Efficient and Strategyproof Spectrum Allocations in Multichannel Wireless Networks

Ping Xu, Xiang-Yang Li, *Senior Member*, *IEEE*, ShaoJie Tang, and JiZhong Zhao, *Member*, *IEEE Computer Society*

Abstract—In this paper, we study the spectrum assignment problem for wireless access networks. We assume that each secondary user will bid a certain value for exclusive usage of some spectrum channels for a certain time period or for a certain time duration. A secondary user may also require the exclusive usage of a subset of channels, or require the exclusive usage of a certain number of channels. Thus, several versions of problems are formulated under various different assumptions. For the majority of problems, we design PTAS or efficient constant-approximation algorithms such that overall profit is maximized. Here, the profit is defined as the total bids of all satisfied secondary users. As a side product of our algorithms, we are able to show that a previously studied Scheduling Split Interval Problem (SSIP) [2], in which each job is composed of t intervals, cannot be approximated within $O(t^{1-\epsilon})$ for any small $\epsilon > 0$ unless $\mathrm{NP} = \mathrm{ZPP}$. Opportunistic spectrum usage, although a promising technology, could suffer from the selfish behavior of secondary users. In order to improve opportunistic spectrum usage, we then propose to combine the game theory with wireless modeling. We show how to design a truthful mechanism based on all of these algorithms such that the best strategy of each secondary user to maximize its own profit is to truthfully report its actual bid.

Index Terms—Wireless networks, spectrum, disk graph, interval graph, PTAS, approximation, strategyproof.

1 Introduction

Wireless technology is expected to play a bigger and more fundamental role in the new Internet than it has today. The radio frequency spectrum has been chronically regulated with static spectrum allocation policies since the early $20th$ century. With the recent fast growing spectrum-based services and devices, the remaining spectrum available for future wireless services is being exhausted, known as the *spectrum scarcity problem*. The current fixed spectrum allocation scheme leads to significant spectrum *white spaces* where many allocated spectrum blocks are used only in certain geographical areas and/or in brief periods of time. A huge amount of precious spectrum (below 5 GHz), perfect for wireless communications that is worth billions of dollars, sit there silently. Recognizing that the traditional spectrum management process can stifle innovation, and it is difficult to provide a certain quality of service (QoS) for systems operated in unlicensed spectrum, the FCC has proposed new spectrum management models and the use of a measure of interference temperature. Current spectrum management methods include command and control (e.g., for public safety), exclusive usage based on license (e.g., for cellular communication), commons (e.g., ISM bands), interference temperature (also called opportunistic usage), and fast command and control [24].

One promising technology is the opportunistic spectrum usage. In opportunistic spectrum usage, the secondary users observe the channel availability dynamically and explore it opportunistically. Secondary users are cognitive devices that can sense the environment and adapt to appropriate frequency, power, and transmission schemes. They can opportunistically access unused spectrum vacated by idle primary users, who have strict priority on spectrum access and will share spectrum with others under certain protections. While opportunistic spectrum has several advantages, it suffers from the selfish behavior of the secondary users. For secondary users, usually they are selfish. Obtaining more spectrum could mean more benefit economically. Therefore, it is difficult, if not possible, to require each secondary user faithfully conform to a distributed channel assignment method that may reduce its own chance to use the spectrum, for the overall system benefit. It is then reasonable and appropriate to model them as *rational* agents. Thus, we propose to combine the game theory [20], specifically, mechanism design theory with wireless communication modeling. More specifically, we study how to share the spectrum and how to charge the secondary users such that the *overall social benefit* or *profit by spectrum owner* is maximized even in the presence of selfish behavior.

In this paper, we assume that each secondary user v_i will bid a value b_i for usage of some channels vacated by the primary users. A secondary user may also imply (by its locations) or specify some additional constraints for the usage of the spectrums, such as the region where it will use the channels.

The first constraint is where the spectrums will be used. We assume that each user v_i will specify a two-dimensional *space requirement*, which is typically a disk $D(v_i, r_i)$ centered at node v_i with a radius r_i. In other words, secondary user v_i wants an exclusive usage of $\mathcal{F}_i$ inside the disk $D(v_i, r_i)$.

- *P. Xu, X.-Y. Li, and S. Tang are with the Department of Computer Science, Illinois Institute of Technology, 10 West 31st Street, Chicago, IL 60616. E-mail: {pxu3, stang7}@iit.edu, xli@cs.iit.edu.*
- *J. Zhao is with the Department of Computer Science and Tech., Xi'an Jiaotong University, China. E-mail: zjz@mail.xjtu.edu.cn.*

Manuscript received 13 Sept. 2008; revised 17 Feb. 2009; accepted 23 Nov. 2009; published online 6 Dec. 2010.
Recommended for acceptance by V. Leung.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-09-0465.
Digital Object Identifier no. 10.1109/TC.2010.241.

Notice that the requirement that the space required by each secondary user is a disk is immaterial to results presented in this paper. Our results will hold as along as regions required by all secondary users have similar size and constant-bounded aspect ratio. Here, the aspect ratio of a region (without hole) is defined as its area divided by the square of its perimeter's length. For simplicity, all our proofs assume that the space requirement by a secondary user is a disk with unit radius.

The second constraint is the time constraint which specifies a time interval $[s_i, e_i]$ or a time duration d_i that node v_i wants an exclusive usage of $\mathcal{F}_i$ for that time interval or duration. In other words, generally, we assume that for a bundle $\mathcal{F}_i \subseteq \mathcal{F}$ of channels, a *space requirement* specified by a disk $D(v_i, r_i)$, and a time constraint (time interval $[s_i, e_i]$ or a duration d_i) user v_i places a bid b_i when it is single-minded. Otherwise, it could bid $b_{i,j}$ for each channel $\mathbf{f}_j$ separately with possible different time constraints for each channel.

The central authority will design a mechanism which, upon receiving bidding requests from different secondary users, computes how to share the channels among the secondary users, and computes a schedule of channels for the secondary users subject to the time-space restriction: no two secondary users that are interfered with each other are given the same channel at anytime. Last but not the least, the mechanism should compute how much we should charge the secondary users. The difficulty of this problem arises from two aspects. First, the correlation of the time and space among requests from secondary users introduces high complexity compared with traditional auctions [3], [6], [8], [16], [21], [25]. The problem is *NP-complete* by the reduction from the problem the maximum weighted independent set of disk graphs even the time requirements of all agents over all channels are the same or from the set packing problem even when all nodes are colocated. Interference along already makes numerous related problems untractable, not to mention the scheduling of the channels.

The main contributions of this paper are as follows: First, we design efficient algorithms to allocate channels in different cases such that the social efficiency (defined as the total valuations of all satisfied bidders) are approximately maximized. Specifically, we design a polynomial time approximation scheme (PTAS) for maximizing the social efficiency when all secondary users are colocated, not single-minded, but could have mixed time requirements. Here, a secondary user is single-minded if it requests a subset of channels and the assignment is valid only if the *exact* subset of channels is allocated to this user at the exact requested time. All users are *colocated* if the space requirements of all users overlap at some point. We design PTAS for maximizing the social efficiency when 1) all secondary users are not single-minded (they can be allocated any number of channels from their required channels), and 2) the time requirement by each secondary is specified by a time interval. Constant approximation algorithms are designed when users are not single-minded but with possible mixed time requirements.

When users are single-minded and each user will bid for a subset of channels, the problem does not have an approximation ratio within $m^{1/2-\varepsilon}$ for any $\varepsilon > 0$, unless NP = ZPP,[1] where m is the total number of channels in $\mathcal{F}$ [9], [16]. In this case, we then design approximation algorithms with ratio $\Theta(\sqrt{m})$ when the space requirements by all secondary users are unit disks. As a side product of our algorithms, we are able to show that a previously studied Scheduling Split Interval Problem (SSIP) [2] cannot be approximated within $O(t^{1-\epsilon})$ for any small $\epsilon > 0$ unless $NP = ZPP$. There is a $2t$ approximation method for SSIP [2].

Based on these approximation algorithms, we then design strategyproof mechanisms to charge the secondary users such that, under our mechanisms, each secondary user will maximize its own profit if it truthfully reported its own bid, no matter what others will do. We essentially show that our approximation algorithms satisfy a monotone property: if a secondary user is satisfied with bid b_i then it will still be satisfied with a bid $b_i' > b_i$ while the bids of all other secondary users remain the same. Then, a strategyproof mechanism can always be designed, in couple with our algorithms with monotone property, based on results from [13]. Notice that here we focus on strategyproof mechanism solution concept to address the noncooperative game among secondary users. A number of other interesting solution concepts could be used to study these games, such as Nash equilibria.

The rest of the paper is organized as follows: In Section 2, we define in detail the problems to be studied in this paper. Then, we review related results on those spectrum assignment problems in Section 3. From Sections 4 to 7, we discuss algorithms for several versions of problems described in Section 2. In Section 8, we show how to design strategyproof mechanisms based on the algorithms we designed. And we do simulations in Section 9 to study the performance of our algorithms in experiment. At last, we conclude the paper in Section 10 with the discussion of some possible future works.

2 Preliminaries

2.1 Network Model

Consider a wireless network system formed by some primary users $\mathcal{U} = \{u_1, u_2, \ldots, u_p\}$ who hold the right of some spectrum channels, secondary users $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ who want to lease the right to use some channels in some region for some time period. In certain applications, each secondary user v_i may provide service to κ_i clients $\mathcal{Z}_i = \{z_{i,1}, z_{i,2}, \ldots, z_{i,\kappa_i}\}$ within a geometry region. Here, u_j, v_i, and $z_{i,k}$ also represent the fixed two-dimensional geometry location of these users. For simplicity, we treat all primary users as one unified central authority. Let $\mathcal{F} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$ be the set of m frequencies that can be used by some secondary users for a given time interval $[0, T]$. For some wireless network systems, it is possible that the primary users will only lease a spectrum frequency for a certain time interval in a certain geographical region. If this is the case, we assume that for each $\mathbf{f}_i \in \mathcal{F}$, we associate it with a region Ω_i

1. Zero-error Probabilistic Polynomial time (ZPP) is the complexity class of problems for which a probabilistic Turing machine exists with these properties: 1) It always returns the correct YES or NO answer; and 2) The running time could be unbounded, but is polynomial on average for any input.

and a set of time intervals $\mathcal{T}_i$ that it is available. We assume that every channel will be available everywhere and at all time slots. Our results can easily deal with a general Ω_i and $\mathcal{T}_i$.

We divide the time into multiple time intervals and make allocation on each interval at the beginning of that interval. New secondary users may join the system. But they cannot lease channels until next allocation begins. Existing secondary users may leave the system at anytime. Observe that the total time duration required by a user may last for several time intervals. Thus, at the beginning of each time interval, some combination of channels and locations could be hold by users from previous allocations. Our channel allocation schemes will not use these channels at some specific regions.

We assume that a secondary user v_i may wish to lease a set of channels $\mathcal{F}_i \in \mathcal{F}$. For a bidding, the secondary user will also specify two additional constraints: *space requirement* and time condition. We assume that each user v_i will specify a two-dimensional region which is typically a disk $D(v_i, r_i)$ centered at node v_i with a radius r_i. In other words, secondary user v_i wants an exclusive usage of $\mathcal{F}_i$ inside the disk $D(v_i, r_i)$. Additionally, user v_i also specifies a *time interval* $[s_i, e_i]$ or a *time duration* d_i that node v_i wants an exclusive usage of $\mathcal{F}_i$ for that time interval or duration. Here, it is assumed that $0 \leq s_i < e_i \leq T$ and $0 < d_i \leq T$. A time interval requirement implies that the secondary user wants the exclusive usage of the channels $\mathcal{F}_i$ from time s_i to time e_i. A time duration requirement d_i implies that the secondary user wants the exclusive user of channels for a time that lasted continuously for d_i time units. It can be started from anytime. Generally, we use T_i to denote the time constraint of user v_i, where T_i is either $[s_i, e_i]$ or a scalar $d_i > 0$.

Two different models of secondary users will be studied in this paper. The first model assumes that every secondary user is *single-minded*: when a secondary user v_i bids for $\mathcal{F}_i$, the valuation of v_i over an assignment is 0 if the assignment does not satisfy *all* requirements and not all frequencies in $\mathcal{F}_i$ are allocated. The secondary user v_i will be called *flexible* if it will pay the central authority based on the frequencies allocated. For a flexible user v_i, we assume that for each channel $\mathbf{f}_j \in \mathcal{F}_i$, user v_i will bid $b_{i,j}$ for the right to use the channel $\mathbf{f}_j$ for a certain time requirement and geographical requirement. In this case, we use $b_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,m}\}$ to denote the bid vector of user i, where $b_{i,j} = 0$ if v_i did not bid for $\mathbf{f}_j$. Thus, a bidding by a user v_i will be written as follows:

$$B_i = [b_i, \mathcal{F}_i, D(v_i, r_i), T_i].$$

Upon receiving the bids from secondary users, the central authority decides an allocation method:

1. An allocation method $X = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in \{0, 1\}$ denotes whether user v_i's bid will be satisfied, and also a time interval $[\mathbf{s}_i, \mathbf{e}_i]$ with $\mathbf{e}_i - \mathbf{s}_i = d_i$ when user v_i required a time duration d_i in the bid B_i.
2. A payment scheme $P = \{p_1, p_2, \ldots, p_n\}$ where p_i denotes how much user v_i should pay.

The allocation must be conflict free among satisfied bids. Here, two bids B_i and B_j conflict if $\mathcal{F}_i \cap \mathcal{F}_j \neq \emptyset$, $D(v_i, r_i) \cap D(v_j, r_j) \neq \emptyset$, and $[\mathbf{s}_i, \mathbf{e}_i] \cap [\mathbf{s}_j, \mathbf{e}_j] \neq \emptyset$. The objective of an allocation is to maximize $\sum_{i=1}^{m} x_i b_i$. For simplicity, given a set of bids Y, we use $\omega(Y)$ to denote the total weight of bids in Y, i.e., $\sum_{B_i \in Y} b_i$.

2.2 Introduction to Game Theory and Mechanism Design

A game includes a finite set of players (decision makers), a set of actions, and a set of utility functions that players wish to maximize. Other games may include additional components, such as the information available to each player and communication mechanisms. For example, in a sequential game, one player makes his decision before others do so and the others may have some information of the first decision which will affects their strategies. And in a repeated game, players are allowed to observe the actions of the other players, remember past actions, and attempts to predict future actions of players.

Traditional applications of game theory attempt to find equilibria in these games. In an equilibrium each player has adopted a strategy that they do not want to change. And many equilibrium concepts, for example, Nash equilibrium, have been developed. Equilibrium analysis mainly attempts to mathematically capture behavior in strategic situations, in which an individual's success in making decision depends on the decision of others.

On the other hand, mechanism design, another branch of game theory, is the study of designing rules of a game to achieve a specific outcome [17], [18]. Unlike equilibrium analysis, mechanism design mainly focus on how to design rules to achieve a maximized social benefit even though each user is selfish and how to design a mechanism such that selfish users have incentive to follow the mechanism truthfully. In a word, mechanism design uses the techniques of games theory to develop rules for a game.

A standard model for mechanism design is as following: There are n agents $1, 2, \ldots, n$. Each agent i has some private information t_i, which is called type, and a set of strategy A_i from which it can choose. For each input vector $a = (a_1, a_2, \ldots, a_n)$ where $a_i \in A_i$ is the strategy played by agent i, the mechanism $M = (O, P)$ computers and output $o = O(a)$ and a payment vector $p(a) = (p_1(a), p_2(a), \ldots, p_n(a))$. Here, the payment $p_i(a)$ is the money we charge agent i depending on the strategies vector a.

In our case, each secondary user v_i is an agent i. The valuation of request by v_i is private type t_i. Moreover, bid b_i, frequencies $\mathcal{F}_i$ and region $D(v_i, r_i)$ are the strategy played by secondary user v_i. We need to design a strategyproof mechanism $M = (O, P)$, i.e., for each secondary user, reporting its valuation truthfully will maximize its profit. Strategyproof mechanisms should satisfy both incentive compatible (IC) and incentive rational (IR) properties. An incentive compatible mechanism is a mechanism in which an agent will maximize its utility by reporting its private type truthfully. An individual rational mechanism is a mechanism in which the utilities of the agents participating in the output is not less than that of those no participating in the output. In Section 8, we will show how to design strategyproof mechanisms based on the algorithms we designed.

2.3 Problems Formulation

In this paper, we study several versions of spectrum assignment problem by separately assuming whether the

secondary users are single-minded or not, whether their required regions $D(v_i, r_i)$ overlap or not, whether all secondary users ask for a fixed time interval, or all users ask for some duration, or the time requirements are mixed among users. Then, we design strategyproof mechanisms based on those approximation algorithms. Based on those algorithms and performance analysis, the network system could decide an assumption combination which is feasible in practice and results in more benefits theoretically.

For notational convenience, we use CRT to denote a problem, where

- C denotes choice of spectrums by secondary users. Here, C will be either S (denoting that some secondary users require a subset of channels and they are single-minded), or F (denoting that every secondary user will bid separately for each channel and is flexible), or Y (denoting that there is only one channel available in the system, i.e., $m = 1$).
- R denotes region requirement by secondary users. Here, R will be either O (denoting that the required regions overlap at some single point) or U (denoting that the required regions are denoted by unit disks in 2D).
- T denotes time requirement by secondary users. Here, T will be either I (denoting that the required time by every user v_i will be an interval $[s_i, e_i]$) or D (denoting that the required time by every user v_i will be a duration d_i) or M (denoting that some users required a time interval and some users required a time duration).

For example, problem SUI represents the case that each user v_i will bid for a subset of channels $\mathcal{F}_i$ and is single-minded, will require a unit disk region $D(v_i, 1)$, and a fixed time interval $[s_i, e_i]$. For each problem where every secondary user will bid separately for each channel (called problem with $\mathrm{C} = \mathrm{Y}$), we have a corresponding $\mathrm{C} = \mathrm{Y}$ problem when considering each user requires k channels as k duplicated users that each of them requires one channel. Therefore, we don't discuss these $\mathrm{C} = \mathrm{F}$ problems as they are special cases of $\mathrm{C} = \mathrm{Y}$ problems.

Notice that each version of problem includes two parts, the allocation methods and payment scheme as mentioned in Section 2.1. In rest of this paper, we mainly focus the allocation methods. Therefore, when we mention a version of problem, we indicate its allocation problem without special explanation.

We would like to point out some relations between the complexity of different problems. For any problem in which users have a mixed time requirement, i.e., problems in the format $\mathrm{T} = \mathrm{M}$, it can be solved if the corresponding problems in the format $\mathrm{T} = \mathrm{D}$ and $\mathrm{T} = \mathrm{I}$ are solved. For example, problem SUM can be solved if problem SUD and problem SUI are solved. More details and performance analysis will be given in Section 4.

Some versions of the problems turn out to be some well-studied problems in the literature and some well-studied problems turn out to be a special case of the above problems. Problem YOI is essentially to compute the maximum weighted independent set in interval graphs, which has a well-known polynomial time algorithm by dynamic programming. Problem YOD is essentially a knapsack problem, which has a well-known PTAS [5].

In this paper, we will mainly focus on the problems YOM, YUI, YUD, and SUI. Problems YUM can be solved by Algorithm 2 based on our methods for problem YUI and YUD. Also observe that SOI, SOD, and SOM are special cases of SUI, SUD, and SUM, respectively. And SUM can be solved based on our method for SUI and a method for SUD. Thus, the only challenging question that is left unsolved is SUD.

3 Literature Reviews

In this section, we briefly review literature related to the problems studied in following sections. The allocation of spectrums to users is essentially the combinatorial auctions, which have been well-studied in the literature and a number of algorithms [16], [21] have been proposed. And game theory is also applied widely in resource management problems [1], [14]. Specifically, the problems we will discuss are at the intersection of a lot of famous problems, such as knapsack, rectangle packing, set packing, combinatorial auctions, and so on. Here, we review results for some of these problems.

Knapsack problem, which is same as simple problem YOD in our definition, has a classic fully polynomial time approximation scheme (FPTAS) by the means of rounding [5], [22]. However, we cannot design a strategyproof mechanism using this FPTAS since it is not monotone. An alternative rounding scheme was proposed by Briest in [4], which gave a new rounding scheme leading to a monotone FPTAS for knapsack problem.

Rectangle packing problem is that, there is a set of small rectangles with specific profits and a big rectangle, we try to packing small rectangles into the big rectangle with maximal total profit and no overlap among small rectangles. In [11], Jansen and Zhang presented a $(2 + \epsilon)$- approximation algorithm for rectangle packing problem which is similar with our problem YUI. Specifically, unit-height rectangle packing problem is a special case of YUI if we don't consider intersections in space. Kovaleva described a PTAS for unit-height rectangle packing problem in [15]. We extend this PTAS to a PTAS for problem YUI as described in following section.

For the problem SUI such that secondary users require a subset of channels and they are single-minded, it is a special case of set packing problem. In [9], Hastad proved that set packing problem cannot be approximated within $m^{\frac{1}{2}-\epsilon}$, unless $NP = ZPP$. Another special case of set packing problem is Scheduling Split Intervals Problem, which considers how to schedule weighted jobs, where each job is given as a group of nonintersecting t segments on the real line. Bar-Yehuda et al. [2] gave an algorithm with $2t$-approximation ratio. We show how to convert the problem SUI to SSIP and our method achieves an approximation ratio, which is asymptotically tight, in following sections.

In this paper, we also discuss how to design truthful mechanism based on our algorithms to compute payment scheme. The field of mechanism design [17], [18] aims to study how privately known preferences of many people can be aggregated toward a "social choice." Some tools of mechanism design [7], [19] were applied to optimization problems that involve selfish participants, which is similar to our problem. When users arrive in an online fashion and

the system need make a decision immediately or within a time limit, recently Xu et al. [26], [27] developed some mechanisms that can ensure certain truthfulness and performance guarantee under some assumptions. Using some concepts of Myerson auction, Jia et al. [12] designed truthful spectrum auction for spectrum access to (approximately) maximize the revenue.

4 Algorithm for Problem YOM

In this section, we design an approximation algorithm for problem YOM, which is equivalent to the following problem. Assume that we are given n_1 secondary users whose time requirements are time intervals $[s_i, e_i]$ for $0 \le s_i < e_i \le T, 1 \le i \le n_1$, with a weight $b_i > 0$ for each interval; and n_2 secondary users whose time requirements are time durations d_i for $0 < d_i \le T$, where $n_1 < i \le n_2 + n_1$, with a weight $b_i > 0$ for each duration. Here, $n = n_1 + n_2$. Our objective is to select some secondary users such that their requests can be satisfied in time segment $[0, T]$ without intersections while the total weight is maximized. Algorithm 1 provides our method that essentially uses algorithms for problem YOD, and YOI.

Algorithm 1. Constant Approximation for YOM
Input: n_1 secondary users requiring time intervals and n_2 secondary users requiring time duration.
Output: Some secondary users that can be satisfied in time segment $[0, T]$ without intersections.

1. Find the optimal solution with the first n_1 secondary users by using dynamic programming. The detail of the dynamic programming is omitted here due to its simplicity and space limit. Let S_1 denote the solution.
2. Find the approximated optimal solution with all users that request time duration by using the PTAS for knapsack problem, e.g., [5], [22]. Let S_2 denote the solution.
3. Return one of the solution with larger total weight from S_1 and S_2.

Theorem 1. *Algorithm 1 is $(1-\epsilon)/2$-approximation for any $1 > \epsilon > 0$.*

Proof. For the n_1 secondary users who request time intervals, the optimal solution can be found in polynomial time by dynamic programming. For the n_2 secondary user who request time duration, the optimal solution can be approximated within $1/(1+\epsilon')$ for any $1 > \epsilon' > 0$ by the PTAS for knapsack problem. Here, we also use S_1, S_2, and OPT to denote the total weight of these solutions. Then, $S_1 + \frac{1}{1+\epsilon'} S_2 > OPT$. According to pigeonhole principle,

$$\max(S_1, S_2) \ge \frac{1 - \frac{\epsilon'}{2-\epsilon'}}{2} OPT.$$

For each $1 > \epsilon > 0$, let $\epsilon = \frac{\epsilon'}{2-\epsilon'}$, we finish the proof. □

Moreover, here we show a general method for those problems in which users have a mixed time requirement, i.e., problems in the format $\mathrm{T} = \mathrm{M}$, when the corresponding problems in the format $\mathrm{T} = \mathrm{D}$ and $\mathrm{T} = \mathrm{I}$ are solved. The algorithm is as following.

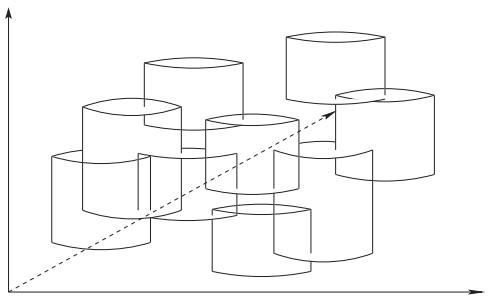


Fig. 1. An illustration of cylinder graph.

Theorem 2. *Algorithm 2 is at least $\frac{\alpha_1\alpha_2}{\alpha_1+\alpha_2}$-approximation.*

Algorithm 2. Approximation Method for Problems with Mixed Time Requirement

1. Find a channel allocation to all users who require for a time interval, i.e., $\mathrm{T} = \mathrm{I}$, using a method with an approximation ratio $\alpha_1 \le 1$;
2. Find a channel allocation to all users who require for a time duration, i.e., $\mathrm{T} = \mathrm{D}$, using a method with an approximation ratio $\alpha_2 \le 1$;
3. Return the better solution of these allocations as the solution of the problem.

Proof. In the optimal solution, suppose that $\beta (0 \le \beta \le 1)$ of the total profit comes from the users who require for a time duration; and $1 - \beta$ of the total profit comes from the users who require for a time interval. Algorithm 2 will give us a $\max(\beta\alpha_1, (1-\beta)\alpha_2)$ approximation by pigeonhole principle. The minimum value of $\max(\beta\alpha_1, (1-\beta)\alpha_2)$ is $\frac{\alpha_1\alpha_2}{\alpha_1+\alpha_2}$ when $\beta = \frac{\alpha_2}{\alpha_1+\alpha_2}$. Therefore, we finish the proof. □

5 PTAS for Problem YUI

In this section, we present a polynomial time approximation scheme for the problem YUI, where there is only a single channel available, the required region by every secondary user v_i is a unit disk, and each user v_i asks for a time interval $[s_i, e_i]$. The PTAS runs in $O(n^{\frac{1}{\epsilon^2}})$ time and provides an approximation factor of $(1-\epsilon)$ where n is the number of secondary users.

Notice that finding the set of bids with the maximum value is equivalent to solve the maximum weighted independent set in the following intersection graph of cylinders. Each bid

$$B_i = [b_i, \mathcal{F}_i, D(v_i, r_i), T_i]$$

defines a three-dimensional cylinder $\mathbf{B}_i = (D(v_i, r_i) \times [s_i, e_i])$ with weight b_i. See Fig. 1 for illustration. For simplicity, we assume that the three axes are X, Y, and Z and the axis Z denotes the time dimension. The disk $D(v_i, r_i)$ is called the base of the cylinder $\mathbf{B}_i$. The intersection graph has all the cylinders as its vertices, and two vertices form an edge if the corresponding two cylinders overlap. The weight of a vertex is the bid value of the corresponding secondary user.

Our PTAS is based on the shifting strategy developed in [10]. We will partition the space using hyperplanes perpendicular to X-axis and hyperplanes perpendicular to

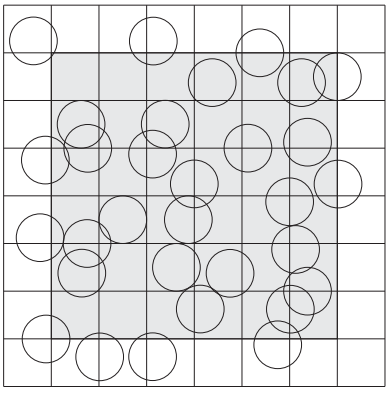


Fig. 2. An illustration of partition of space using hyperplanes ($k = 6$). This is a view from the top (Z-axis).

Y-axis. See Fig. 2 for illustration, which shows a slice produced by cut the space using the hyperplane $Z = 0$. In such a partition, we will throw away some cylinders that intersect some special hyperplanes $X \equiv a \bmod k$ and $Y \equiv b \bmod k$ for some special values of a and b and a given integer k, and then solve the subinstances of cylinders contained in each cell individually. Here, a cell is a three-dimensional cuboid defined by $\{(x, y, z) \mid a + ik \le x \le a + (i + 1)k, a + jk \le y \le a + (j + 1)k\}$ for some integers i and j. Let instance I be the set of all n cylinders. For any given integer $k > 1$, we derive k^2 polynomially solvable subinstances from the given instance I in polynomial time. The best value of those subinstance solutions is at least $(1 - \frac{2}{k})\omega(OPT(I))$, where $OPT(I)$ is the optimal solution on I and $\omega(OPT(I))$ is the value of the solution. Then, solve these subinstances using a dynamic programming procedure and return the solution with best value.

5.1 Deriving Subinstances

For simplicity, we assume that the diameter of all disks $D(v_i, r_i)$ is 1 and we also assume that the disk is open disk. Draw a grid consisting of hyperplanes $x = i$ (for $i \in Z$) perpendicular to X-axis and hyperplanes $Y = j$ (for $j \in Z$) perpendicular to Y-axis. The distance between every two parallel neighbor hyperplanes is the diameter of unit disk. So each cylinder will be hit by at most one hyperplane perpendicular to X-axis and at most one hyperplane line perpendicular to Y-axis.

For each i, j belong to $\{0, 1, \ldots, k - 1\}$, we compose a subinstance $I_{i,j}$ containing all cylinders except those being hit by a hyperplane from $\{x = p \mid p \equiv i \bmod k\}$ or hit by a hyperplane from $\{y = p \mid p \equiv j \bmod k\}$. There are k^2 different subinstances. For each subinstance, we calculate its optimal solution using the dynamic programming in every of the $k \times k$ grids, which will be described in detail in the next section. We first establish a technical lemma for the performance guarantee, then prove our algorithm achieve $(1 - \epsilon)$-approximation for any $1 > \epsilon > 0$.

Lemma 3. *For at least one subinstance $I_{i,j}, 0 \le i, j < k$,*

$$\omega(OPT(I_{i,j})) \ge \left(1 - \frac{2}{k} + \frac{1}{k^2}\right)\omega(OPT(I)).$$

Proof. Let us project all cylinders to the plane $Z = 0$ and we only focus on this plane $Z = 0$. If a disk is hit by a horizontal line i, it will be dropped k times in the subinstances $I_{i,0}, I_{i,2}, \ldots, I_{i,k-1}$. Same result also applies to the case that it is hit by a vertical line j. Here, we double count the number of dropping once when the disk is hit by a horizontal line and a vertical line at same time. Therefore, each disk is included in at least $k^2 - 2k + 1$ subinstances. Therefore, the disks in the optimal solution of I are also included in at least $k^2 - 2k + 1$ subinstances, which means

$$\sum_{i,j \in \{0,1,\ldots,k-1\}} \omega\Big(I_{i,j} \bigcap OPT(I)\Big) \ge (k^2 - 2k + 1)\omega(OPT(I)).$$

The best value of these subinstance solutions

$$\max_{0 \le i,j < k} \omega\Big(I_{i,j} \bigcap OPT(I)\Big) \ge \frac{1}{k^2} \sum_{0 \le i,j < k} \omega\Big(I_{i,j} \bigcap OPT(I)\Big) \ge \left(1 - \frac{2}{k} + \frac{1}{k^2}\right)\omega(OPT(I)).$$

The lemma then follows directly from the fact that $\omega(OPT(I_{i,j})) \ge \omega(I_{i,j} \bigcap OPT(I))$. □

Theorem 4. *Algorithm 3 is $(1 - \epsilon)$-approximation for any $1 > \epsilon > 0$.*

Algorithm 3. A PTAS for Problem YUI
Input: A given instance I and a given integer $k > 0$.
Output: A group of secondary users that can be satisfied without intersections.

1. Derive k^2 subinstances from the given instance I in polynomial time. The procedure of deriving will be shown in Section 5.1.
2. Solve each subinstances in polynomial time. The procedure of solving will be shown in Section 5.2.
3. Return the solution with the largest total weight in those solutions to sub-instances.

Proof. According to previous lemma, by setting $k = 2/\epsilon$, our algorithm implies that we have a PTAS (i.e., finding a solution whose total value is at least $1 - \epsilon$ times of the optimum) for problem **SUI** in time $n^{\frac{1}{\epsilon^2}}$. □

For each subinstance, each disk is in a $k \times k$ grid. The disks in different grid won't intersect each other. So the solution in each $k \times k$ grid is independent. In the following section, we show that the problem in a $k \times k$ grid is polynomially solvable.

5.2 Dynamic Programming

Here, we describe a dynamic programming approach to find a maximum weighted independent set for an instance $I_{i,j}$. We only consider cylinders contained in one $k \times k$ cell of $I_{i,j}$.

It is easy to show that the following simple greedy method does not work. A simple greedy method works as follows: it sorts the secondary users in the decreasing order of their bids, say $\{v_1, v_2, \ldots, v_n\}$ is the sorted list, and for $i = 1$ to n, a user v_i is granted the channel only if it will not cause any conflict with previously assigned users. Since the time requirement of that secondary user could intersect with a

group of secondary users whose total bids are arbitrarily larger, the approximation ratio could be arbitrarily bad.

For cylinders in each $k \times k$ cell, our dynamic programming method first sort them in nondecreasing order of their ending time e_i. For simplicity, let $\mathbf{B}_1, \mathbf{B}_2 \ldots, \mathbf{B}_i, \ldots, \mathbf{B}_n$ be the n cylinders contained in one cell in the sorted order. In the rest of section, we will use i to denote the cylinder $\mathbf{B}_i$.

Definition 1 (Pile). *A pile is an ordered collection $\langle j_1, j_2, \ldots, j_q \rangle$ of pairwise nonintersecting cylinders that intersect a common hyperplane $z = b$ (for some value b) perpendicular to Z-axis. Here, j_i is the* ending *time of a cylinder $\mathbf{B}_{j_i}$ and $j_t < j_{t+1}$.*

Given a hyperplane $z = b$ for some fixed value b, in a pile that was hit by $z = b$, there are at most $2k^2$ cylinders in the pile. Notice that all cylinders in this pile intersect the hyperplane $z = b$ and are disjoint from each other. Then, an area argument implies that the number of cylinders in a pile is at most $k^2/\frac{\pi}{4} < 2k^2$.

Lemma 5. *The total number of all possible piles in each $k \times k$ grid is polynomial of n.*

Proof. There are k hyperplanes perpendicular to X-axis and k hyperplanes perpendicular to Y-axis in a $k \times k$ grid. So there are k^2 intersections defined by these hyperplanes in each cell. Also there are k^2 center points in the k^2 small squares in each grid. Each unit disk in the $k \times k$ grid will be hit by either an intersection or a center point. The disks hit by the same intersection or center points must intersect with each other.

For each intersection or center point, we enumerate number of disks hit by the intersection or center point. Then, we can get $O(n^{2k^2})$ combinations which include all possible piles in the grid. Since there are $O(2k^2)$ cylinders in a pile, we can check whether a combination of at most $2k^2$ cylinders is a pile in time $O(k^4)$. □

Our dynamic programming approach will first sort the piles based on an order defined below; then, find the optimum solution of all cylinders that are ordered in front of a pile.

Definition 2. *Define an* order *on the set of piles as following: $\langle j_1, j_2, \ldots, j_l \rangle \prec \langle i_1, j_2, \ldots, i_m \rangle$ if 1) $j_l < i_m$, or 2) $j_l = i_m$ and $\langle j_1, j_2, \ldots, j_{l-1} \rangle \prec \langle i_1, j_2, \ldots, i_{m-1} \rangle$, or 3) $m = 0$.*

Definition 3. *Let $OPT(X \mid j_1, j_2, \ldots, j_l)$ be the maximum total weight of pairwise nonoverlapping cylinders from set X given that cylinders $j_1, j_2, \ldots, j_l$ already occupy their places. For a set of nonoverlapping cylinders $\{j_1, j_2, \ldots, j_l\}$ and a cylinder t with $t < j_1$, we define the marginal contribution, denoted as $R_{j_1, j_2, \ldots, j_l}(t)$, of cylinder t to $OPT(i : i \le t \mid j_1, j_2, \ldots, j_l)$ as*

$$(OPT(i \le t \mid j_1, j_2, \ldots, j_l) - OPT(i < t \mid j_1, j_2, \ldots, j_l))^+.$$

Here, $(x)^+ = \max\{0, x\}$.

Based on the above definition, a cylinder t has positive marginal contribution $R_{j_1, j_2, \ldots, j_l}(t)$ will clearly be used in an optimum solution $OPT(i \le t \mid j_1, j_2, \ldots, j_l)$. If its marginal contribution is 0, then it means that there is one optimum solution $OPT(i \le t \mid j_1, j_2, \ldots, j_l)$ that will not use the cylinder t.

As proved in [15], it is easy to show that

$$OPT(i \le t \mid j_1, j_2, \ldots, j_l) = \sum_{i=1}^{t} R_{j_1, j_2, \ldots, j_l}(i).$$

Here, $R_{j_1, j_2, \ldots, j_l}(t) = b_t + OPT(i < t \mid j_1, \ldots, j_l, t) - OPT(i < t \mid j_1, \ldots, j_l)$. We then present our dynamic programming to find an optimal solution in each $k \times k$ cell as following Algorithm 4.

Algorithm 4. Find Maximum Weighted Independent Set of Cylinders in Each Cell

Input: A set S of weighted cylinders contained in a cell.

Output: An optimum maximum weighted independent set in S.

1. Find all the piles $\mathcal{P}$ of cylinders from S and sort them in the order of $\prec$;
2. Take the piles one by one in the order starting from the least one. For each pile $\langle j_1, j_2, \ldots, j_l \rangle$, calculate and store in the memory the value $R_{j_2, \ldots, j_l}(j_1)$ according to the formula:
$$R_{j_2, \ldots, j_l}(j_1) = \left(b_{j_1} + \sum_{i : i < j_1} R_{j_1, j_2, \ldots, j_l}(i) - \sum_{i : i < j_1} R_{j_2, \ldots, j_l}(i) \right)^+$$
3. After all the piles corresponding values of $R_{j_2, j_3, \ldots, j_l}(j_1)$ is calculated for every possible pile $p = \langle j_1, j_2, \ldots, j_l \rangle \in \mathcal{P}$, we schedule the cylinders in the following way. Consider cylinders from S in the order of decreasing number. Take a cylinder j next in that order. Suppose that cylinders $\{j_1, j_2, \ldots, j_l\}$ are already scheduled, then schedule j iff $R_{j_1, j_2, \ldots, j_l}(j)$ is positive.

Theorem 6. *The running time of our dynamic programming is at most $O(n^{2k^2+1})$ in the worst case.*

Proof. From Lemma 5, we know that the total number of piles is at most n^{2k^2}. Thus, it takes at most $O(n^{2k^2} \lg n)$ to sort them. To find each cylinder's marginal contribution $R_p(j)$, it costs $n \cdot n^{2k^2}$. Thus, the total time is at most $O(n^{2k^2+1})$. □

Notice that $O(n^{2k^2} \lg n)$ is the worst-case time complexity in which we assume that, in the computation of the dynamic programming, there is a certain hyperplane such that there are $\Theta(n)$ cylinders, defined by the bids with bases contained in a $k \times k$ region, which intersect this hyperplane. The actual time complexity actually depends on the number of independent set of cylinders intersecting a hyperplane, which could be much smaller than n in practice. For example, if the arrivals of requests from users are randomly distributed, the network system could set a larger interval $[0, T]$ comparing with the users' time requirement. The number of independent set of cylinders intersecting a hyperplane would decrease significantly and total time complexity would be much lower than the theoretical bound. Our simulation studies verified our claim and showed that the algorithm actually runs efficiently.

6 Algorithm for Problem YUD

In this section, we design an approximation algorithm with a constant approximation ratio for problem YUD, where

there is only a single channel available, the required region by every secondary user v_i is a unit disk and each user v_i asked for a time duration d_i. We can use the same graph of cylinders which is mentioned in the previous section. See Fig. 1 for illustration. Notice that the cylinders can move along the axis Z in this problem since each user asks for a time duration.

The main idea here is to partition cylinders into g groups such that we can solve the maximum weighted independent set in each group and then take the group with the best solution. By pigeonhole principle, we know that it will give us $1/g$ approximation. We mainly will focus on designing a partition with minimum constant value g.

For simplicity, we assume that the radius of every disk $D(v_i, r_i)$ is 1 and we also assume that the disk is open disk. Draw a grid consisting of hyperplanes $x = \frac{2+\sqrt{2}}{3} i$ (for $i \in Z$) perpendicular to X-axis and hyperplanes $y = \frac{2+\sqrt{2}}{3} j$ (for $j \in Z$) perpendicular to Y-axis. The distance between every two parallel neighbor hyperplanes is $\frac{2+\sqrt{2}}{3}$ times the radius of unit disk.

Lemma 7. *All cylinders in instance are included in at least one of these groups.*

Proof. Consider the X-Y plane. All cylinders are unit disks on the plane. Since the radius of these disks is 1 and the distance between every two parallel neighbor hyperplanes is $\frac{2+\sqrt{2}}{3}$, a disk is hit by at most two parallel neighbor hyperplanes perpendicular to one axis. Therefore, all disks must be in a 3×3 cell. Group $G_{0,0}$, $G_{0,1}, \ldots, G_{2,2}$ represents all different groups with 3×3 cell on the plane. So all cylinders are included in at least one of these groups. □

Theorem 8. *Algorithm 5 is $1/9$-approximation and polynomial time solvable.*

Algorithm 5. Constant Approximation Method for SUI
Input: An instance for problem SUI.
Output: A group of secondary users who could be satisfied without intersections.

1. For each $i, j \in \{0, 1, 2\}$, find a group that contains all cylinders except those being hit by a hyperplane from $\{x = \frac{2+\sqrt{2}}{3} p \mid p \equiv i \bmod 3\}$ or hit by a hyperplane from $\{y = \frac{2+\sqrt{2}}{3} p \mid p \equiv j \bmod 3\}$. Denote the group as $G_{i,j}$.
2. Solve the subproblem for each group.
3. Return the solution with largest total weight in those solutions to all groups.

Proof. For each group, the disks in different cells cannot intersect with each other. So the solution in each cell is independent. We can solve these subproblems in each cell one by one and achieve the final result.

Considering the subproblem in each 3×3 cell, the disks must intersect with each other, which means that we can select only one disk at anytime. Therefore, the subproblem becomes a knapsack problem. We can solve it with a FPTAS. So each group can be solved with a FPTAS. We can then solve the nine groups in polynomial time and have a $1/9$ approximation by pigeonhole principle. □

7 Algorithm for Problem SUI

In this section, we design an approximation algorithm with approximation ratio $\Theta(\sqrt{m})$ for problem SUI where m is the total number of available channels. Here, we say an algorithm for SUI has approximation ratio α, if for all instances it will return a solution at least $1/\alpha$ of the optimum.

Notice that the set packing problem is a special case of the problem SUI due to the following observation. Recall that in a set packing problem, we are given a universal set $\mathcal{E}$ (with size m) and a set of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where each subset $S_i \subset \mathcal{E}$ is associated with a weight w_i. We need to find a subset of $\mathcal{S}$ such that the selected subsets are disjoint and the summation of their weights is maximized. Given a set packing problem, we can easily define a problem SUI as follows: $\mathcal{F} \leftarrow \mathcal{E}, \mathcal{F}_i \leftarrow S_i, b_i \leftarrow w_i$, and $D(v_i, r_i)$ centered at the point $(0, 0)$. These two problems will have the same optimum solution. Recall that set packing problem cannot be approximated within $m^{1/2-\varepsilon}$ for any $\varepsilon > 0$, unless NP = ZPP, i.e., some NP problem can be solved in probabilistic polynomial time. Thus, we have

Lemma 9. *Problem SUI cannot be approximated within $m^{1/2-\varepsilon}$ for any $\varepsilon > 0$, unless $NP = ZPP$.*

Using the same partition method in Section 5, we can get an asymptotical optimum approximation based on the approximation optimal solution in each group. Next, we will focus on designing an algorithm for each group with approximation ratio $\Theta(\sqrt{m})$. In the rest of the section, we assume that the disks $D(v_i, r_i)$ of *all* bidders conflict with each other, e.g., having a common point.

For set packing problem, a greedy algorithm was proposed in [16] that will find a solution whose total weight is at least $\frac{1}{\sqrt{m}}$ fraction of the optimum solution. Their method,[2] by interpreting using the terms in this paper, will first sort the bidders in decreasing order of

$$\frac{b_i}{\sqrt{|\mathcal{F}_i|}},$$

where $|\mathcal{F}_i|$ denotes the number of frequencies in $\mathcal{F}_i$. The method processes the bidders in this order and a bidder i is selected only if it will *not* cause conflict with any of the previously selected bidders.

In addition, since the set packing problem is essentially the maximum weighted independent set (MWIS) problem, we could also use heuristics for MWIS to find a good solution for set packing. In [23], Sakai et al. studied the performances of three simple greedy approaches: GWMIN, GWMAX, and min-weight-ratio. The heuristic GWMIN always recursively adds a vertex v minimizing $\frac{W(v)}{d_{G_i}(v)+1}$ among all vertices in graph G_i; and then remove v and all adjacent vertices from G_i to get graph G_{i+1}. It starts with $G_1 = G$ for the first step. The heuristic GWMAX always recursively adds a vertex v minimizing $\frac{W(v)}{d_{G_i}(v)(d_{G_i}(v)+1)}$ among all vertices in graph G_i; and remove v and all its adjacent vertices. The heuristic min-weight-ratio will always adds a vertex v minimizing

2. Notice that here the sorting is *not* based on naive criterion $\frac{b_i}{|\mathcal{F}_i|}$.

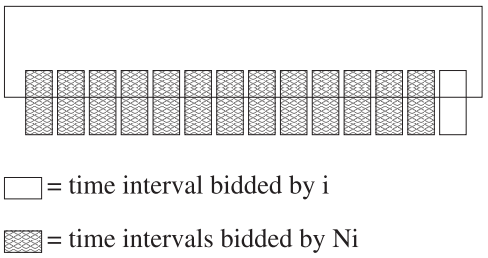


Fig. 3. This is a view from the lateral face (XY-axis).

$$\frac{\sum_{u \in N_{G_i}(v)} W(u)}{W(v)},$$

where $N_{G_i}(v)$ is the set of adjacent vertices of v in graph G_i. It was proved in [23] that heuristics GWMIN and GWMAX both have approximation ratio $\Theta(\frac{1}{\Delta})$ and the heuristic min-weight-ratio has approximation ratio

$$\sum_{v \in V} \frac{W(v)^2}{\sum_{u \in N_G(v)} W(u)}.$$

Here, Δ is the maximum node degree and $w(u)$ is the weight of the vertex u.

We begin our investigation of algorithm by first showing that a simple straightforward extension of the above algorithms could result in an arbitrarily large approximation ratio for our SUI problem.

First, let us consider the method in [16]. Sorting the bidders by descending

$$\frac{b_i}{\sqrt{|\mathcal{F}_i|} \times d_i}$$

is a very reasonable idea. But this approach may perform arbitrarily bad. Consider the following example. Assume that there are only two bidders, i and j: $b_i \gg b_j$ and d_i is so large that

$$\frac{b_i}{\sqrt{|\mathcal{F}_i|} \times d_i} < \frac{b_j}{\sqrt{|\mathcal{F}_j|} \times d_j},$$

if we choose bidder j but not bidder i, the total weight we get is b_j, that is arbitrarily smaller than the optimum solution b_i.

Second naive method is to sorting the bidders by nondecreasing order of $\frac{b(N_i)}{b_i}$. Here, we use $b(N_i)$ to denote the total weight bided by the bidders who intersect bidder i, and we call N_i the neighbors of bidder i. Unfortunately, a simple example shows that this approach may also perform quite poorly; see Fig. 3 for illustration. Assume that bidder i bids $\sqrt{n} + \varepsilon$ and each i's neighbor bids exactly 1. According to nondecreasing order of $\frac{b(N_i)}{b_i}$, we will choose bidder i instead of all its neighbors since $\frac{n}{\sqrt{n}+\varepsilon} < \frac{\sqrt{n}+\varepsilon}{1}$. The total weight we get is $\sqrt{n} + \varepsilon$, but the optimum solution is to select all nodes bidder 1, which give us n. This example shows that this naive method will have approximation ratio at least $\Omega(\sqrt{n})$, i.e., the solution returned is at most $O(\frac{1}{\sqrt{n}})$ of the optimum for some instances.

In the rest of section, we will design a polynomial time algorithm for problem SUI with approximation ratio $\Theta(\sqrt{m})$.

7.1 $\Theta(\sqrt{m})$-Approximation Algorithm for SUI with Single Time Interval

Consider each frequency set bided by different bidders. Assume that every bidder bids k frequencies and single time interval. When $k = 1$, obviously the optimum solution is the union of the optimum solutions OPT_i, where OPT_i is the optimum solution for the set of users who do bid for frequency $\mathbf{f}_i$. Notice that we can compute the optimal solution OPT_i using dynamic programming.

Next, we will focus on the scenario in which $k > 1$, i.e., each bidder bids for at least k different frequencies. In the following part, we will use OPT to denote the global optimal solution. For each frequency $\mathbf{f}_i$, let $OPT|_i$ be the set of users in OPT that bid for frequency $\mathbf{f}_i$. Obviously, we have

$$\sum_{i=1}^{m} OPT|_i \geq k \times OPT,$$

since each user in OPT will appear in at least k different $OPT|_i$. Thus,

$$\max\{OPT|_1, OPT|_2, \ldots, OPT|_m\} \geq \frac{k}{m} \times OPT.$$

We will show algorithms with approximation ratio $\Theta(\sqrt{m})$ for both groups, respectively. Then obviously, the maximum of these two solutions will give us $\Theta(\sqrt{m})$ approximation for SUI.

Lemma 10. *For SUI with group Z_1, there is a polynomial time method with approximation ratio $\Theta(\sqrt{m})$.*

Proof. First, for users in group Z_1, we just use $\max\{OPT_1, OPT_2, \ldots, OPT_m\}$ as our solution. Since each bidder bids at least $k \geq \sqrt{m}$ frequencies,

$$\max_{i=1}^{m} OPT(Z_1)|_i \geq \frac{\sqrt{m}}{m} OPT(Z_1) = \frac{1}{\sqrt{m}} OPT(Z_1).$$

Here, $OPT(Z_j)$ is the optimal solution for users in group Z_j for $j = 1, 2$. For bidders in Z_1, we use I_i to denote the set of bidders in Z_1 that bid for frequency $\mathbf{f}_i$. Then, we find the maximum weighted independent set OPT_i of I_i using standard dynamic programming. Obviously, $OPT_i \geq OPT(Z_1)|_i$. Thus, we have

$$\max_{i=1}^{m} OPT_i \geq \max_{i=1}^{m} OPT(Z_1)|_i \geq \frac{1}{\sqrt{m}} OPT(Z_1).$$

This finishes the proof. □

Lemma 11. *For users in group Z_2, there is a polynomial time method with approximation ratio $\Theta(\sqrt{m})$.*

Proof. We will convert this problem into the Scheduling Split Intervals Problem [2]. The ordinary SSIP considers the problem of scheduling jobs that are given as groups of nonintersecting segments on the real line. Each job J_i is associated with an interval, I_j, which consists of up to t segments, for some $t \geq 1$, and a positive weight, w_j. Two jobs are in conflict if any of their segments intersect. The objective is to schedule a subset of nonconflicting jobs with maximum total weight.

In [2], they gave an algorithm with $2t$-approximation ratio for problem SSIP. Here, we create a special instance

for SSIP problem as follows: Let $[0, T]$ be the original time period where bidders can place their time interval $[s_i, e_i]$, i.e., $0 \le s_i \le e_i \le T$ for every bidder i. We then create a bigger time period $[0, m \cdot T]$ where m is the total number of frequencies. Then, a user i will be associated with following $t_i = |\mathcal{F}_i| < \sqrt{m}$ segments in $[0, m \cdot T]$:

$$\{[(j-1) \cdot T + s_i, (j-1) \cdot T + e_i] \mid \mathbf{f}_j \in \mathcal{F}_i, 1 \le j \le m\}.$$

In other words, we duplicate the period $[0, T]m$ times for each of the frequencies in $\mathcal{F}$, and a user i will have a segment in the jth duplication if it bids for frequency $\mathbf{f}_j$. Then, there are at most $\sqrt{m}$ segments for every bidder. Then, based on algorithms in [2], we get $2\sqrt{m}$-approximation solution, i.e., find a solution with value at least $\frac{OPT(Z_2)}{2\sqrt{m}}$. □

Theorem 12. *Algorithm 6 has approximation ratio $\Theta(\sqrt{m})$ for problem SUI with single time interval.*

Algorithm 6. Asymptotically Optimum Method for SUI with Single Time Interval
Input: An instance for problem SUI
Output: A group of secondary users who could be satisfied without intersections.

1. Partition the bidders into two groups:
   a. Z_1 contains all the bidders that bid at least $\sqrt{m}$ frequencies; and
   b. Z_2 contains all the other bidders, i.e., bid less than $\sqrt{m}$ frequencies.
2. Approximate the optimal solution for Z_1 and Z_2 using best available polynomial time method.
3. Return the larger solution.

Proof. Let $OPT(Z_1)$ and $OPT(Z_2)$ be the optimum solution for groups Z_1 and Z_2, respectively. Then, the maximum of these two solutions is at least

$$\max\left(\frac{OPT(Z_1)}{\sqrt{m}}, \frac{OPT(Z_2)}{2\sqrt{m}}\right) \ge \frac{1}{3\sqrt{m}} OPT,$$

since either $OPT(Z_1) \ge \frac{1}{3} OPT$, or $OPT(Z_2) \ge \frac{2}{3} OPT$ from $OPT(Z_1) + OPT(Z_2) \ge OPT$. We finish the proof. □

Notice that by using a different group partition, where group Z_1 contains the bidders that bid for at least $\sqrt{\frac{m}{2}}$, we get an algorithm with approximation ration $\frac{\sqrt{2}}{4\sqrt{m}}$.

As a byproduct of our result, we show that for problem SSIP, there is no approximation algorithm with ratio $O(t^{1-\epsilon})$ for any $\epsilon > 0$ unless $NP = ZPP$. To the best of our knowledge, we are the first one to prove this.

Theorem 13. *For problem SSIP, there is no polynomial time approximation algorithm with ratio $O(t^{1-\epsilon})$ for any $\epsilon > 0$ unless $NP = ZPP$.*

Proof. We prove this by contradiction. Assume that there is a polynomial time algorithm $\mathcal{A}$ with ratio $O(t^{1-\epsilon})$ for some $\epsilon > 0$. We will use this algorithm to design an algorithm for solving the problem SUI with approximation ratio $O(m^{\frac{1}{2}-\epsilon'})$ for some $\epsilon' > 0$. We partition the bidders into two groups: group Z_1 contains all bidders requiring at least $m^{\frac{1}{2-\epsilon}}$ frequencies; group Z_1 contains all other bidders. We then solve the SUI problem as our previous discussion: finding a maximum independent set in Z_1 using $\max_{1 \le i \le m} OPT(I_i)$; finding a maximum independent set in Z_2 using a SSIP transformation. Obviously, we have

$$OPT(Z_1) = \max_{1 \le i \le m} OPT(I_i) \ge \frac{m^{\frac{1}{2-\epsilon}}}{m} \cdot OPT,$$

and for some constant $c > 0$,

$$OPT(Z_2) \ge c \cdot \frac{1}{m^{\frac{1}{2-\epsilon}(1-\epsilon)}} \cdot OPT.$$

It is easy to show that $\max\{OPT(Z_1), OPT(Z_2)\}$ will give us a solution that is at least $O(\frac{1}{m^{\frac{1}{2}-\frac{\epsilon}{4-2\epsilon}}})$ times of the optimum. Notice that here ϵ is a certain fixed constant. This implies that we have an algorithm for SUI (and thus set packing problem) with approximation ratio $m^{\frac{1}{2}-\epsilon'}$, where constant $\epsilon' = \frac{\epsilon}{4-2\epsilon} > \frac{\epsilon}{4}$. This is clearly impossible if NP is not equal to ZPP. This finishes the proof. □

7.2 $\Theta(t\sqrt{m})$-Approximation Algorithm for SUI with t Time Intervals

Different from the above case, assume that every secondary user could requires t time intervals (or at most t time intervals) and bids for a set of frequencies. We need to find a subset of nonconflicting users with the maximum total bid value. Obviously, when all users bid for more than $\lfloor |\mathcal{F}|/2 \rfloor$ frequencies, every pair of users will have a common requested frequency. Thus, for this case, computing the approximation solution for each frequency is exactly the traditional SSIP. And we can get $2t$-approximation solution for this special case.

Here, we show that Algorithm 6 also works for this case.

Lemma 14. *For users in group Z_1, there is a polynomial time algorithm for SUI with approximation ratio $\Theta(t\sqrt{m})$.*

Proof. For group Z_1, for each frequency $\mathbf{f}_i$, let $OPT(Z_1)|_i$ be the set of users in $OPT(Z_1)$ that bid for frequency $\mathbf{f}_i$ and let OPT_i be the optimum solution for users in Z_1 whose bid contains frequency $\mathbf{f}_i$. Clearly, $\omega(OPT_i) \ge \omega(OPT(Z_1)|_i)$. We just use $\max_{i=1}^{m} OPT_i$ as our solution. Assume that each user bids at least k frequencies. Then, we have $\sum_{i=1}^{m} OPT(Z_1)|_i \ge k \times OPT(Z_1)$, which implies that

$$\max_{i=1}^{m} OPT(Z_1)|_i \ge \frac{k}{m} OPT(Z_1).$$

Since each bidder bids at least $k \ge \sqrt{m}$ frequencies,

$$\max_{i=1}^{m} OPT_i \ge \frac{\sqrt{m}}{m} OPT(Z_1) = \frac{1}{\sqrt{m}} OPT(Z_1).$$

The SSID problem has a polynomial time algorithm with approximation ratio $\frac{1}{2t}$, which implies an algorithm with approximation ratio $\frac{1}{2t \cdot \sqrt{m}}$ for problem SUI with users in Z_1. □

Lemma 15. *For users in group Z_2, there is a polynomial time algorithm with approximation ratio $\Theta(t\sqrt{m})$.*

Proof. We can also convert this problem into SSIP [2]. Similarly, we duplicate the period $[0, T]$ m times for each of the frequencies in $\mathcal{F}$, and a user i will have t segments in the jth duplication if it bids for frequency $\mathbf{f}_j$. Then, there are at most $t \times \sqrt{m}$ segments for every bidder. Then, based on algorithms in [2], we get $2t \times \sqrt{m}$-approximation solution for $OPT(Z_2)$. □

Theorem 16. *Algorithm 6 is $\Theta(t\sqrt{m})$ approximation for problem SUI with t time intervals.*

Proof. Let $OPT(Z_1)$ and $OPT(Z_2)$ be the optimum solution for groups Z_1 and Z_2, respectively. Then, the maximum of the above two solutions is at least

$$\max\left(\frac{OPT(Z_1)}{t\sqrt{m}}, \frac{OPT(Z_2)}{2t\sqrt{m}}\right) \geq \frac{1}{3t\sqrt{m}} OPT,$$

since either $OPT(Z_1) \geq \frac{1}{3}OPT$, or $OPT(Z_2) \geq \frac{2}{3}OPT$ from $OPT(Z_1) + OPT(Z_2) \geq OPT$. □

Notice that by using a different group partition, where group Z_1 contains the bidders that bid for at least $\sqrt{\frac{m}{2}}$, we can also get an algorithm that finds a set of secondary users with profit that is at least $\frac{\sqrt{2}}{4t\sqrt{m}}$ fraction of the optimum.

8 Strategyproof Mechanism Design

From Sections 4 to 7, we mainly focus on the allocation methods, i.e., who will get the channels, for different versions of problems. Recall that we also need to compute payment schemes, i.e., how much each secondary user should pay. In this section, we show how to design a truthful mechanism based on the algorithms discussed in previous sections to compute payment schemes. The mechanism should be strategyproof, i.e., for each secondary user, reporting its valuation truthfully is its dominant strategy.

A strategyproof mechanism should satisfy both *IC* and *IR* properties which was introduced in Section 2.2. To satisfy these properties, the underlying allocation method must be a monotone output algorithm O. Then, we can design the payment based on a critical payment scheme P. It was proved that all strategyproof mechanisms should have a monotone allocation method and their payment schemes are based on the cut value [13]. Here, an allocation algorithm is monotone if, for an agent that is allocated, the agent will still participate in the output when it increases its bid. We define a critical value θ_i, i.e., the minimum valuation v_i that makes agent i participate in the output. A critical value payment scheme P^O for an algorithm O is that $p_i = \theta_i$ if agent i is in the output, otherwise, $p_i = 0$. If O is a monotone algorithm and P^O is a critical value payment scheme for $O, M = (O, P^O)$ is a strategyproof mechanism [13]. Now let us consider whether algorithms described in previous sections are monotone. If they are monotone, we also have relevant strategyproof mechanism.

In these algorithms, we used technology of dynamic programming, grouping and the classic FPTAS for knapsack problem. Dynamic programming and grouping clearly do not affect the monotone property. However, the classic FPTAS [5], [22] for knapsack problem is not monotone. We know that classic FPTAS for knapsack uses $\lfloor \frac{np_i}{\epsilon P} \rfloor$ to round all values down to a smaller value, where n is the number of items, p_i is the profit of item i, and P is the maximal possible profit. Then, solve the knapsack by using dynamic programming. For more details, see [5], [22]. Here is a counterexample why these methods are not always monotone. Suppose a knapsack with size 300, items 1, 2, and 3's profits and sizes are all 100, items 4, 5, 6, and 7's profits and sizes are all 75. We choose $\epsilon = \frac{7}{15}$. After rounding, items 1, 2, and 3's profits are both 15, items 4, 5, 6, and 7's profits are all 11. So the solution of knapsack is items 1, 2, and 3. However, if item 1 increases its profit to 101, after rounding, item 1's profit is 15, items 2 and 3's profits are both 14, items 4, 5, 6, and 7's profits are all still 11. Therefore, solution will be items 4, 5, 6, and 7. Item 1 will not participate in the solution when increasing its bid, which means that this FPTAS is not monotone.

Now our question becomes whether there is a monotone FPTAS for knapsack problem. If we have such FPTAS, all algorithms in previous sections can be monotone, because combinations of monotone methods is still monotone. In [4], Briest proposed an alternative rounding scheme that transforms a pseudopolynomial algorithm into a monotone FPTAS for knapsack problem. Using this FPTAS for knapsack problem, all algorithms presented in previous sections will be monotone. Therefore, we can design strategyproof mechanisms $M = (O, P^O)$ for all problems discussed in previous sections. Observe that the payment of any agent i in a strategyproof mechanism is always at most its bid $\mathbf{b}_i$. Due to space limit and its simplicity, we omit all mechanisms here.

Dynamic user join and leaving. So far we assumed that when we make spectrum allocation and decide the payment from secondary users, we know all the requests by all secondary users. This is also the assumption made by the majority results in the literature. In practice, it is possible that secondary users may arrive in an online fashion and we are required to make a decision on whether to grant the request of a secondary user immediately or within a certain time limit. To address such challenge, we propose a simple solution here. We divide the time into multiple time intervals (the length of a time interval depend on applications) and make allocation on each interval at the beginning of that interval. New secondary users may join the system anytime. However, they cannot lease channels until next allocation begins, i.e., they will be considered at the beginning of next time interval. Existing secondary users may leave the system at anytime since we already charged the existing secondary users. After a secondary user leaves, it releases the right to the channels allocated at its required region. If a secondary user leaves earlier than its reserved time, here we assume that no compensation from the system will be given. At the beginning of any time interval, we run our mechanisms for only newly arrived secondary users without using the channels at regions already occupied by existing users. When secondary users will not lie about their arrival time, we can show that our mechanisms are still truthful and efficient.

9 Simulations

We conduct extensive simulations to study the performances of our algorithms, mainly algorithms for problem YUI. We also study the performance of the mechanisms by

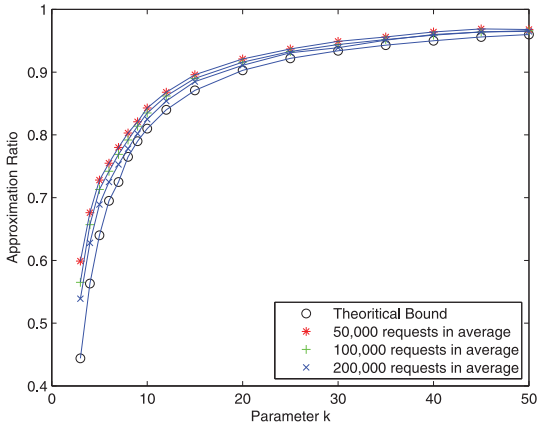


Fig. 4. The approximation ratio of algorithm for problem YUI.

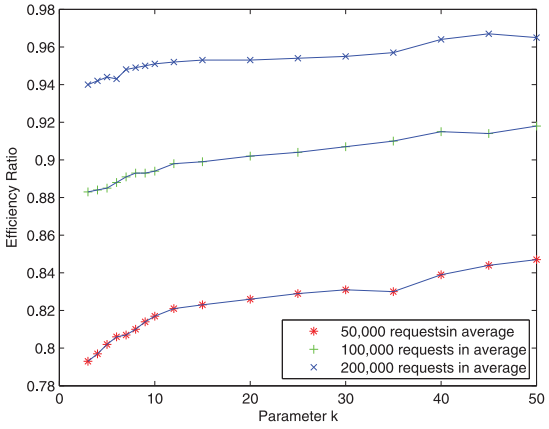


Fig. 5. The efficiency ratio of mechanism for problem YUI.

comparing the payment charged to all users and the total valuations of all allocated users.

In our simulations, we generate random requests with random bid, random time requirement, and random space requirement. The bid of each request is uniformly drawn from all integers in $[1, 100]$. The length of each time interval is uniformly drawn from all integers in $[1, 10]$ and the start time of each time interval is uniformly distributed in $[0, 100]$. The space requirement of each user is a unit disk. Each unit disk is uniformly distributed in a 100×100 square area.

One aspect may affect the results is the degree of competition. We use the average number of requests generated to represent the degree of competition. For mechanism design, more competition often means that the collected payment by our strategyproof mechanism is closer to the total valuations of all allocated users.

We first study the approximation ratio of our method for problem YUI in simulations. From Fig. 4, obviously, the approximation ratio increases when k increases, where k is the size of a cell used in our method. This is because when k increases, less cylinders defined by users' requests are thrown away during the step of dynamic programming. Therefore, the approximation ratio increases significantly. On the other hand, we found that the degree of competition does not affect approximation ratio much. Three curves which represent different degree of competition are very close to each other. It means the approximation ratio mainly depends on parameter k. We can observe that the approximation ratio is a litter bit worse when the competition is intense. This is because when there are more cylinders defined by users' requests, they tend to distribute in different area evenly.

Of course, the performances of our method in simulations are always better than the theoretical bound in performance analysis. We can see that even with a small parameter k, i.e., $k = 10$, our method can achieve more than 80 percent of the optimum. And the time complexity in simulations is not as bad as our theoretical analysis in Section 5.2. Most results can be computed within a few minutes ($<$10) when there are totally 200,000 requests.

Then, we study the efficiency ratio of the strategyproof mechanism that were described in Section 8. Here, the efficiency ratio is defined as the total payment charged from users who are granted the usage of channels over their total valuations. Clearly, for a strategyproof mechanism, its efficiency ratio is always at most 1. From Fig. 5, we can see that the efficiency ratio increases significantly when the competition is harder. This is because it is easier for our method to find a good replacement of users (without reducing the payment from users) when a user may reduce its bid when the competition is hard.

On the other hand, the parameter k also affect the efficiency ratio. The efficiency ratio increases slightly when parameter k increases. This is because less cylinders, defined by the bids of users, have been thrown away before the step of dynamic programming. This is a positive factor for finding a good replacement.

10 Conclusions

In this paper, we combine the game theory with communication modeling to solve some channel allocation problems. We study how to assign the spectrum and how to charge the secondary users such that the *overall social benefits* are maximized. More specifically, we formulate several versions of spectrum assignment problems by separately assuming whether the secondary users are single-minded or not, whether their required regions overlap or not, whether they ask for fixed time intervals or time durations.

When secondary users require only one channel, we show a simple $\frac{1}{2} - \epsilon$ approximation algorithm for problem YOM. For problem YUI, we present a polynomial time approximation scheme by deriving k^2 polynomially solvable subinstances from the given instance, where the best value of those subinstances' solutions is at least a $1 - \frac{2}{k} + \frac{1}{k^2}$ approximation of optimum. For problem YUD, we design a polynomial time algorithm with a constant approximation ratio. For problem YUM, we also give a polynomial time algorithm with a constant ratio by combining the results of problems YUD and YUI. On the other hand, for those problems such that some secondary users require a subset of channels and they are single-minded, we show an $\Theta(\sqrt{m})$-approximation algorithm for problem SUI. Furthermore, if at most t time intervals are required by a user, there is an approximation algorithm with ratio $\Theta(t\sqrt{m})$. We also show how to design strategyproof mechanisms based on those described algorithms that have a monotone property.

There are still a number of interesting problems to be studied. We leave it as a future work whether we can design

efficient approximation algorithms for problem SUD, where single-minded secondary users request unit disk region and a time duration. The second interesting problem is to study the noncooperative games among second users using some other solution concepts such as Nash equilibrium, or study the repeated games by these users. Another interesting game model is when the primary users are also noncooperative and a secondary user requirement can be satisfied by several primary users. Then how to design schemes to achieve market clearance between the primary users and secondary users. The last, but not the least, question is that we should design spectrum allocation methods for the case when the channels are available to secondary users for only some time intervals.

ACKNOWLEDGMENTS

The research of authors are partially supported by NSF CNS-0832120, NSF CNS-1035894, National Natural Science Foundation of China under Grant No. 60828003, the National High Technology Research and Development Program of China (863 Program) under grant No. 2007AA01Z180, and program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program).

REFERENCES

[1] I. Ahmad and J. Luo, "On Using Game Theory for Perceptually Tuned Rate Control Algorithm for Video Coding," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 3, no. 2, pp. 202-208, Feb. 2006.

[2] R. Bar-Yehuda, M.M. Halldórsson, J.S. Naor, H. Shachnai, and I. Shapira, "Scheduling Split Intervals," *Proc. 13th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '02),* pp. 732-741, 2002.

[3] Y. Bartal, R. Gonen, and N. Nisan, "Incentive Compatible Multi Unit Combinatorial Auctions," *Proc. Ninth Conf. Theoretical Aspects of Rationality and Knowledge (TARK '03),* pp. 72-87, 2003.

[4] P. Briest, P. Krysta, and B. Vöcking, "Approximation Techniques for Utilitarian Mechanism Design," *Proc. 37th Ann. ACM Symp. Theory of Computing (STOC '05),* pp. 39-48, 2005.

[5] C. Chekuri and S. Khanna, "A PTAS for the Multiple Knapsack Problem," *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '00),* pp. 213-222, 2000.

[6] E.H. Clarke, "Multipart Pricing of Public Goods," *Public Choice,* vol. 11, pp. 17-33, 1971.

[7] R.K. Dash and N.R. Jennings, and D.C. Parkes, "Computational-Mechanism Design: A Call to Arms," *IEEE Intelligent Systems,* vol. 18, no. 6, pp. 40-47, Nov. 2003.

[8] T. Groves, "Incentives in Teams," *Econometrica,* vol. 41, pp. 617-631, 1973.

[9] J. Hastad, "Clique is Hard to Approximate within $n^{1-\varepsilon}$," *Acta Mathematica,* vol. 182, pp. 105-142, 1999.

[10] D.S. Hochbaum and W. Maass, "Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI," *J. ACM,* vol. 32, pp. 130-136, 1985.

[11] K. Jansen and G. Zhang, "On Rectangle Packing: Maximizing Benefits," *Proc. 15th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '04),* pp. 204-213, 2004.

[12] J. Jia and Q. Zhang, and Q. Zhang, and M. Liu, "Revenue Generation for Truthful Spectrum Auction in Dynamic Spectrum Access," *Proc. 10th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing,* pp. 3-12, 2009.

[13] M.-Y. Kao, X.-Y. Li, and W. Wang, "Towards Truthful Mechanisms for Binary Demand Games: A General Framework," *Proc. ACM Conf. Electronic Commerce,* pp. 213-222, 2005.

[14] S.U. Khan and I. Almad, "Replicating Date Objects in Large-Scale Distributed Computing Systems Using Extended Vickrey Auction," *Int'l J. Computational Intelligence,* vol. 3, no. 1, pp. 14-22, 2006.

[15] S. Kovaleva, "Improved Dynamic Programming Subroutine in the PTAS for the Unit-Height Rectangle Packing Problem," *Proc. APPOL II Workshop,* 2002.

[16] D.J. Lehmann, L.I. O'Callaghan, and Y. Shoham, "Truth Revelation in Approximately Efficient Combinatorial Auctions," *Proc. ACM Conf. Electronic Commerce,* pp. 96-102, 1999.

[17] R.B. Myerson, "Mechanism Design by an Informed Principal," *Econometrica,* vol. 51, no. 6, pp. 1767-1797, Nov. 1983.

[18] R.B. Myerson, "Mechanism Design," *Allocation Information, and Markets,* J. Eatwell, M. Milgate, and P. Newman, eds., pp. 191-206, Norton, 1981.

[19] N. Nisan and A. Ronen, "Algorithmic Mechanism Design," *Proc. Ann. ACM Symp. Theory of Computing (STOC),* pp. 129-140, 1999.

[20] M.J. Osborne and A. Rubinstein, *A Course in Game Theory.* MIT Press, 2002.

[21] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos, "An Approximate Truthful Mechanism for Combinatorial Auctions with Single Parameter Agents," *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '03),* 2003.

[22] D. Pisinger and P. Toth, "Knapsack Problems," *Handbook of Combinatorial Optimization,* vol. 1, pp. 299-428, Springer, 1998.

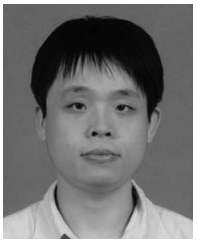
[23] S. Sakai, M. Togasaki, and K. Yamazaki, "A Note on Greedy Algorithms for the Maximum Weighted Independent Set Problem," *Discrete Applied Math.,* vol. 126, nos. 2-3, pp. 313-322, 2003.

[24] J.A. Stine, "Spectrum Management: The Killer Application of Ad Hoc and Mesh Networking," *Proc. IEEE Symp. New Frontiers in Dynamic Spectrum Access Networks (DySPAN),* Nov. 2005.

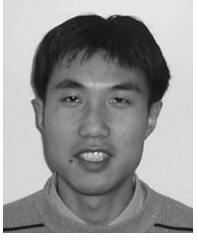
[25] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," *J. Finance,* vol. 16, pp. 8-37, 1961.

[26] P. Xu and X.-Y. Li, "Online Market Driven Spectrum Scheduling and Auction," *Proc. ACM Workshop Cognitive Radio Networks (CoRoNet),* 2009.

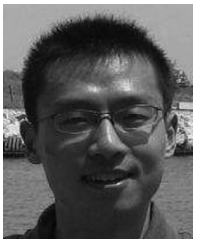
[27] P. Xu and X.-Y. Li, "SOFA: Strategyproof Online Frequency Allocation for Multihop Wireless Networks," *Proc. Int'l Symp. Algorithms and Computation (ISAAC),* 2009.



Ping Xu received the BS and MS degrees in electronic engineering from Shanghai Jiaotong University, China, in 1999 and 2003, respectively. He has been a PhD student of Computer Science Department at the Illinois Institute of Technology since 2006. His current research interests include algorithm design and analysis for wireless ad hoc networks, wireless sensor networks, online algorithms, and algorithmic game theory.



Xiang-Yang Li (M'99, SM'08) received the bachelor's degrees at the Department of Computer Science and at the Department of Business Management from Tsinghua University, China, both in 1995, the MS degree in 2000, and the PhD degree at the Department of Computer Science from University of Illinois at Urbana-Champaign. He has been an associate professor (since 2006) and an assistant professor (from 2000 to 2006) of computer science at the Illinois Institute of Technology. His research interests span wireless ad hoc and sensor networks, game theory, computational geometry, and cryptography and network security. He served as a cochair of the ACM FOWANC 2008 workshop, a cochair of the AAIM 2007 conference, a TPC cochair of WTASA 2007, and TPC members of the ACM MobiCom, the ACM MobiHoc, the IEEE INFOCOM, the IEEE ICDCS, and so on. He is an editor of the *Ad Hoc & Sensor Wireless Networks: An International Journal*, and the *Journal of Networks*. He has served as a guest editor of the *IEEE Journal on Selected Areas in Communications (JSAC)* and the *ACM Mobile Networks and Applications (MONET)*. He is a senior member of the IEEE.



ShaoJie Tang received the BS degree in radio engineering from Southeast University, China, in 2006. He has been a PhD student of Computer Science Department at the Illinois Institute of Technology since 2006. His current research interests include algorithm design and analysis for wireless ad hoc networks, wireless sensor networks, and online social networks.

JiZhong Zhao received the BS and MS degrees from the Department of Mathematics at Xi'an Jiaotong University, China, and the PhD degree in computer science, focus on Distributed System, from Xi'an Jiaotong University in 2001. He is a professor of Computer Science and Technology Department, Xi'an Jiaotong University, China. His research interests include computer software, pervasive computing, distributed systems, and network security. He is a member of the IEEE Computer Society and the ACM.