \overline{b}_i)$. There are 5 strategies for i.

1) If $b'_i = b_i$, it will be admitted and then preempted. Let ℓ_i be the unserved timeslots when bidding b_i. Then its utility is $(1 - \beta\frac{\ell_i}{t_i})b_i - \mathbf{p}_i + \beta\frac{\ell_i}{t_i}b_i = b_i - \mathbf{p}_i > 0$.
2) If $b'_i \geq \overline{b}_i$, then user i will be admitted and will not be preempted. It will be charged a payment $\mathbf{p}_i < \overline{b}_i$ and thus will get a profit $b_i - \mathbf{p}_i > 0$.
3) If $b'_i < \underline{b}_i$, it will not get admitted and thus get 0 utility.
4) If $b_i < b'_i < \overline{b}_i$, then i will be admitted and then be preempted with a shorter unserved time $\ell'_i \leq \ell_i$, due to the monotone property of our method. It will get a compensation $\mu(b'_i, \ell'_i, t_i) = \beta\frac{\ell'_i}{t_i}b'_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta\frac{\ell'_i}{t_i})b_i - \mathbf{p}_i + \beta\frac{\ell'_i}{t_i}b'_i = b_i - \mathbf{p}_i - \beta\frac{\ell'_i}{t_i}(b_i - b'_i)$, which is larger than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i > b_i$.
5) If $\underline{b}_i < b'_i < b_i$, then i will be admitted and then be preempted with a longer unserved time $\ell'_i \geq \ell_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta\frac{\ell'_i}{t_i})b_i - \mathbf{p}_i + \beta\frac{\ell'_i}{t_i}b'_i = b_i - \mathbf{p}_i - \beta\frac{\ell'_i}{t_i}(b_i - b'_i)$, which is smaller than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i < b_i$.

Thus, in all subcases, reporting lower will not gain any benefit for i.

Case 3: $b_i \leq \underline{b}_i$. There are several strategies again for i.

1) If $b'_i \leq \underline{b}_i$, it will not get admitted and thus has 0 profit.
2) If $b'_i \geq \overline{b}_i$, it will be admitted and not be preempted. In this case, its profit is $b_i - \mathbf{p}_i < 0$.
3) If $\underline{b}_i < b'_i < \overline{b}_i$, it will be admitted and be preempted with an unserved time ℓ'_i. In this case, its profit is $b_i - \mathbf{p}_i - \beta\frac{\ell'_i}{t_i}(b_i - b'_i) < 0$ since $b_i < \mathbf{p}_i$ and $b_i < b'_i$.

In all subcases, bidding untruthfully is not beneficial to user i. □

Observe that, in our scheme, the only scenario (case 2.4) that a secondary user can gain benefit is when $b_i \in (\underline{b}_i, \overline{b}_i)$ and it bids a value $b'_i \in (b_i, \overline{b}_i)$. The user i must bid larger than its true valuation b_i, but not larger than $\overline{b}_i$. Observe that value $\overline{b}_i$ is computed at time $a_i + \gamma + \Delta$ using the requests submitted during time interval $[a_i, a_i + \Delta]$, where user i does not know when it arrives at time a_i. Thus, we have

Theorem 19: If any user i does not know the requests within Δ time after its arrival, our mechanism TOFU is truthful, *i.e.*, to maximize its profit, every secondary user will bid truthfully even it learned history.

It is not difficult to construct examples in which a user can gain profit by bidding larger than its true valuation if it can know the bids of other users. It will be an interesting future work to study the total payment of a mechanism compared with the best possible payments collected by a truthful mechanism. Notice that, when we know the distributions of all bids, and there is only one spectrum, and no spatial and temporal reuse, Myerson [17] presented a mechanism that guarantees the payment is optimal. We note that it is very challenging to design a mechanism with a total payment close to optimum for the problem studied here.

7 Simulation Studies

We conducted extensive simulations to study the performance of our algorithm $\mathcal{G}$, specifically, the competitive ratio of our algorithm in practice. We set $c_1 = 2$ and $c_2 = \sqrt{\frac{\gamma+1}{\Delta}}$ for algorithm $\mathcal{G}$. Suppose the total service time is always 2000 time slots, which is large enough comparing with time requirements. In our experiment, we first generate n secondary nodes that are randomly placed. We randomly deployed 100 nodes in a 5×5 square area. Assume that all secondary users locate at these positions. The geometry location of a request is exactly the location of secondary user who claimed that request. We assume that any two requests within distance 1 will conflict with each other if they requested the same channel for some timeslots. We then generate random requests with random bid values, and time requirement. The bid of each request is uniformly distributed in $[0, 1]$, the time requirement of each request is uniformly distributed in $[1, \Delta]$.

Observe that the average load of a node in our simulation is $\frac{J \cdot D}{T \cdot n}$, where J is total number of requests, D is the average duration of a request (thus $D = \Delta/2$ in our setting), T is total duration of scheduling (thus $T = 2000$ timeslots here), and n is number of secondary users ($n = 100$ here). Observe that the expected number of users in a unit area is $n/5^2 = 4$ in our setting. Thus, varying J and Δ, we tested both lightly loaded system and also highly loaded system.

7.1 Competitive Ratio of Algorithm $\mathcal{G}$

A number of parameters may affect the performances of algorithm $\mathcal{G}$, *e.g.*, the number of total requests, the *time bound* Δ and the *delay factor* γ. To study the affect of these parameters, we first fix the *time bound* Δ and study the competitive ratios of $\mathcal{G}$ while the *delta factor* γ varies. Then we fix the *delta factor* γ and study the competitive ratios of $\mathcal{G}$ while the *time bound* Δ changes. We set three different total numbers of requests in these experiments to study how the degree of competition affects $\mathcal{G}$'s performance.

In Figure 4, we plot the competitive ratios of $\mathcal{G}$ in various cases while the *delay factor* γ is fixed (see Figure 4 (a)-(c)); we also plot the competitive ratios of algorithm $\mathcal{G}$ in various cases while *time bound* Δ is fixed (see Figure 4 (d)-(f)). In most cases, method $\mathcal{G}$ makes a total profit that is more than 95% of the optimum offline method when $\gamma \simeq \Delta$. Observe that although it is NP-hard to find the optimum offline solution, we implemented a method (omitted here due to space limit) that will find an almost optimum solution (within a factor $1-\epsilon$ for an arbitrarily small $\epsilon > 0$) when the conflict graph is a disk-intersection graph. Our experimental results are much better than the theoretical bound we proved in previous sections.

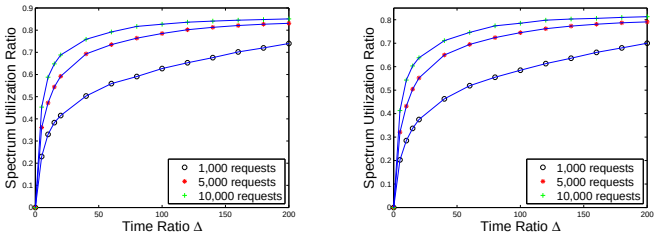
The competitive ratios decrease when Δ increases. This is exactly what our theoretical analysis implies. When $\gamma < \Delta$, the competitive ratio of $\mathcal{G}$ increases significantly when $dfactor$ increases. This implies that increasing the *delay factor* is a good way to achieve better performance if the *delay factor* is relatively small. We also observe that $\mathcal{G}$ almost achieves the optimum offline solution when *delay factor* γ is close to or larger than Δ, which verifies our theoretical bound. Furthermore, the competitive ratio will not improve much when $\gamma > \Delta$. Thus, we recommend to set $\gamma \in [\Delta/2, \Delta]$.

The total number of requests also affect the performance of $\mathcal{G}$. We observe that the competitive ratios decrease when the total number of requests increases. This is because $\mathcal{G}$ is conservative: the weak preemption in $\mathcal{G}$ just tries to satisfy the theoretical bound, which may lose some profit potentially. Thus, when there are more requests, the optimal offline algorithm will make more profit but the profit made by $\mathcal{G}$ will not increase so much.

7.2 Efficiency Ratio of Algorithm $\mathcal{G}$

Recall that to ensure that secondary users will not bid a price lower than their actual values, we do not charge what they bid. For secondary user whose request is accepted, we charge the minimum bid such that the request will still be accepted. Thus, the actual profit made is not the winners' total bids but the winners' total payments. Clearly, the total payments are no more than the total bids. Here we define the *efficiency ratio* as the ratio between the profit made by our mechanism and the winners' total bids computed by our method $\mathcal{G}$. We also study the affect of some parameters such as the number of total requests, the *time bound* Δ and the *delay factor* γ.

In Figure 5, we plot the efficiency ratios of our mechanism is various cases. In (a)-(c), we fix *time bound* Δ and plot the efficiency ratio while *delay factor* γ changes; In (d)-(f), we fix *delay factor* γ and plot the efficiency ratio while time bound Δ changes.



(a) *Delay Factor* $\gamma = 0$, (b) *Delay Factor* $\gamma = 20$

Fig. 6. The spectrum utilization ratios of method $\mathcal{G}$ in various cases.

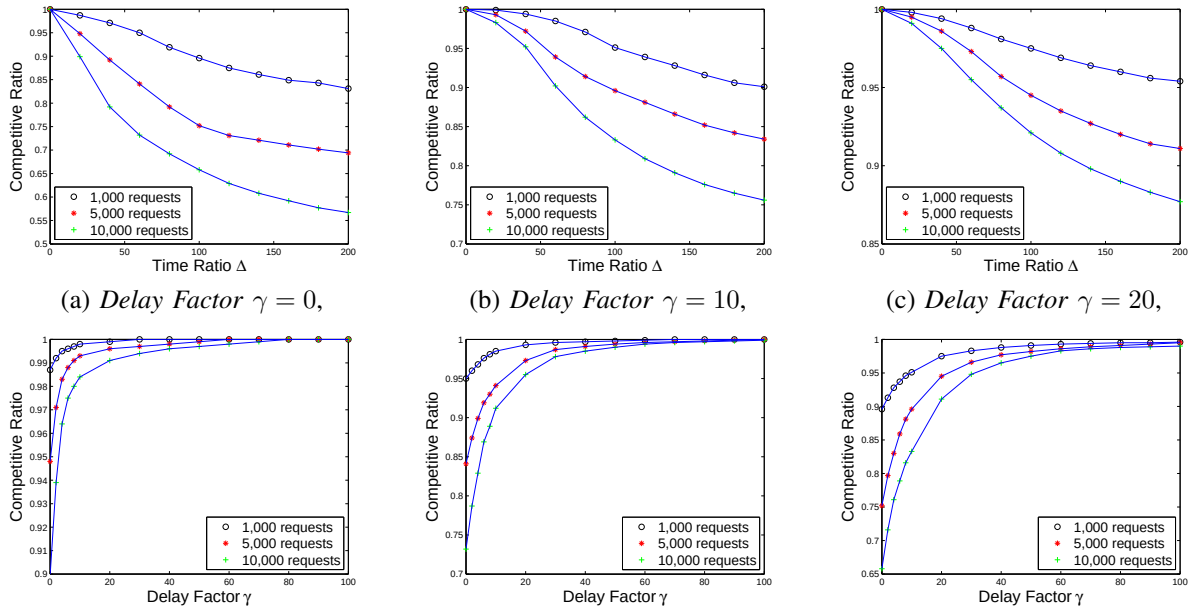
We observe that the efficiency ratios increase when the Δ or the number of total requests increases. In this case, a request has more chance to conflict with other requests. Thus, it is easier to find a replacement which increases the minimum value a winner has to bid. We also find that the efficiency ratios first decrease then increase when the *delay factor* γ increases. Comparing with Figure 4 ((d)-(f)), we know that the efficiency ratios decrease because the competitive ratios increase dramatically when γ is relatively small. The total payments do not increase as fast as the total bids do. When the *delay factor* γ is larger than Δ, the efficiency ratios increase because the competitive ratios increase slowly.

7.3 Spectrum Utilization Ratio of Algorithm $\mathcal{G}$

We also studied the spectrum utilization of our method. The spectrum utilization in a time interval $[1, T]$ under a spectrum allocation method $\mathcal{A}$ is defined as $\sum_{t=1}^{T} \ell_{\mathcal{A}}(t) / \sum_{t=1}^{T} n(t)$, where $\ell_{\mathcal{A}}(t)$ is the number of users who are using the channel (at different locations) at time t without conflict in spectrum allocation produced by $\mathcal{A}$, $n(t)$ is the maximum number of users who can use the channel concurrently without conflict. Figure 6 reports our results. We find that varying γ does not affect the spectrum utilization that much. Observe that our measurement of spectrum utilization compared our method with the offline method that can arbitrarily preempt the spectrum usage. When the system is highly loaded, the spectrum utilization of our method is about 80%, while the spectrum utilization is about 50% for lightly loaded system.

7.4 Compare Algorithm $\mathcal{G}$ with Simple Heuristics

We then compare algorithm $\mathcal{G}$ with two simple greedy algorithms. One greedy algorithm $\mathcal{B}$ always satisfies the virtual spectrum candidate request with the largest $\mathcal{C}_1(t)$ value. When request with larger bid is coming, $\mathcal{B}$ will terminate current assignments with smaller value if necessary. Another greedy algorithm $\mathcal{D}$ always satisfies the virtual spectrum candidate request with the largest $\mathcal{C}_2(t)$ value. When request with larger bid is coming, $\mathcal{D}$ will terminate current assignments with smaller value if necessary. The competitive ratio of these two simple greedy algorithms could be arbitrarily bad theoretically. We conduct simulations to compare them with algorithm $\mathcal{G}$ which is asymptotically optimal theoretically.



(a) *Delay Factor* $\gamma = 0$, (b) *Delay Factor* $\gamma = 10$, (c) *Delay Factor* $\gamma = 20$,

(d) *time bound* $\Delta = 20$, (e) *time bound* $\Delta = 60$, (f) *time bound* $\Delta = 100$.

Fig. 4. The competitive ratios of method $\mathcal{G}$ in various cases.

In Figure 7, we plot the competitive ratios of algorithm $\mathcal{G}$, simple greedy algorithm $\mathcal{B}$ and $\mathcal{D}$ to compare their performances. In Figure 7(a)-(c), we fix *time bound* $\Delta = 20$, and vary the number of total requests for different *delay factors* γ. In Figure 7(d)-(f), we fix *time bound* $\Delta = 100$, and vary *delay factor* γ for different number of total requests. We observe that our algorithm $\mathcal{G}$ outperforms these two simple greedy algorithms significantly in most cases.

The affect of number of total requests is also investigated in Figure 7(a)-(c). We can see that the competitive ratio of algorithm $\mathcal{G}$ is not affected much when the number of total requests increases. On the other hand, the performance of algorithm $\mathcal{B}$ decreases significantly when the number of total requests increases and *delay factor* $\gamma = 0$. The competitive ratio of algorithm $\mathcal{D}$ increases slightly when the number of total requests increases. However, the competitive ratio of algorithm $\mathcal{D}$ is always worse than that of our algorithm $\mathcal{G}$.

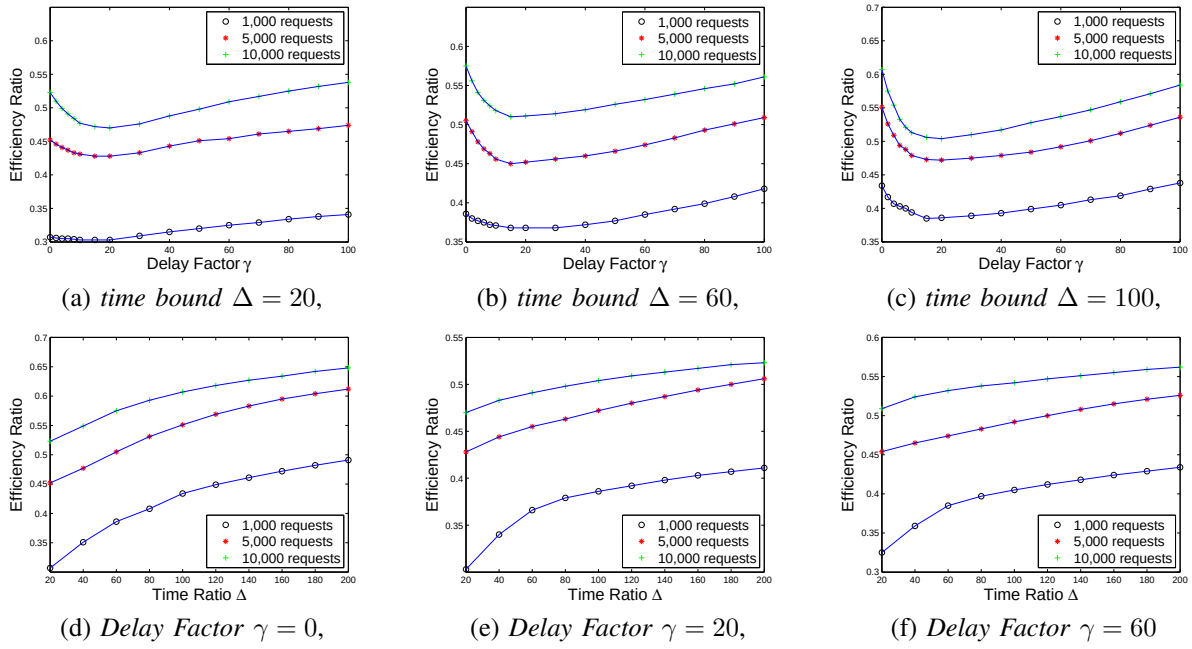
The affect of *delay factor* γ is then investigated in Figure 7(a)-(c). We can see that the performances of all three algorithms increase significantly when the *delay factor* γ increases. When *delay factor* γ is increasing, our algorithm $\mathcal{G}$ is always the best one whose performance is close to the optimum.

8 Literature Reviews

The allocation of spectrums is essentially the combinatorial allocation problem, which have been well studied in the literature [1], [14]. Yuan *et al.* [26] introduced the concept of a time-spectrum block to model spectrum reservation in cognitive radio networks. Li *et al.* [16], [24] designed efficient methods for various dynamic spectrum assignment problems. They also showed how to design truthful mechanism based on those methods. Zhou *et al.* [28] propose a truthful and efficient dynamic spectrum auction system to serve many small players. In [29], Zhou and Zheng designed truthful double spectrum auctions where multiple parties can trade spectrum based on their individual needs. All these results are based on offline models.

In this work, we use the online model which is similar to the online job scheduling problems [8]. Various online scheduling problems focus on optimizing different objective functions. The most common objective function is *makespan*, which is the length of the schedule. Suppose that we are given m *identical* machines, jobs arrive one by one and no preemption is allowed. A number of results have been proved to improve the upper bounds [6], [12], [25] and lower bounds [11]. Closing the gap between the best lower bound (1.88 [11]) and the upper bound (1.9201 [6]) is an open problem. Many authors [7], [18] also investigated the case where preemption is allowed without penalty. Online scheduling problem in which we pay penalty for rejecting jobs was first studied in [2] by Bartal *et al.* and improved later in [10] by Hoogeveen *et al.*. They assume that the penalty is job-dependent only, and is not affected by the preemption time.

When jobs have deadlines, however, it is usually impossible to finish all jobs. Thus, another model aims to maximize the profit or number of completed jobs. There are different variants: preemption-restart, preemption-resume, and preemption-discard. Koren *et al.* [13] gave an algorithm matching the lower bound in [3]. Woeginger [21] studied an online model of maximizing the profit of finished jobs where there is some relationship between the weight and length of job. He provided a 4-competitive algorithm for tight deadline case, and gave a matching lower bound. Hoogeveen *et al.* [9] gave a $\frac{1}{2}$-competitive algorithm which maximizes the number of early jobs. They assume that preemption is allowed while no penalties will be charged. Chrobak *et al.* [4] gave a $\frac{2}{3}$-competitive



(a) *time bound* $\Delta = 20$, (b) *time bound* $\Delta = 60$, (c) *time bound* $\Delta = 100$,

(d) *Delay Factor* $\gamma = 0$, (e) *Delay Factor* $\gamma = 20$, (f) *Delay Factor* $\gamma = 60$

Fig. 5. The efficiency ratios of our mechanism in various cases.

algorithm which maximizes the number of satisfied jobs that have uniform length in the *preemption-restart* model. Zheng *et al.* [27] studied the *preemption-restart* model where a penalty is the weight of the preempted job.

The work that is most similar to our work is a recent result by Constantin *et al.* [5]. They proposed and studied a simple model for auctioning ad slot reservations in advance. A seller will display a set of slots at some point T in the future. Until T, bidders arrive sequentially and place a bid on the slots they are interested in. The seller must decide immediately whether or not to grant a reservation. Their model allows the seller to cancel at any time any reservation made earlier with a penalty proportional to the bid value. The major differences between our model and their model are as follows. Their model considers online requests and *offline* services. The services (ad slots) start from a *fixed* time and last for one unit time. And the preemptions happen before services start. In our model, the services (spectrum usage) can start from *any* time, lasting for an *arbitrary* duration (subject to time bound Δ), and the preemptions happen after the spectrum usages are assigned. We also considered the spatial reuse of spectrum resources.

9 Conclusions

In this paper, we studied online spectrum allocation for wireless networks where a set of secondary users will bid for leasing a spectrum channel for certain time duration in different locations. For a number of variants, we designed efficient online scheduling algorithms and analytically showed that the competitive ratios of our methods are within small constant factors of the optimum. Especially, when γ is around the maximum requested time duration Δ, our algorithm results in a profit that is almost optimum. Our extensive simulations show that our methods perform extremely well in practice.

We showed that no user will bid lower than its willing payment under our mechanism TOFU. It remains open to design a truthful mechanism for online spectrum auction in this setting. It is also interesting to extend our mechanism to deal with different models, such as OFDM networks, and when we know more information about requests, such as the distributions of bids, timeslots requested, and arrival times.

References

[1] Archer, A., Papadimitriou, C., Talwar, K., and Tardos, E. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics 1*, 2 (2004), 129–150.

[2] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., and Stougie, L. Multiprocessor scheduling with rejection. In *ACM SODA* (1996), SIAM, pp. 95–103.

[3] Baruah, S., Koren, G., Mishra, B., Raghunathan, A., Rosier, L., and Shasha, D. On-line scheduling in the presence of overload. *IEEE FOCS* (1991), 100–110.

[4] Chrobak, M., Jawor, W., Sgall, J., and Tichý, T. Online scheduling of equal-length jobs: Randomization and restarts help. *SIAM J. Comput. 36*, 6 (2007), 1709–1728.

[5] Constantin, F., Feldman, J., Muthukrishnan, S., and Pal, M. An online mechanism for ad slot reservations with cancellations. In *ACM-SIAM SODA* (2009), pp. 1265–1274.

[6] Fleischer, R., and Wahl, M. On-line scheduling revisited. *J. of Scheduling 3*, 343-353 (2000).

[7] Goemans, M. X., Wein, J. M., and Williamson, D. P. A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Operations Research Letters 26*, 4 (2004), 149–154.

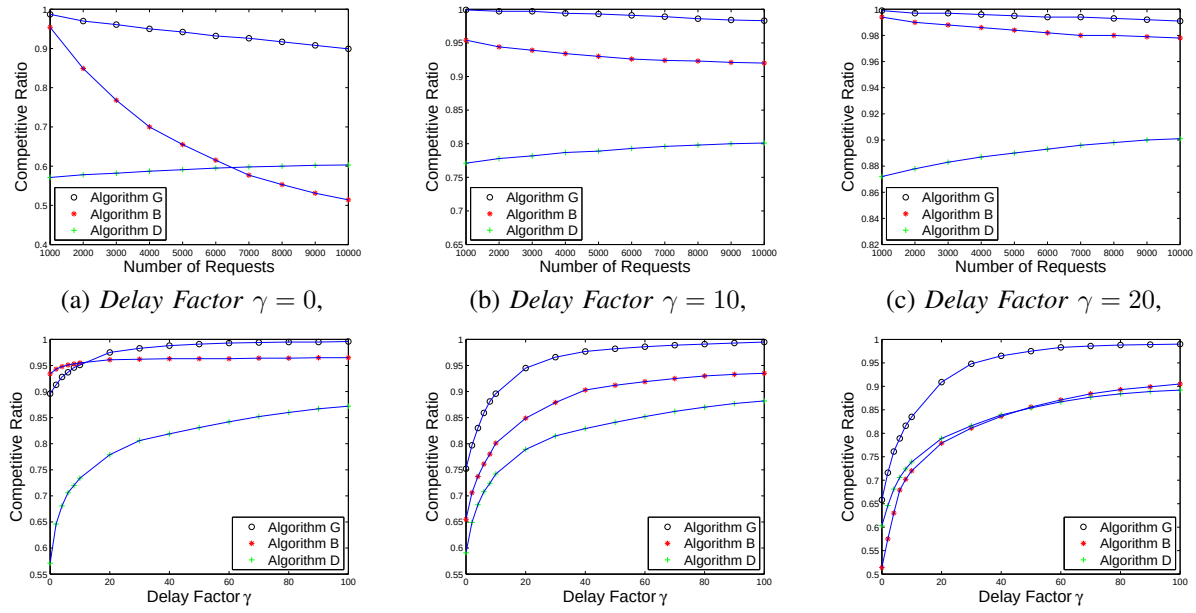
[8] Graham, R. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*, 1563-1581 (1966).

[9] Hoogeveen, H., Potts, C. N., and Woeginger, G. J. On-line scheduling on a single machine: maximizing the number of early jobs. *Operations Research Letters 27*, 5 (2000), 193–197.

[10] Hoogeveen, H., Skutella, M., and Woeginger, G. J. Preemptive scheduling with rejection. In *Algorithms - ESA 2000* (2002), pp. 268–277.

[11] John F. Rudin, I., and Chandrasekaran, R. Improved bounds for the online scheduling problem. *SIAM J. Comput. 32*, 3 (2003), 717–735.

[12] Karger, D. R., Phillips, S. J., and Torng, E. A better algorithm for an ancient scheduling problem. In *ACM SODA* (1994), pp. 132–140.



(a) *Delay Factor* $\gamma = 0$, (b) *Delay Factor* $\gamma = 10$, (c) *Delay Factor* $\gamma = 20$,

(d) 1,000 requests, (e) 5,000 requests, (f) 10,000 requests

Fig. 7. Compare algorithm $\mathcal{G}$ with two simple greedy algorithms.

[13] KOREN, G., AND SHASHA, D. Dover: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM J. Comput. 24*, 2 (1995), 318–339.

[14] LEHMANN, D. J., O'CALLAGHAN, L. I., AND SHOHAM, Y. Truth revelation in approximately efficient combinatorial auctions. In *ACM Conference on Electronic Commerce* (1999), pp. 96–102.

[15] LI, X.-Y., AND WANG, Y. Simple approximation algorithms and PTASs for various problems in wireless ad hoc networks. *Journal of Parallel and Distributed Computing* (2006), 515 – 530.

[16] LI, X.-Y., XU, P., TANG, S., AND CHU, X. Spectrum bidding in wireless networks and related. In *COCOON* (2008), pp. 558–567.

[17] MYERSON, R. Optimal auction design. *Mathematics of operations research 6*, 1 (1981), 58–73.

[18] PHILLIPS, C. A., STEIN, C., AND WEIN, J. Scheduling jobs that arrive over time (extended abstract). In *WADS* (1995), Springer-Verlag, pp. 86–97.

[19] STINE, J. A. Spectrum management: The killer application of ad hoc and mesh networking. In *IEEE DySPAN* (2005).

[20] WANG, S., XU, P., XU, X.-H., TANG, S., LI, X.-Y., AND LIU, X. TODA: Truthful online double auction for spectrum allocation. In *IEEE DySpan* (2010).

[21] WOEGINGER, G. J. On-line scheduling of jobs with fixed start and end times. *Theor. Comput. Sci. 130*, 1 (1994), 5–16.

[22] XU, P., AND LI, X. SOFA: Strategyproof Online Frequency Allocation for Multihop Wireless Networks. In *ISAAC* (2009), Springer-Verlag, p. 311.

[23] XU, P., AND LI, X.-Y. OASIS: Online algorithm for spectrum allocation in multihop wireless networks. Invited to *Algorithmica* (2010), in preparation.

[24] XU, P., LI, X.-Y., TANG, S., AND ZHAO, J. Efficient and strategyproof spectrum allocations in multi-channel wireless networks. *IEEE Transactions on Computers* (Nov 2009).

[25] Y., B., A., F., H., K., AND R., V. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences 51*, 359-366 (1995).

[26] YUAN, Y., BAHL, P., CHANDRA, R., MOSCIBRODA, T., AND WU, Y. Allocating dynamic time-spectrum blocks in cognitive radio networks. In *ACM MobiHoc* (2007), pp. 130–139.

[27] ZHENG, F., XU, Y., AND ZHANG, E. On-line production order scheduling with preemption penalties. *Journal of Combinatorial Optimization 13* (2007), 189–204.

[28] ZHOU, X., GANDHI, S., SURI, S., AND ZHENG, H. eBay in the Sky: strategy-proof wireless spectrum auctions. In *ACM MobiCom* (2008), pp. 2–13.

[29] ZHOU, X., AND ZHENG, H. TRUST: A general framework for truthful double spectrum auctions. In *IEEE INFOCOM* (2009).

APPENDIX

.1 Proof of Theorem 2

Proof: Assume that there is an online method $\mathcal{A}$ with the best possible competitive ratio $\varrho > \theta$. Assume that at time 0, there is only one job $\mathbf{e}_0 = (b_0 = 1, 0, \alpha, \Delta)$ arrived and is the only job known at time γ. Then $\mathcal{A}$ has to accept $\mathbf{e}_0$, in case $\mathbf{e}_0$ is the only request. At time γ, after the decision is made, another job $\mathbf{e}_1 = (b_1, \gamma, \alpha, \gamma)$ arrives and no more jobs arrived before time 2γ. Here b_1 is carefully chosen such that $b_1 + \frac{\gamma}{\Delta} b_0 - \beta \frac{\Delta-\gamma}{\Delta} b_0 = \theta b_0 < \varrho \cdot b_0$. Thus, at time 2γ, $\mathcal{A}$ cannot admit $\mathbf{e}_1$ by preempting $\mathbf{e}_0$ because of the choice of b_1. If $\mathcal{A}$ skips a request $\mathbf{e}_{i-1}$, at time $i \cdot \gamma$, we will release a new job $\mathbf{e}_i = (b_i, \gamma, \alpha, \gamma)$ with

$$b_i + \frac{i\gamma}{\Delta} b_0 - \beta \frac{\Delta - i\gamma}{\Delta} b_0 = \theta b_0 < \varrho \cdot b_0.$$

This is same as

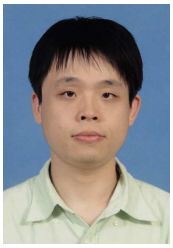
$$b_i = \theta b_0 - \frac{i\gamma}{\Delta} b_0 + \beta \frac{\Delta - i\gamma}{\Delta} b_0 = (\theta + \beta) b_0 - \frac{\gamma}{\Delta}(1+\beta) b_0 \cdot i$$

We repeat this until either $\mathcal{A}$ accepts some job $\mathbf{e}_i$, or i reaches an integer value $k \le \lceil \frac{\Delta}{\gamma} \rceil - 1$. To ensure that $b_i \ge 0$, we also need $i \le k \le \frac{\theta+\beta}{1+\beta} \frac{\Delta}{\gamma}$.

It is easy to show that algorithm $\mathcal{A}$ needs to decline all jobs $\mathbf{e}_i$, for $i \in [1, k]$; otherwise, its competitive ratio is less than ϱ. On the other hand, a better solution may be to ignore $\mathbf{e}_0$ and admit $\mathbf{e}_i$ for $i \in [1, k]$, with total profit $\sum_{i=1}^{k} b_i$. Since $\mathcal{A}$ has a competitive ratio ϱ, we then need $\sum_{i=1}^{k} b_i \le b_0/\varrho \le b_0/\theta$. Consequently, we need

$$\sum_{i=1}^{k} \left((\theta + \beta) b_0 - \frac{\gamma}{\Delta}(1+\beta) b_0 \cdot i \right) \le b_0/\theta$$

Then the maximum possible value for θ is $\theta \leq \frac{2}{a+\sqrt{a^2+4k}}$ where $a = \beta k - (1+\beta)\frac{\gamma}{\Delta}\frac{k(k+1)}{2}$. We then choose $k \in [1, \lceil \frac{\Delta}{\gamma} \rceil - 1]$ such that $\frac{2}{a+\sqrt{a^2+4k}}$ is minimized. For example, we can choose $k = \lfloor \frac{\beta}{1+\beta}\frac{\Delta}{\gamma} \rfloor - 1$. This implies that an upper bound $\frac{2}{a+\sqrt{a^2+4k}} < 1/a \leq \frac{1}{c\beta}$ on the competitive ratio. □



Ping Xu is a Computer Science PhD student at Illinois Institute of Technology. He received B.Eng and M.E. from Shanghai Jiao Tong University, P.R.China, 2003 and 2006 respectively. His research interests span wireless networks, game theoretical study of networks, optimization and security in wireless network. He also worked on cognitive radio networks as a Summer Intern at Microsoft Research Asia in 2008.

Dr. Xiang-Yang Li has been an Associate Professor since 2006 and Assistant Professor of Computer Science at the Illinois Institute of Technology from 2000 to 2006. He was a visiting professor of Microsoft Research Asia from May 2007 to August 2008. He hold MS (2000) and PhD (2001) degree at Computer Science from University of Illinois at Urbana-Champaign. He received both B.Eng. at Computer Science and Bachelor degree at Business Management from Tsinghua University, P.R. China in 1995. His research interests span wireless ad hoc and sensor networks, computational geometry, and algorithms. He serves as an Editor of "IEEE Transitional on Parallel and Distributed Systems (TPDS)", from 2010; an Editor of "Networks: An International Journal" from 2009, and Advisory Board of "Ad Hoc & Sensor Wireless Networks: An International Journal", from 2005. He was a guest editor of special issues for "ACM Mobile Networks and Applications", "IEEE Journal on Selected Areas in Communications", and several other journals.