ABSTRACT

Wireless spectrum is a scarce resource, but in practice much of it is under-used by current owners. To enable better use of this spectrum, we propose an auction approach to dynamically allocate the spectrum in a secondary market. Unlike previous auction approaches, we seek to take advantage of the ability to share spectrum among some bidders while respecting the needs of others for exclusive use. Thus, unlike unlicensed spectrum (e.g., Wi-Fi), which can be shared by any device, and exclusive-use licensed spectrum, where sharing is precluded, we enable efficient allocaction by supporting sharing alongside quality-of-service protections. We present SATYA (Sanskrit for “truth”), a strategyproof and scalable spectrum auction algorithm whose primary contribution is in the allocation of a right to contend for spectrum to both sharers and exclusive-use bidders. Achieving strategyproofness in our setting requires appropriate handling of the externalities created by sharing. We demonstrate SATYA’s ability to handle heterogeneous agent types involving different transmit powers and spectrum needs through extensive simulations.

1. INTRODUCTION

Spectrum is a limited and expensive resource. For example, the 2006 Federal Communications Commission (FCC) auctions for 700 - 800 MHz raised almost $19 billion dollars. Hence, the barrier to entry for potential spectrum buyers is high. One can either buy a lease on spectrum covering a large area at a high price or use the limited spectral bands classified as unlicensed (e.g., Wi-Fi). Such unlicensed bands are subject to a “tragedy of the commons” where, since they are free to use, they are over-used and performance suffers [8]. Efforts such as the recent FCC ruling on white spaces are attempting to free additional spectrum by permitting opportunistic access [4]. However, such efforts are being met with opposition by incumbents (such as TV broadcasters and wireless microphones manufacturers) who have no incentive to permit their spectrum to be shared.

Motivated by these observations, many researchers and companies (e.g., [7, 36, 19]) have proposed allowing spectrum owners and spectrum users to participate in a secondary market for spectrum where users are allocated the use of spectrum in a small area on a dynamic basis based on their short- or medium-term needs. This approach is beneficial for two reasons. First, it allows flexible approaches to determining how best to allocate spectrum rather than relying on the decision making of regulators like the FCC in the United States. Second, it provides an incentive for spectrum that is currently owned but unused or under-used (such as the television spectrum) to be made available by its owners. Note, by secondary market we mean, one in which the owner of a chunk of spectrum leases different frequencies to other users who bid for the spectrum. The FCC has also recognized the potential use of a secondary spectrum market and has begun encouraging spectrum owners in certain bands to sublease the spectrum [18].

Prior work has proposed a number of auction designs to support such a market. However, the possibility of sharing in such markets have not been sufficiently explored. Most auctions provide exclusive access: the allocation is such that no winners interfere. However, this may not be the most efficient use of spectrum. For example, devices like wireless microphones are only used occasionally, so even if they require exclusive access while in use some other device may be able to use the same spectrum on a secondary basis when they are not. This heterogeneity of devices and demands is a source of opportunities for sharing. Further, many devices are capable of using a medium access controller (MAC) to share bandwidth when given the right to contend.

Rather than full sharing, as in the Wi-Fi model, using an auction has two key advantages. First, it provides revenue and thus incentives for primary spectrum owners to open up spectrum to other uses. Second, it provides incentives for different potential users to describe (through bids) their distinct needs for spectrum access, be it exclusive or with sharing. With Wi-Fi, if too many people try to use the same access point, service degrades and may become unacceptable for all of them, and no one has an incentive to consider the (negative) externality their use imposes on others.

Current proposals for secondary-market spectrum auctions are either unable to support the externalities created by two interfering devices sharing the same channel, or do not scale to realistic problem sizes and interference graph topologies. Solutions that ignore the possibility of sharing is rely on bid-
ders caring only about whether or not they are allocated a channel. With sharing, bidders also care about with whom they share the channel.

We present SATYA, a scalable, strategyproof auction algorithm that permits different classes of spectrum users (sharing and exclusive) to co-exist and share the spectrum, while appropriately accounting for the resulting externalities. Allowing bidders to report arbitrary externalities would yield an intractable allocation problem. But the externalities in our setting have significant structure, and SATYA uses a simple, yet expressive, language to allow bidders to express their value for different allocations given probabilistic activation patterns, interference, and different requirements for shared vs exclusive-access spectrum. Using this language, we can quantify the utility of a bidder for an allocation in terms of the fraction of the bidder’s demand that is satisfied. Bidders only interfere with other nearby bidders and, given a model for resolving contention by devices allocated shared spectrum and both simultaneous active, we can quantify the resulting externality.

Even without sharing, finding an optimal channel assignment involves solving a graph coloring problem and is NP-hard [20], and we are unable to find an optimal allocation fast enough to be reasonable for network deployment. Despite focusing on a single-parameter mechanism design problem by assuming that the components of a bidder’s type that reflect its interference, usage patterns, and penalty for being blocked from accessing allocated (shared) spectrum by an exclusive-use device on the network are known, a key difficulty remains: unlike in settings without externalities a straightforward greedy allocation approach that still allows sharing fails to be monotonic.

An allocation algorithm in our domain is monotonic if, submitting a larger bid for access to some fraction of a channel when active, always leads to an allocation in which the bidder receives a (weakly) larger fractional share of channel capacity, in expectation with respect to the activation patterns of other devices, and whatever the bids of other devices. Monotonicity is well known to be necessary and sufficient for strategyproofness in single-parameter domains [31]. Strategyproofness, a property that guarantees that it is optimal for each bidder to report his true value regardless of the actions of other bidders, is desirable for two main reasons. First, it provides strategic simplicity for bidders: they do not need to perform any sophisticated reasoning about the actions of others in order to determine how to participate. Second, it greatly simplifies evaluating a potential auction algorithm. If the algorithm is strategyproof, we can simply assume bidders report their true valuations. Otherwise, we would need to analyze the auction to determine the structure of equilibrium bids; naively assuming that bidders would report truthfully can overestimate both the efficiency of the allocation that results and the revenue that is raised.

In providing monotonicity, SATYA therefore employs a novel combination of bucketing bids into intervals wherein they are treated equally within an interval (this idea was employed in Ghosh and Mahdian [15]), and a computational ironing procedure that is used to validate the monotonicity of an allocation and perturb the outcome as necessary to ensure monotonicity (this idea was introduced by Parkes and Duong [32]). These techniques prevent cases where, if a bidder raises his bid, the greedy algorithm selects a different allocation that, at the time, is as good or better, but ends up being worse.

To evaluate SATYA we use real world data sources to determine participants in the auction, along with the sophisticated Longley-Rice propagation model [3] and high resolution terrain information to generate conflict graphs. We compare the performance of SATYA against other auction algorithms and baseline computations. Our results show that, when spectrum is scarce, allowing sharing using SATYA increases social welfare by 40% over previous approaches.

In summary, this paper makes the following contributions:

- The first strategyproof, scalable auction design for dynamic spectrum access that allows sharing and exclusive access by appropriately dealing with the externalities this creates.
- An approach that accommodates different classes of wireless users, each with a different transmit power, spectrum access, and activation patterns.
- The use of sophisticated propagation models and real world data to demonstrate in simulation the efficacy of SATYA.

1.1 Related Work

There has been significant work on spectrum auctions where a regulatory agency, such as the FCC in the United States, sells rights to spectrum across large areas (see, e.g. [10, 11]). However, we focus on secondary-market auctions, where the existing owner of spectrum wishes to resell it to a large number of smaller users subject to interference constraints.

Most secondary-market spectrum auction algorithms do not allow auction participants that interfere to share a channel [6, 13, 17, 34, 36]. Among these designs, VERITAS [36] was the first spectrum auction algorithm based on a monotone allocation rule, and thus strategyproof. However, VERITAS does not support sharing. Zhou et al. [35] proposed TRUST, which uses a double auction for cases when multiple owners are selling channels. In terms of supporting technologies, the use of a spectrum database in facilitating secondary market auctions has been proposed [19].

Turning to sharing, Jia et al. [24] envision spectrum owners auctioning off spectrum rights to a secondary user when it is not otherwise being used by the owner, and investigate how revenue can be maximized in this setting. While winners share with the spectrum owner in this way, there is no sharing among participants and no interference is tolerated.

Gandhi et al. [14] use an approach that allocates many small channels, and effectively enables sharing. However,
their algorithm allows sharing only among bidders who want only a portion of a channel. Thus, it cannot take advantage of bidders who are not always active (e.g., wireless microphones). Their approach is not strategyproof, and they perform no equilibrium analysis. Kasbekar and Sarkar [25] use a strategyproof auction and allow bidders to express arbitrary externalities, but this lack of structure makes their approach intractable except in a very simple case.

The issue of externalities in auctions has been considered more generally. Jehiel et al. [22, 23] consider situations, such as the sale of nuclear weapons, where bidders care not just about winning but about who else wins. However, the settings studied are without the computational challenges or need for expressiveness of our domain. A number of papers have considered externalities in online advertising [9, 15, 16, 27, 33]. However, this work (and similarly that of Krysta et al. [28] on the problem of externalities in general combinatorial auctions) is not directly relevant, as the externalities in spectrum auctions have a spectral structure, of which SATYA takes advantage.

2. AGENT MODEL

In order to find opportunities to share among heterogeneous agents (e.g., a user with a wireless device, or a TV station), we need a language to describe the requirements of each type of agent.

Our model uses discrete intervals of time (called epochs), with auctions clearing periodically and granting the right to agents to contend for access to particular channels over multiple epochs. The ultimate allocation of spectrum arises through random activation patterns of agents and interference effects, and depends on specifics of the medium-access control (MAC) contention protocol. The effect of this MAC protocol is also modeled within SATYA.

The interference between agents and their associated devices is modeled through a conflict graph, $G = (V, E)$, such that each agent $i$ is associated with a vertex $(i \in V)$ and an edge, $e = (i, j) \in E$, exists whenever agents $i, j$ would interfere with each other if they are both active in the same epoch and on the same channel.

Recall that we allow for exclusive-use and “willing to share” types of agents, where the former must receive access to a channel without contention from interfering devices whenever they are active, while the latter can still obtain value through contending for a fraction of the channel with other interfering devices. We say that a channel is free, from the perspective of agent $i$ in a particular epoch, if no exclusive-use agent $j$, who interferes with $i$ and is assigned the right to the same channel as $i$, is active in the epoch.

Formally, we denote the set of agent types $T$. Each type $t_i \in T$ is a tuple $T_i = (x_i, a_i, d_i, p_i, C_i, v_i)$, where

- $x_i \in \{0, 1\}$ denotes whether the agent requires exclusive use of a channel in order to make use of it ($x_i = 1$) or willing to share with another agent while both are active on the channel ($x_i = 0$).
- $a_i \in [0, 1]$ denotes the activation probability of the agent: the probability that the agent will want to use the channel, and be active, in an epoch.
- $d_i \in (0, 1]$ is the fractional demand of the channel that an agent who is willing to share access requires in order to achieve full value when active.
- $p_i \in \mathbb{R}^+$ denotes the per-epoch penalty incurred by the agent when it is active, but the assigned channel is not free. Both exclusive use and non exclusive use agents can have a penalty.
- $v_i \geq 0$ denotes the per-epoch value received by the agent in an epoch in which it is active, the channel is free, and in the case of non exclusive-use types, the agent receives at least a share $d_i$ of the available spectrum.

In this model, each agent demands a single channel. An extension to multiple channels is discussed in our technical report [26]. As the following examples show, this type space provides a rich language for agents to describe their intended usage pattern. In order to fully describe the auction we also require a model of what fraction of an agent’s potential value is realized when it shares a channel with other agents. We defer a full description of this model to the Appendix.

Examples

An agent who wishes to run a low-power (local) TV station on a channel would be unable to share it with others when active ($x_i = 1$), would be constantly broadcasting ($a_i = 1$), and would have a very large penalty $p_i$, since it is unacceptable for the broadcast to be interrupted by someone turning on another (exclusive use) device. Another agent might want to use a device like a wireless microphone that also cannot share a channel when active ($x_i = 1$), but might be used only occasionally ($a_i = 0.05$) and might have a smaller value of $p_i$ since it may be acceptable if it is occasionally unable to be used because there is another exclusive agent also trying to use the channel. For example, it might make sense to have several such devices share a channel if they interfere with each other sufficiently rarely.

There are also classes of agents capable of using a MAC and thus sharing a channel ($x_i = 0$). For example, someone who wants to run a wireless network could have constant traffic ($a_i = 1$) that consumes a large portion of the channel ($d_i = 0.9$), and might have a large penalty similar to a TV station because completely disconnecting users is unacceptable. However, such an agent is willing to share the channel with other non-exclusive types, and pay proportionately less for a smaller fraction of the bandwidth. There might also be opportunistic data users, for example a delay tolerant network [21], who occasionally ($a_i = 0.2$) would like to send a small amount information ($d_i = 0.4$) if the channel is avail-
Fig. 1: A potential violation of monotonicity. Nodes A and B are in contention range. At node A’s location channels 1 and 2 are free; at B only channel 1 is free.

3. DESIGN OF SATYA

Turning to the design of SATYA, we assume that the only component of an agent’s type that can be misreported is \( v_i \), with some bid \( b_i \neq v_i \) possible. This makes our auction an example of what Blum and Hartline [5] termed an attribute auction, where, in addition to the bid, the auctioneer knows some additional characteristics about each bidder. This is a reasonable assumption to make in practice. Most other characteristics, such as how often the agent makes use of the channel, how much of the channel he uses, whether his devices can use a MAC, and on what channels they can legally broadcast, can be observed by the auctioneer. Assuming he knows an agent’s penalty is a somewhat stronger assumption, but we expect that the auctioneer will have at least a broad idea of how well different applications tolerate preemption. In addition, we assume that the auctioneer knows the structure of the conflict graph.

Even if no agents are permitted to share channels, finding the efficient allocation is NP-Hard [20], as assigning bidders to channels such that no two neighbors have the same channel is a graph coloring problem. Therefore we adopt the same approach as previous strategyproof algorithms for channel allocation, and seek to assign agents to channels greedily. But in our setting, the effect of externalities and sharing is that a straightforward greedy algorithm will fail to satisfy the key property of monotonicity, which is necessary and sufficient in providing for strategyproofness.

3.1 Externalitys and Monotonicity

Let us first define the property of monotonicity in our setting. Given a joint bid vector \( b = (b_1, \ldots, b_n) \) received from agents (with \( b_j \geq 0 \) for all \( j \)) and a joint type vector \( t = (t_1, \ldots, t_n) \), an allocation algorithm \( A \) produces an allocation \( A(b, t) \). Each agent \( i \) has some utility \( U_i(A(b, t)) \) for this allocation.

Fixing the bids \( b_{-i} = (b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n) \) of other agents, an allocation algorithm is monotone if:

\[
U_i(A(b_i', b_{-i}, t)) \geq U_i(A(b_i, t)),
\]

for all bids \( b_i' \geq b_i \). This insists that the expected share of a channel available to an agent, and thus its expected utility, (weakly) increases as the agent’s bid increases.

With no sharing (and no externalities), a greedy allocation algorithm is monotone. However, Figure 1 shows how monotonicity can fail for simple greedy algorithms in the presence of sharing and externalities. The greedy algorithm considers each agent in (decreasing) order of bids and allocates him to the best available channel in terms of social welfare (or no channel if that is better). If there is a tie, it uses some tie-breaking rule, such as the lowest channel number. If agent A has a lower bid than agent B, the algorithm assigns agent B to channel 1, then agent A to channel 2, and both are fully satisfied. If agent A raises his bid so that it is higher than agent B’s bid, then the algorithm greedily assigns him to channel 1. Then, assuming sharing is better than not assigning B a channel, it has no other option than to assign agent B to channel 1, so the agent’s share the channel and are less well off.

3.2 The SATYA Algorithm

To prevent violations of monotonicity from happening while still using a greedy allocation rule, SATYA brings to bear a novel combination of (a) forbidding some shared allocations during the process (using a bucketing approach), and (b) canceling some shared allocations in a post-processing step (using an ironing approach). SATYA treats all agents in a bucket as if they bid the same amount, so changes of bid that do not change the bucket to which the agent is assigned have no effect on the allocation and thus do not violate monotonicity. Furthermore, agents in different bucket are allowed share only in a limited way, which prevents the greedy assignment from introducing externalities, and thus monotonicity violations, in such cases.

SATYA begins by assigning each agent \( i \) to a bucket \( K_i \) based on his bid \( b_i \). There many ways this can be done as long as it is monotone in the agent’s bid. For example, agent \( i \) with a bid \( b_i \) in the range \([2^k, 2^{k+1})\) could be assigned to bucket \( K_i = k \). In general, we assume that this is done according to some function \( \beta(k) \), such that bucket \( k \) contains all agents with bids \( b_i \) in the range \([\beta(k), \beta(k+1))\).

The agents are assigned channels greedily, in descending order of buckets, with the order of assignment for agents within the same bucket determined randomly. A channel \( c \) is considered available to allocate to agent \( i \) at some step in the algorithm, and given the intermediate allocation \( A \), if,

- the channel \( c \) is in \( C_i \);
- assigning \( i \) would not cause an externality to a neighbor from a higher bucket;
- and the combined demands of \( i \) and his neighbors from higher buckets assigned to \( c \) are less than 1.

Note that this does not preclude allocations where some agent does not have his demand fully satisfied. In simply requires that, in such cases, the agent is sharing with others in his own bucket.

For the next agent to be allocated, \( i \), SATYA finds the channel that will have the maximum marginal effect on the total value of all currently allocated agents and agent \( i \) itself. To do so, for every channel \( c \) that is available to the
agent, and including ⊥ and thus not allocating any spectrum to the agent. SATYA estimates the expected value to each agent \( j \) after assigning \( i \) to \( c \). This estimate differs from the agent’s actual bid by assuming that each agent in a given bucket shares the same value. This is important for achieving monotonicity as we need to ensure the way an agent is treated depends only on his bucket.

Finally, agent \( i \) is greedily assigned to the channel that maximizes the sum of the expected bid values of each agent (already allocated, and itself) without leaving any agent with a negative utility. The decision could be to allocate ⊥ and thus no spectrum to the agent. In the event of a tie, the agent is assigned to the lowest numbered among the tied channels (including preferring ⊥, all else equal).

After all agents in a bucket are assigned channels, there is an ironing step in which monotonicity of the allocation is verified, and the allocation perturbed if this fails. First, the allocation procedure is re-run for each agent, to determine what would happen had he been in a lower bucket. These counterfactuals are used to determine if the agent might have been able to be allocated a channel in a lower bucket. If so, this might cause a monotonicity violation where an agent bids more but ends up less well off, and so the provisional allocation is modified by changing the assignments of the neighbors with whom he shared a channel to ⊥. This is the ironing step: removing failure of monotonicity.

A full description of the allocation and pricing algorithms and a proof that this approach ensures monotonicity are available in the Appendix.

### 4. EVALUATION

In this section we compare the performance of SATYA to VERITAS. Since VERITAS does not permit sharing, we modify it slightly and implement VERITAS-S, which permits sharing as long as there are no externalities imposed (i.e. sharing is permitted only when the combined demands of sharing agents do not exceed the capacity of the channel). We also implement GREEDY, a version of SATYA without bucketing and ironing that provides higher overall efficiency. GREEDY is neither strategyproof nor monotone. Thus, agents’ bids need not match their true values. However, to set as high a bar as possible, we assume they do so. Since it gets to act on the same information but has fewer constraints than SATYA, GREEDY serves as an upper bound for our experiments.

#### Parameters:

As shown in Table 1, all our experiments use four classes of agents bidding for spectrum. Note that, in the table, we have normalized the values so the table reflects the range of \( a_i v_i \) rather than the range of \( v_i \). Each class represents different applications. For example, a TV station serving a local community is an agent who wants exclusive access for a long period of time. A wireless microphone is an example of an agent who wants exclusive access but for short periods of time. A low-cost rural ISP is an example of a Sharing-High agent who expects to actively use the spectrum but can potentially tolerate sharing, and a regular home user is an example of a Sharing-Low class agent whose spectrum access pattern varies. Note, each class of agents may have different transmit powers and coverage areas than the others. Since our goal is to evaluate the efficacy of SATYA in exploiting opportunities for sharing, we assign 5% of the total agents as exclusive-continuous, 15% exclusive-shared, 30% Sharing-High, and the remaining 50% Sharing-Low.

#### Methodology:

Each auction algorithm takes as input a conflict graph for the agents. To generate this conflict graph in a realistic manner, we implement and use the popular Longley-Rice [1] propagation model in conjunction with high resolution terrain information from NASA [2]. This sophisticated model estimates signal propagation between any two points on the earth’s surface factoring in terrain information, curvature of the earth, and climactic conditions. We use this model to predict the signal attenuation between agents, and consequently the conflict graph between the bidding agents.

We use the FCC’s publicly available CDBS [12] database to model the transmit power, location, and coverage area of Exclusive-Continuous users. Note, this information as well as the signal propagation predictions are sensitive to geographic areas. We model the presence of all other types of agents using population density information. Agents are scattered across a 25 mile x 25 mile urban area in a random fashion that factors in population density. Since each class of agent has a different coverage area, we determine that a pair of nodes conflicts if the propagation model predicts signal reception higher than a specified threshold.

We repeat each run of the experiment 10 times and present averaged numbers across runs. In these experiments, the number of channels is 5. In tuning SATYA, we experimented with a variety of methods for determining to which bucket to assign an agent. We omit these results for space reasons, but based on them use buckets of size 500 (\( \beta(k) = 500k \)).

In our experiments, we evaluate our approach using two metrics. The first, social welfare, is the sum of the valuations of winning agents (includes externalities). This measures how happy the participants are with the resulting allocation. Results for other measures of the efficiency are similar and are reported in the Appendix. The second is revenue, the sum of agents’ payments. This measures the incentive spectrum owners have to make their spectrum available to the auctioneer.

#### 4.1 Varying the Number of Agents

Figure 2 shows the performance of various algorithms as a function of the number of agents participating in the auc-
We consider social welfare the most important measure of performance: a market that finds success in the long run will allocate resources to those that find the most value. However, in our setting revenue may also be important to provide an incentive for current spectrum owners to participate in the secondary market. First, we measure the total revenue obtained as a function of the number of agents bidding for spectrum without reserve prices. We do not include GREEDY in this analysis because it is not strategyproof and it is not clear what agents will bid and thus what the actual revenue would be. As seen in Figure 3, the revenue obtained by SATYA and is much lower than VERITAS for smaller numbers of agents. We omit VERITAS-S from the figure for readability, but its performance also suffers. This is a consequence of sharing making it easier to accommodate agents.

To improve revenue, we institute reserve prices, minimum bids agents must make to participate in the auction. VERITAS explored a similar opportunity to increase revenue by limiting the number of channels sold. Using a reserve price of 400, we experiment to measure revenue by varying the number of bidders. As Figure 3 shows, this increases revenue for the auctioneer significantly for all algorithms. The increase is most pronounced with 50 agents (not shown because the improvement is so large) where revenue goes from essentially zero to approximately ten thousand. SATYA, which without a reserve price lost revenue by being too efficient in allocating agents, benefits slightly more than VERITAS. With a large number of agents, the reserve price is essentially irrelevant because of the amount of competition; with 550 agents the gain is below 12%. Hence, the main takeaway is that with appropriate reserve prices, the increase in efficiency enabled by SATYA does not have to come at the cost of revenue. Additional results about reserve prices are available in the Appendix.

5. CONCLUSIONS

We have presented SATYA, an auction approach to allocating wireless spectrum in a secondary market. Unlike previous auction approaches, SATYA allows sharing both among bidders as well as exclusive access, while still allowing for quality-of-service guarantees that Wi-Fi like approaches do not permit. From a technical perspective, introducing sharing also introduces interesting externalities. Using realistic simulations, we showed that the ability of SATYA to share channels results in superior allocations.
REFERENCES

[1] The ITS Irregular Terrain Model Algorithm, NTIA, Department of Commerce.


APPENDIX

A. ALLOCATION MODEL

In this section, we formally specify our model of an allocation and how agent utilities depend on the types of other agents with whom they share. Let allocation $A = (A_1, \ldots, A_n)$, denote the channel $A_i \in C_i \cup \{\perp\}$ is the channel allocated to each agent $i$, where $\perp$ indicates the agent has not been assigned a channel. To allocate a channel means that the agent has the right to contend for the channel when it is active, along with other agents that interfere with the agent (are neighbors on the conflict graph) and are allocated the same channel. Exclusive-use agents take priority over non-exclusive-use agents, and interfere with each other when multiple are simultaneously active. Non-exclusive-use agents share the channel when active at the same time and when the channel is otherwise free of exclusive-use agents.

Let $V_i(A, t)$ denote the expected value to agent $i$ for allocation $A$ given type profile $t = (t_1, \ldots, t_n)$. This also depends on conflict graph $G$ but we omit this term for notational simplicity. The efficient allocation of spectrum maximizes the expected total value across the agent population, that is

$$A^* \in \arg \max_A \sum_i V_i(A, t)$$

All allocations are feasible in our setting, since the expected value captures the negative externality due to interference. For this, we define $V_i(A, t)$

$$\begin{cases} 
0 & \text{if } A_i = \perp, \\
 v_i \cdot a_i \Pr_i(F|A, t) \frac{E_A[S_i|F, t]}{a_i} & \text{otherwise}
\end{cases} - p_i \cdot a_i (1 - \Pr_i(F|A, t)),
$$

(2)

where $\Pr_i(F|A, t) \in [0, 1]$ is the probability that the channel is free ($F$), with no exclusive-use agent interfering with the allocated channel, and $E_A[S_i|F, t] \in [0, 1]$ to denote the expected fraction of a channel that is available to agent $i$ to use given an epoch in which the channel is free and the agent is active (we will insist that $E_A[S_i|F, t] \leq d_i$). For an exclusive-use agent, this measure is always $E_A[S_i|F, t] = 1$ because it receives complete access to the channel when it is active and the channel is otherwise free.

An agent’s expected value depends on the entire allocation because of the effect of externalities with other, potentially interfering and sharing or exclusive-use agents. The first term takes the expected fraction of channel capacity (necessarily less than $d_i$) supplied in an epoch in which the agent is active and the channel is free from exclusive-use agents, and multiplies by the probability the channel is free and the agent is active $a_i \Pr_i(F|A, t)$, and the agent’s value for receiving $d_i$ fraction of the channel in an epoch. This expression assumes that an agent’s value is linear in the available bandwidth (up to max-demand $d_i$). The second term determines the expected per-epoch penalty due to the channel not being free when an agent is active (the probability of which is $a_i \cdot (1 - \Pr_i(F|A, t))$).

The probability that the channel allocated to agent $i$ is free, given allocation $A$ and type profile $t$, is

$$\Pr_i(F|A, t) = \prod_{j \in N_i \text{ s.t. } A_j = A_i \wedge x_j = 1} (1 - a_j),$$

(3)

where $N_i$ is the set of neighbors of $i$ in $G$. This is the joint probability that no exclusive-use neighbor in the conflict graph, allocated the same channel as $i$, is active in an epoch.

To compute $E_A[S_i|F, t]$, the expected fraction of a channel available to an agent in an epoch when it is active and the channel is free, we first consider the effect of a fixed number of active (non-exclusive-use) neighbors in such an epoch. Certainly, we insist that $E_A[S_i|F, t] \leq d_i$ because agent $i$ does not have more demand for the channel than this.

We assume that agents use a Carrier Sense Multiple Access (CSMA) style MAC that shares bandwidth as evenly as possible among the active (non-exclusive use) agents, sub-

ject to the constraint that no agent $i$ receives more than its demand $d_i$. Formally, if $N$ is the set of $i$ and $i$’s currently active neighbors with whom he shares a channel and $N_f = \{j \in N \mid d_j < f\}$, agent $i$ receives a share of the bandwidth equal to,

$$\text{share}_i(N, t) = \min \left( d_i, \max_{f \in [0,1]} \frac{1 - \sum_{j \in N_f} d_j}{|N| - |N_f|} \right)$$

(4)

The agent either gets his full demand or, failing that, his fair share (which the max in the equation determines) If all agents have the same demand $d_i$, this reduces to each either having his full demand satisfied if $d_i \leq 1/|N|$ or receiving a $1/|N|$ share of the channel otherwise. If some agents demand less than their fair share, the remainder is split evenly among the others.

Let $\nu_i(A, c)$ denote the set of neighbors of $i$ that, in allocation $A$, are allocated channel $c$, with $\nu_i(A)$ to denote the set of neighbors allocated the same channel as $i$. The probability that a particular set, $N \subseteq \nu_i(A)$ is active in any epoch is,

$$\text{active}_i(N, t) = \left( \prod_{j \in N} a_j \right) \left( \prod_{\ell \in \nu_i(A) - N} (1 - a_\ell) \right)$$

(5)

From this, an agent’s expected share of the channel, given that the agent is active and the channel is free (where the expectation is computed with respect to random activation patterns of interfering neighbors) is

$$E_A[S_i | F, t] = \begin{cases} 0 & \text{if } \Pr_f(F|A, t) = 0 \\ 1 & \text{if } x_i = 1 \\ \sum_{N \subseteq \nu_i(A)} \text{active}_i(N, t) \cdot \text{share}_i(N, t) & \text{o.w.} \end{cases}$$

(6)

The two special cases cover exclusive-use agents (who always get their full demand when active and when no other exclusive-use agent is active), and agents for whom the channel is never free (for whom we arbitrarily define it to be 0 as the value in this case turns out to be irrelevant).

Note that this requires computing a sum over all subsets of neighbors and is potentially quite expensive. In general, computing $E_A[S_i | F, t]$ requires time exponential in the number of neighbors $\nu_i(A)$ with which $i$ shares a channel. If sharing is limited to $r \ll n$ neighbors in practice, this can be done in time $O(2^n)$. For example, if $r$ is small due to the nature of the conflict graph or an imposed constraint on the amount of sharing permitted, then this can be computed. Indeed, in our experiments even with hundreds of agents participating in the auction we did not need to impose such a limitation.

### A.1 SATYA’s use of a MAC

As previously mentioned, we use a simple model to calculate what happens when agents share a channel. Our simple model can be replaced by a more sophisticated model from prior work that has explored the capacity of CSMA based wireless networks (e.g., [29, 38, 37, 30]) as long as, in expectation, having more neighbors decreases an agent’s share of the channel. This model can also be extended in other interesting ways. For example, we could add for each agent $i$ a parameter $\ell_i$, such that if he receives less than an $\ell_i$ fraction of the channel it is useless. This simply requires defining his share to be 0 if it would be less than $\ell_i$. Or we could adopt TDMA rather than CSMA.

For implementation, SATYA does not require drastic changes to existing MACs. The primary requirement is for a user to stop transmitting when it is another user’s turn (in the case of exclusive users). This is not unique to SATYA and is, for example, required of devices that use white spaces. However, a small change is required to a user’s network stack to seek to transmit only when he wins the auction (and therefore is allowed to contend for a channel). This can be implemented anywhere in the software stack.

## B. ALGORITHM DETAILS

### B.1 Allocation Algorithm

In this section, we give additional details about the SATYA allocation algorithm. We begin by formally defining several terms that were defined informally before. In the allocation algorithm, a channel is available if:

- the channel $c$ is in $C_i$;
- assigning $i$ would not cause an externality to a neighbor from a higher bucket: for all $j \in N_i$, with $K_j < K_i$,

$$\sum_{\ell \in \nu_i(A, c) \cup \{i\}} \text{share}_\ell(N, t) \leq 1$$

(7)

- and, the combined demands of $i$ and his neighbors from higher buckets assigned to $c$ are less than 1:

$$d_i + \sum_{j \in \nu_i(A, c), K_j > K_i} d_j \leq 1$$

(8)

When evaluating the an allocation $A$ with bids $b$ for agent $j$ who is in bucket $K_j$ we use the estimate

$$e_j(A, b) = \beta(K_j) \text{Pr}_j(F|A, b) \frac{E_A[S_i | F, b]}{d_j} - a_j \cdot p_j (1 - \text{Pr}_j(F|A, b))$$

(9)

This differ’s from $j$’s actual utility by assuming that its value is $\beta(K_j)$.

The complete allocation algorithm is specified in pseudocode as Algorithm 1. In the specification, we use distinct names to be able to refer to allocations created along the way. The variable $A_i(k, i, j)$ denotes the state of the allocation in bucket $k$ after considering the $j$th agent in the order given by the permutation $\pi$. Some of these allocations will be used for the counterfactual questions asked by ironing, so $i$ is the agent currently being omitted ($i = 0$ if there is no such agent).

**THEOREM 1.** Algorithm 1 is monotone.
Algorithm 1 SATYA Allocation Algorithm

$\pi \leftarrow$ a random permutation of $1 \ldots n$
$\mu \leftarrow \max_i K_i$
$\nu \leftarrow \min_i K_i$
$Allocation_i \leftarrow \perp \forall i$
$\mathcal{A}_i(M + 1, 0, n) \leftarrow \perp \forall i$

// Do Provisional Allocation
for $k = M$ to $m$ by $-1$
    for $j = 1$ to $n$
        $A(k, 0, 0) \leftarrow A(k + 1, 0, n)$
        for $c = 1$ to $\nu$
            if $K_c = k$
                $\mathcal{A}(k, 0, j) \leftarrow A(k, 0, j - 1)$
                $c \leftarrow \text{AssignChannel}(A(k, 0, j), \pi(j))$
                $\mathcal{A}_c(k, 0, j) \leftarrow c$
                $Allocation_{\pi(j)} \leftarrow c$
        end if
    end for
end for

// Counterfactuals to use for ironing
for $i = 1$ to $n$
    $A(K_i, i, 0) \leftarrow A(K_i + 1, 0, n)$
    for $j = 1$ to $n$
        if $K_c(j) = K_i$ or $\pi(j) \neq i$
            $c \leftarrow \text{AssignChannel}(A(K_i, i, j), \pi(j))$
            $\mathcal{A}_c(K_i, i, j) \leftarrow c$
        end if
    end for
end for

// Do ironing
for $i = 1$ to $n$
    $\text{free} \leftarrow \exists \text{avail. } c$ for $\pi(i)$ given $A(K_{\pi(i)}, \pi(i), n)$
    if $\mathcal{A}(\pi(i)) \neq \perp$ and $\text{free}$
        $\text{nbrs} \leftarrow \nu(\pi(i), Allocation)$
        while $d_{\pi(i)} + \sum_{j \in \text{nbrs}} d_j > 1$
            $j \leftarrow \text{last } j \in \text{nbrs}$ according to $\pi$
            $Allocation_j \leftarrow \perp$
            $\text{nbrs} \leftarrow \text{nbrs} \setminus \{j\}$
        end while
    end if
end for

return $\text{Allocation}$

AssignChannel($A$, $i$):
channels $\leftarrow \{ c \text{ available for } i \text{ given } A \}$
for all $c \in \text{channels} \cup \{ \perp \}$
    $\mathcal{A}_i \leftarrow c$
    $value_c = \sum_{j = 1}^n e_j(A, b)$
    if $\exists j \text{ s.t. } e_j(A, b) < 0$
        $value_c = 0$
    end if
end for

return $\arg \max_c value_c$
(break ties in favor of $\perp$, then lowest channel number)

Proof. First, we observe that an agent’s bid is only used to determine his bucket and is afterward ignored by the algorithm (estimates of utility use the agent’s bucket rather than his bid). Thus it is sufficient to consider deviations that cause $i$ to change buckets. If $A_i = \perp$, then $Pr_i(F|A) = E_A[S_i|F] = 0$, so the claim is trivially true. Otherwise, $i$ moves up to some bucket $k_2 > k_1$. Recall that $\nu_i(A, c) = \{ j \in N_i \mid A_j = c \}$ denotes the set of $i$’s neighbors assigned channel $c$ according to $A$. An important observation about the algorithm is that once it makes an assignment that some $A_i(k, 0, j) = c$, it never changes this for any later $k$ and $j$.

This is the reason the ironing step only changes $\text{Allocation}$ and not $A$. Thus, the set $\nu_i(A(k, 0, j), c)$ grows monotonically as the algorithms iterates over $k$ and $j$.

Since $i$ was assigned to $c$ in the assignment $A$, $c$ must have been available to him when he was assigned. By the third part of the definition of availability and the monotonic growth of $\nu$, $i$ would have his demand satisfied with neighbors $\nu_i(A(k, 0, j), c)$ for all $k \geq k_1 + 1$ and all $j$. In particular, this means his demand is satisfied with neighbors $\nu_0(A(k_2, \pi^{-1}(i) - 1), i, c)$.

When computing $\text{Allocation}^{'}$ with the new bids $b'$, the algorithm computes a new set of incremental allocations $A^{'}$. Since the algorithm does not look ahead, $A^{'}(k_2, 0, \pi^{-1}(i) - 1) = A(k_2, 0, \pi^{-1}(i) - 1)$. This means that, in $\text{AssignChannel}(A^{'}(k_2, 0, \pi^{-1}(i), j))$, $i$ could be assigned to $c$ and have his demand satisfied. Therefore he will be assigned to some such channel $c'$ (not necessarily $c$ as there might be a lower numbered channel available). Furthermore, on $c$ he does not impose any externality on his neighbors (all their demands are satisfied by the second part of the definition of availability). Therefore, since the algorithm greedily maximizes the total value, this true on $c'$ as well.

Again since the algorithm does not look ahead, $i$ increasing his bid does not change anything before bucket $k_2$, so $A^{'}(k_2 + 1, 0, n) = A(k_2 + 1, 0, n)$. Since the algorithm does not consider allocating $i$ a channel in bucket $k_2$ when computing $A$ (because he is in the lower bucket $k_1$) or when asking the counterfactual about what would have happened had $i$ not been in bucket $k_2$ in $A^{'}$, $A^{'}(k_2, i, n) = A(k_2, 0, n)$. Thus $\nu_i(A^{'}(k_2, 0, n), c) = \nu_i(A(k_2, 0, n), c)$ and in the ironing step running on $b'$, all of $i$’s neighbors with which it might have shared a channel with be reassigned to $\perp$ until its demand is satisfied. Since $i$’s neighbors were satisfied when $i$ was assigned to $c'$ and neighbors are ironed in the opposite order from that in which they were added, $i$ will not be ironed by any of its neighbors. Thus $Pr_i(F|A) = E_A[S_i|F] = 1$ and the allocation is monotone.

B.2 Pricing Algorithm

Given a monotonic allocation algorithm, then the expected payment to collect from each allocated agent are defined as is standard from Myerson [31].

In our case, these prices have a particularly simple form.
Because of the way ironing works, there is exactly one bucket in which an agent can receive an allocation in which he shares a channel with other agents. In any lower bucket he does not get allocated a channel; in any higher bucket he is guaranteed by ironing to have his demand fully satisfied. Thus there are only three possible allocations and three possible prices. Algorithm 2 shows how this bucket can be determined and what price should be charged in each case.

### Algorithm 2 Pricing Algorithm

\[ M \leftarrow \max_i K_i \]

\[ m \leftarrow \min_i K_i \]

for \( i = 1 \) to \( n \) do

if Allocation\( _i \) = ⊥ then

\[ P_i = 0 \]

else

run Algorithm 1 without agent \( i \) to get \( A'(k, 0, n) \forall k \)

\[ k = M \]

while \( k > m - 1 \) \& \( \exists c \in C_i \) do

s.t. \( c \) is available in \( A'(k, 0, n) \)

\[ k = k - 1 \]
end while

end if

**Recall that we assume that the only information about an agent’s type that can be misreported is its per-epoch value. Thus, by strategyproof we mean that agents have no incentive to lie about their value when bidding, whatever the bid values from other agents.**

**Theorem 2.** SATYA, which allocates channels using Algorithm 1 and charges payments according to Algorithm 2 is strategyproof.

Theorem 2 essentially follows from Myerson’s theorem. However, the theorem would have to be slightly adapted because in our model agents’ utilities depend on their penalty \( p_i \) in a way that makes them not quite fit the definition of a single-parameter domain. For this reason, we provide a direct proof of strategyproofness, which also gives insight into how payments work in our domain.

**Proof.** Previously, we have discussed agents in terms of their reported bids \( b_i \). In order to prove strategyproofness, we also need to take into account their true per-epoch expected values \( V_i \). Thus, so the true expected value for an allocation is computed by replacing \( b_i \) with \( V_i \) in the appropriate equation:

\[ u(A, i) = \begin{cases} 0 & \text{if } A_i = ⊥, \\ V_iPr_i(F|A)E_{A}[S_i|F] - p_i(1 - Pr_i(F|A)). \end{cases} \]

As observed there are only 3 possible allocations and sets of prices. An agent either gets nothing and pays nothing for a utility of 0, ends up in bucket \( k \) in which he might share and gets \( v_ia_iF - p_ia_i(1 - f) \) and pays \( β(k)fS - p_ia_i(1 - f) \) for a utility of \( (v_ia_i - β(k))fs \) or ends up in a higher bucket and gets \( v_ia_i \) (he has a channel to himself) and pays \( β(k + 1) - (β(k + 1) - β(k))fs \) for a utility of \( v_ia_i - β(k)(k + 1) - β(k))fs \).

First suppose that \( v_ia_i < β(k) \). If he ends up sharing his utility is \( (v_ia_i - β(k))fs < 0 \). If he ends up with a channel to himself his utility is

\[ v_ia_i - β(k + 1) + (β(k + 1) - β(k))fs < \]

Thus his optimal strategy is to bid his true value and share.

Now suppose that \( β(k) ≤ v_ia_i ≤ β(k + 1) \). If he bids truthfully, his utility is \( (v_ia_i - β(k))fs ≥ 0 \), so he cannot gain by lowering his bid. If he raises his bid above \( β(k + 1) \) he will end up with a utility of

\[ (v_ia_i - β(k + 1) + (β(k + 1) - β(k))fs = \]

Thus his optimal strategy is to bid his true value and share.

Finally, suppose that \( v_ia_i > β(k + 1) \). If he bids truthfully, his utility is \( v_ia_i - β(k + 1) + (β(k + 1) - β(k))fs > 0 \), so he does better than if he is not allocated. If he lowers his bid to be in bucket \( k \), his utility is

\[ (v_ia_i - β(k))fs ≤ v_ia_i - β(k + 1) + (β(k + 1) - β(k))fs \]

Thus his optimal strategy is to bid his true value.

**B.3 Running time**

Recall that \( n \) is the number of agents, and let \( χ = |C| \) denote the number of channels. The running time of SATYA is determined largely by the implementation of the AssignChannel procedure. As discussed in Section A, this require computation that scales exponentially in the number of neighbors with which \( i \) shares each channel considered. Thus, by in domains where this is limited to at most \( r \) neighbors then the call to AssignChannel requires time \( O(\chi n2^r) \). Indeed, we did not need to impose any limit on the number of neighbors in generating our simulation results, because agents’ utilities were such that it did not make sense for agents to share with a large number of other agents.

**Theorem 3.** SATYA’s running time is determined by the time needed for \( O(n^3) \) calls to AssignChannel, so the total running time is at most \( O(\chi n4^2) \).
Proof. SATYA needs to calculate $A(k, 0, n)$ for each non-empty bucket $k$ and $A(K_i, i, n)$ for each agent $i$. There are at most $n$ non-empty buckets and $n$ agents for a total of $2n$ allocations to be computed. Each allocation requires assigning a channel to each agent at most once, so there are $O(n^2)$ calls to AssignChannel. Ironing takes time $O(\chi n)$ per agent for a total of $O(\chi n^2)$, so the running time of the allocation is dominated by the calls to AssignChannel (which needs at least time $\chi$ to consider each channel).

The pricing algorithm runs for each agent and runs the allocation algorithm twice: once to determine in which bucket the agent might share and once to determine what his share would be in that bucket. Thus SATYA requires $2n + 1$ runs of the allocation algorithm for a total of $O(n^3)$ calls to AssignChannel.

B.4 Extensions

An earlier auction proposal, VERITAS [36], suggests a number of ways to extend auction algorithms to multiple channels. In particular, agents can either require a specific number of channels or be willing to accept a smaller number than they request. Agents may also wish to insist that an allocation of multiple channels be contiguous. SATYA can be extended to allow all of these. Due to space considerations we omit discussion of the algorithmic changes required, but we present simulation results in the appendix. Essentially, these changes require appropriately adapting the notion of when a group of channels is “available” to an agent.

SATYA has a number of parameters that can be altered in various ways. One obvious choice is the function $\beta$ used to assign agents to buckets. Any function that is monotone in an agent’s bid can be used. This includes functions that take into account other facts about the agent, for example his type or the number of neighbors he has in the conflict graph.

Another area of flexibility in defining SATYA is in the role of the permutation $\pi$. Rather than choosing it randomly, any method that does not depend on agent bids can be used. Some natural possibilities include ordering agents by their degree in the conflict graph (so that agents who interfere less are allocated first), ordering by a combination of activation probability and demand (so that agents who use less spectrum are allocated first) or considering exclusive agents last since they impose much larger externalities on those with whom they share. We leave further exploration of this for future work.

C. ADDITIONAL EVALUATION

In our additional experiments, we use the following metrics:

- **Allocated Agents**: The total number of agents that are allocated at least one channel by the auction algorithm.
- **Social Welfare**: The sum of the valuations for the allocation by winning agents (includes externalities).
- **Satisfaction**: The sum of the fraction of his demand each agent had satisfied.
- **Spectrum Utilization**: The sum of satisfaction weighted by activation probability and demand. From a networking perspective, spectrum utilization is a measure of how much the spectrum is being used (similar to the total network capacity).
- **Revenue**: The sum of agents’ payments.

Allocated agents, satisfaction, and spectrum utility are all alternatives to social welfare for measuring efficiency. As Figure 4 shows, the results for these other measures show the same trend we saw in Figure 2(a).

We also measure the effect of varying the number of channels auctioned on the overall outcome of the auction. The results shown in Figure 5 demonstrate the following trend: as the number of auctioned channels increases, the gap in performance among the algorithms reduces. This is similar to having fewer bidders participate in the auction; with more channels, there is a reduced need for sharing and all algorithms perform similarly. As Figure 5(a) shows, SATYA is still able to assign more bidders than other algorithms until about 20 auctioned channels. Similarly, in Figure 5(b), we see that SATYA outperforms VERITAS by 20-60% in social welfare up until about 10 channels.

We also vary the number of agents and the number of channels simultaneously and the results for SATYA are shown in Figure 5(c). We see that as the number of agents increases, SATYA takes advantage of the increased opportunity for sharing and allocate more agents.

Hence, the main takeaway is SATYA provides substantial benefits when the number of channels makes spectrum scarce.

Finally, the results from a simulation that varies the reserve prices are shown in Figure 6 for 300 bidding agents. Figure 6(a) shows that with a reserve price of 0 (i.e. no reserve price) VERITAS performs better than SATYA and VERITAS-S in terms of revenue. As the reserve price begins to increase, the revenue derived from all three auctions increases. However, at around a price around 700 (depending on the algorithm), there is an inflection point in the revenue. As seen in Figure 6(b), this is because significantly fewer agents are allocated by the auction and social welfare decreases (Figure 6(c)).
Figure 4: Effect of varying the number of agents in the auction (compared to VERITAS)

Figure 5: Effect of varying the number of channels auctioned

Figure 6: Effect of reserve prices with 300 agents