Efficient Feasibility Checking in Reverse Clock Auctions for Radio Spectrum

by

Neil Newman

BA.Sc. in Engineering Science, University of Toronto, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

January 2017

Abstract

We investigate the problem of building a feasibility checker to repack stations in the reverse auction part of the FCC's ongoing, multi-billion-dollar "incentive auction". In this work, we describe the design of a feasibility checker, SATFC, that has been adopted by the FCC for use in the incentive auction. We also construct a reverse auction simulator both in order to evaluate SATFC and also to gain insight into how the performance of the feasibility checker impacts the overall cost and efficiency of the auction. Through running simulations that differ only in the feasibility checker used, we show that the feasibility checker has a significant impact on auction cost and efficiency.

Preface

A significant fraction of Chapter 3 was published in the Thirtieth AAAI Conference on Artificial Intelligence held in 2016 as "Solving the Station Repacking Problem" [17]. This paper was authored by Alexandre Fréchette, myself, and my supervisor Kevin Leyton-Brown. I was involved in writing text, running experiments, analyzing data, and coding SATFC. The remainder of this thesis has not been published elsewhere. I wrote the code for the simulator described in Chapter 4, and performed all of the experiments and data analysis in Chapter 5.

Table of Contents

List of Tables

List of Figures

Glossary

FCC	Federal Communications Commission
SATFC	SAT-based Feasibility Checker
VHF	very high frequency
UHF	ultra high frequency
LVHF	lower VHF
HVHF	higher VHF
BIA	BIA Kelsey
NAB	National Association of Broadcasters
SAT	Propositional Satisfiability
UNSAT	unsatifiable
MIP	Mixed-Integer Programming
SMAC	Sequential Model-based Algorithm Configuration
ALO	at least one
AMO	at most one
NP	nondeterministic polynomial time
VCG	Vickrey-Clarke-Groves

CSP Constraint Satisfaction Problem

AC-3 Arc Consistency Algorithm #3

ECDF Empirical Cumulative Distribution Function

Acknowledgments

I would like to thank my supervisor Kevin Leyton-Brown for all of his support throughout my degree and for giving me the opportunity to work on such an exciting project. I would also like to thank Alexandre Fréchette, who helped me settle into graduate school by providing mentoring and guidance, and for laying the groundwork for this research direction.

I am indebted to Paul Milgrom for many helpful conversations on the topic of the incentive auction and for his efforts in transforming my research into software used in production. I also want to recognize Ilya Segal for providing clarifications of the incentive auction rules and for his and Paul's valuable inputs on the simulation experiments.

I gratefully acknowledge support from Auctionomics and the FCC.

I am grateful to Hu Fu for taking the time to be my second reader.

SATFC was enhanced by the contributions of several undergraduate students, of which I had the pleasure of working directly with Paul Cernek and Emily Chen. Thanks for all of your hard work!

Lastly, I want to thank everyone in our GTDT research group—Jason, Chris, Alice, Lars, James and Hedayat—for many stimulating conversations, both research related and not. The positive and friendly lab culture has been one of my favorite parts of graduate school.

For Mom, Dad, and Robert. Thanks for all your support.

Chapter 1

Introduction

Many devices, including cell phones, operate by sending and receiving electromagnetic signals. These signals can interfere with each other, so transmission is regulated to ensure that devices can effectively communicate. In the US, this regulation is enforced by the Federal Communications Commission (FCC), a government body that allocates licenses to broadcast over specific frequency bands in a geographic region. Since electromagnetic spectrum bands suitable to wireless transmission are scarce, the FCC has used *spectrum auctions* to allocate these licenses efficiently since 1994.

Presently, the FCC is running a novel spectrum auction known as the *incentive auction*. What makes the incentive auction unique compared to prior spectrum auctions is that rather than allocating previously unlicensed spectrum, this auction reallocates spectrum currently held by television broadcasters. These broadcasters hold licenses in either the very high frequency (VHF) band, spanning channels 2-13, or the ultra high frequency (UHF) band, spanning channels 14-51. The UHF frequencies, which are in the 600 megahertz (Mhz) range, are particularly well suited to wireless data transmission on mobile devices, as they can penetrate walls and travel long distances [28]. Given that over-the-air television has become more redundant with the rise of cable, satellite, and streaming services[1], and that in parallel demand for mobile data is increasing, the incentive auction is a unique

[1] A 2012 estimate says that only roughly 10% of viewers rely solely on over-the-air broadcasts for video programming [10].

opportunity to reallocate spectrum to higher-value use.

The reallocated spectrum is most useful to mobile carriers if the same channels can be acquired across the country. Therefore, the auction process removes stations from the upper channels of the UHF band, either moving each station into the lower channels or purchasing that station's broadcast rights when this cannot be achieved. Broadcast rights must be acquired for some stations because stations interfere with each other, so it will not be possible to pack all of the existing stations into the reduced channel set. Once the upper channels are cleared of stations, broadcast licenses in those channels can be auctioned off.

The incentive auction is thus composed of two related parts: a *reverse auction* in which the FCC will acquire spectrum from some broadcasters while remaining broadcasters will be "repacked" into a reduced set of channels, and a *forward auction* in which the FCC sells the freed up channels to mobile carriers. The choice of how many channels to reallocate, known as the *clearing target*, links these two parts. The incentive auction alternates between the two parts in a series of stages with progressively shrinking clearing targets until demand in the forward auction covers the cost of purchasing and reassigning stations in the reverse auction. While the forward auction process resembles prior spectrum auctions, the reverse auction was designed specifically for the incentive auction and is an unprecedented undertaking; this thesis will focus exclusively on the reverse auction.

The reverse auction mechanism decides which stations will be *winners* (have their broadcast rights purchased) and which stations to reassign to a new channel, and must do so in a way that does not violate interference constraints. There will be many feasible partitions—how should it choose between them? A sensible efficiency-related goal for the reverse auction might be, for any given clearing target, to maximize the total value of the stations that remain on-the-air, or equivalently, to minimize the total value of the winning stations. This suggests a straightforward maximization problem (e.g., a Vickrey-Clarke-Groves (VCG) mechanism), but in practice at the national scale this is computationally intractable.[2]

The FCC chose to use a descending-clock auction for the reverse auction. Roughly,

[2] An approximate VCG solution is not a promising direction either, as even small optimization errors can lead to dramatic pricing errors and destroy strategy-proofness, as laid out by Milgrom and Segal (2014).

it works as follows: stations are approached in a round-robin fashion and made a series of decreasing price offers for their spectrum. When a station declines an offer, it exits the auction and is guaranteed to be assigned a channel in its pre-auction band when the auction concludes. As more and more stations exit over time, it may no longer be possible to make this guarantee to certain stations: a *feasibility checker* is used to determine whether or not there is a way to assign channels to a station along with all of the exited stations that does not cause interference. When the feasibility checker detects that a station can no longer safely exit, that station's spectrum is acquired at its most recently accepted price offer.

This auction format does not escape computational intractability either, as the job of the feasibility checker—determining whether or not a set of stations can be packed into a set of channels—is a nondeterministic polynomial time (NP)-complete problem, meaning that it belongs to a class of problems not known to be tractable. Since the reverse auction must conclude in a timely manner, and since the feasibility checker will be queried tens of thousands of times throughout the auction, a tight cutoff—on the order of minutes—is imposed on each query. There is no guarantee that a proof of feasibility or infeasibility will be found in the alloted time. In the event of a timeout, the auction proceeds as if the result was infeasible.

The consequences of mislabeling an answer are not symmetric: it would be terrible to incorrectly conclude that a repacking problem is feasible (this could make the auction outcome infeasible), whereas wrongly claiming that a problem is infeasible prevents a price offer from being lowered, possibly decreasing the auction's efficiency and causing the government to overpay for a station, but does not pose a fundamental problem for the auction itself. Nevertheless, these timeouts should be minimized for the sake of both the efficiency and cost of the auction (the descending clock is geometric, so early individual unsolved problems can cost the FCC millions of dollars each!).

We built SAT-based Feasibility Checker (SATFC), the feasibility checker used in the reverse auction.[3] Our goal was to design an efficient feasibility checker that minimizes timeout results. Working with anonymized versions of repacking prob-

[3]The reverse auction uses SATFC 2.3.1, released April 13, 2016.

lems from FCC simulations[4][5], we combined a Propositional Satisfiability (SAT) encoding, constraint graph decomposition, algorithm configuration, algorithm portfolios, local and complete search, domain-specific heuristics, and a novel caching scheme to create an efficient feasibility checker. This thesis describes both SATFC and a reverse auction simulator that we use to evaluate SATFC's performance. We also use the simulator to explore ways in which the feasibility checker affects the efficiency and cost of the reverse auction by comparing simulations that differ only in their feasibility checkers.

The incentive auction is interesting to study both due to its novelty and its economic importance: the spectrum reallocation is expected to improve social welfare by putting spectrum to a higher-value use, and, according to the Congressional Budget Office, the resale of spectrum is forecast to net tens of billions of dollars for the US government [5]. As a result, the auction has been the subject of considerable recent study by the research community, mostly focused on elements of auction design [3, 9, 25, 29, 33, 35, 40]. From a theory perspective, the reverse auction is a *deferred-acceptance* auction [34], an auction that selects its allocation by repeatedly rejecting the least attractive bids. Deferred acceptance clock auctions have many desirable properties: They are *obviously strategy-proof* [30], since an agent can conclude that deviating from honest reporting can never lead to better payoffs regardless of how other agents behave (and, in this context, can come to this conclusion even without understanding the details of how price offers are decremented!). These auctions are also *weakly group strategy-proof*, meaning that there are no profitable group deviations from honest reporting that benefit each agent in the deviating group. However, the highly non-uniform interference constraints used in feasibility checking make it difficult to analyze the reverse auction

[4]In order to validate the auction design, the FCC ran extensive simulations of the auction, based on a wide variety of assumptions about station participation and bidding behavior. These simulations explored a very narrow set of answers to the questions of which stations would participate and how bidders would interact with the auction mechanism; they do not represent a statement either by us or by the FCC about how these questions are likely to be resolved in the real auction. It is of course impossible to guarantee that variations in the assumptions would not have yielded computationally different problems.

[5]Note that there is an interesting bootstrapping problem here, as the reverse auction simulator and feasibility checker co-evolve, and so the problems generated by the simulator change as the feasibility checker improves and handles denser packings.

solely with theory. Simulations have been used as a tool to forecast the cost of reallocating spectrum and the amount of spectrum that might be cleared [2]; to evaluate methods for setting price decrements [36]; to determine what must happen in any repacking solution based on the interference constraints (i.e., without appealing to the reverse auction process or prices) [26]; and to evaluate the impact of proposed rule changes, such as alternative scoring rules[6] [6]. Each of these works solves station repacking problems, although efficient feasibility checking is not their primary focus and none of them use SATFC, the feasibility checker optimized for this exact setting and deployed in the live incentive auction. While the station repacking problem has been studied in other contexts, falling under the umbrella of *frequency assignment problems*[7], we are not aware of other work with the goal of optimizing feasibility checking in the setting of the incentive auction.

The remainder of this document is as follows: Chapter 2 describes the reverse auction algorithm. Chapter 3 details the station repacking problem and our methodology for designing an efficient feasibility checker. Chapter 4 walks through the design of our reverse auction simulator. Chapter 5 shows experimental results of evaluating our feasibility checker on our reverse auction simulator. Finally, Chapter 6 provides directions for future research.

[6] Scoring rules are used to set station-specific prices in the reverse auction. They are covered in more detail in Section 2.1.2.

[7] See e.g., Aardal et al. (2007) for a survey and a discussion of applications to mobile telephony, radio and TV broadcasting, satellite communication, wireless LANs, and military operations.

Chapter 2

The Reverse Auction

The goal of this chapter is to present the reverse auction algorithm used in the incentive auction, which is the mechanism by which the FCC decides from which stations to purchase broadcast licenses and at what prices. Both UHF and VHF stations participate in the reverse auction: Even though VHF stations do not have broadcast rights in the more valuable UHF band, removing VHF stations creates space to move UHF stations into VHF. While utilizing the VHF band leads to more efficient solutions, it complicates the auction's rules significantly. We therefore begin with a simplified description which only considers UHF stations in Section 2.1, and then extend our description to handle VHF stations in Section 2.2. For a more detailed description of the auction process see Appendix D of the FCC's December 2014 public notice [12].

2.1 The Reverse Auction Limited to UHF Stations

We now proceed with a description of the reverse auction in a world with only UHF stations. First, stations respond to opening clock prices and decide to participate in the auction or exit permanently. Next, the feasibility checker identifies a feasible assignment for all non-participating stations. The auction then proceeds over a series of rounds, which consist of: (1) decrementing the clock and offering new prices, (2) collecting bids, and (3) processing bids. Only the processing step is not straightforward: Bids are considered sequentially. When processing a bid, the

feasibility checker first determines whether a station is still packable along with the exited stations. If it is packable, the station either accepts the new clock price or exits the auction. Otherwise, the station becomes *frozen*, meaning that it can no longer be packed in its original band. The station is now a *provisional winner*[1] and will be paid according to its most recently accepted offer. To complete this description we now show how prices are computed.

2.1.1 Computing Prices

The prices offered to each station differ based on station-specific characteristics. These characteristics are captured by a *volume*, which is computed by a *scoring rule*. The price P for a station s at round t is computed by multiplying its volume with a base clock price c_t. The base clock price is decremented each round by d_t, the maximum of 5% of its previous value or 1% of its initial value. More formally,

$$d_t = \max\{0.05 \cdot c_{t-1}, 0.01 \cdot c_0\} \tag{2.1}$$

$$c_t = c_{t-1} - d_t \tag{2.2}$$

$$P_{s,t} = c_t \cdot \text{Volume}(s) \tag{2.3}$$

where c_0 is the initial base clock price, used to compute the opening prices.

2.1.2 Scoring Rule

Scoring rules compute station volumes. The choice of scoring rule is important because varying the prices non-uniformly between stations is a way to influence the order in which stations exit and the prices they are paid, which impacts the auction's efficiency and cost. For example, it may be advantageous to encourage a station that causes much interference to remain in the auction, because by exiting it could freeze many other stations (and if it exits early, they would be frozen at very high prices!). Similarly, from a cost perspective, it might be possible to save money by offering lower prices to stations that service smaller populations, with the expectation that a station's value for its broadcast rights is proportional to the

[1] Provisional in the sense that the winning offer is conditional on the auction terminating in its current stage, which depends on demand in the forward auction.

size of the population it serves. The FCC uses a scoring rule that considers both of these issues, in which a station's score is proportional to both a heuristic estimating how difficult it makes other stations to repack and its population:

$$\text{Volume}(s) = A \cdot \sqrt{\text{Interference}(s)} \cdot \sqrt{\text{Population}(s)} \tag{2.4}$$

In this formula, $\text{Interference}(s)$ is computed by taking the sum over each other station s' of the maximum number of channels that s can prohibit s' from being assigned to, $\text{Population}(s)$ is the interference-free population that s reaches, and A is a scaling constant used to make the maximum volume one million (therefore capping the largest price offer at one million times the initial base clock price). Volumes for each station are computed only once before the auction begins—they do not vary throughout the auction.

2.2 Generalizing the Reverse Auction to Include VHF Stations

We now extend our description of the reverse auction to handle participation by VHF stations. VHF can be further divided into lower VHF (LVHF) (channels $\{2,\ldots,6\}$) and higher VHF (HVHF) (channels $\{7,\ldots,13\}$). In what follows, we refer to the band that a station is on prior to the reverse auction as the *home band* of that station. The *relinqushiment options*—that is, the ending states for stations in the auction—are therefore going off air (OFF), LVHF, HVHF, and UHF, ordered in terms of quality. When we say that one band is above or below another, we refer to this ordering. The reverse auction places restrictions on the options that stations can bid on. First, a station cannot bid on an option above its home band. Second, the reverse auction is a *ladder* auction, meaning that if a station ever bids to move to a band b, it will no longer be allowed to bid on options below b. The ladder restriction helps the auction set prices and prevents complex bidding strategies, making the auction simpler from a bidder's perspective. Simplicity is an important goal of the auction in order to encourage participation.

2.2.1 Computing Prices with Multiple Bands

The reverse auction can no longer simply calculate a single price for each station, but must instead calculate a price for each option available to each station. We now explain how these prices are computed.

An Ideal Case: Similar Stations

The easiest case to reason about is when there are three stations which serve identical populations: a UHF station, an HVHF station, and an LVHF station, with the property that none of these stations can co-exist on the same band, and that if any station moves to a lower band, its interference constraints in the new band are identical to the station it displaces. There are then four ways to clear the UHF channel, each of which create the same amount of value and hence should cost the same amount.

1. The UHF station can go to OFF for a price of $\$X$
2. The UHF station can go LVHF for a price of $\$Y$ (where $\$Y < \$X$) and the LVHF station can go to OFF at a price of $\$X - \$Y$
3. The UHF station can move to HVHF for a price of $\$Z$ (where $\$Z < \$Y$) and the HVHF station can go OFF at a price of $\$X - \$Z$
4. The UHF station can move to HVHF for a price of $\$Z$, the LVHF station can go to OFF for a price of $\$X - \$Y$, and the HVHF station can go to LVHF at a price of $\$Y - \$Z$.

Benchmark Prices

The reverse auction uses *benchmark prices* based on the above thought experiment: the auction first pretends that we are in the scenario of the above example where comparable stations exist and computes a benchmark price (which we will denote p), which it later transforms into an actual price as described below.

To compute the benchmark price for a station s in round t for the option of moving into b, a base clock decrement d_t is computed as before in Equation 2.1, except that now the previous round's price is decremented by a **fraction** of d_t. This

fraction is called a *reduction coefficient*, $r_{t,s,b}$. We will describe the computation of reduction coefficients in Section 2.2.1.

$$p_{t,s,b} = p_{t-1,s,b} - r_{t,s,b} \cdot d_t \tag{2.5}$$

The initial benchmark prices, denoted $p_{0,b}$, are chosen prior to the auction (with $p_{0,\text{UHF}} = 0$). The benchmark price is then converted to an actual price as follows.

$$P_{t,s,b} = \text{Volume}(s) \cdot \max\left\{0, \min\left\{p_{t,s,\text{OFF}}, p_{t,s,b} - p_{t,s,\text{Home-Band}(s)}\right\}\right\} \tag{2.6}$$

Let us break down this formula: The prices are weighted by volume as before. The max ensures the price is non-negative and the min upper bounds the price by the price offered to a UHF station to go to OFF. Lastly, the second term in the min reflects the pricing division described in Section 2.2.1, where the total payments for each of the four possibilities sum to the same amount (e.g., a UHF station moving to HVHF would get paid $p_{t,s,\text{HVHF}}$, an HVHF station moving to LVHF would get paid $p_{t,s,\text{LVHF}} - p_{t,s,\text{HVHF}}$, and an LVHF station moving to OFF would get paid $p_{t,s,\text{OFF}} - p_{t,s,\text{LVHF}}$, all of which sum together to a total payment of $p_{t,s,\text{OFF}}$).

Vacancy

Before describing the calculation of reduction coefficients, we need to describe one more concept, known as *vacancy*. Vacancy, denoted $V_{t,s,b}$, is a heuristic that estimates the competition that a station s faces for a spot in band b in round t, with higher values indicating less competition. More formally, vacancy is a volume weighted average of a function f over potential competitors for the vacant space in the band: Let $G(t,s,b)$ denote the set containing both s and stations which have the possibility to interfere with s in b which are currently bidding on options below b. The function f is computed by taking the number of channels in b to which s can be feasibly assigned (given the current assignment of the exited stations) and normalizing by the number of channels in b. If s cannot be assigned to any channels, 0.5 is used in place of the numerator.

$$f(t,s,b) = \frac{\#\text{ of feasible channels for } s \text{ in } b \text{ in round } t}{\#\text{ of channels in } b} \tag{2.7}$$

$$V_{t,s,b} = \frac{\sum_{s' \in G(t,s,b)} \text{Volume}(s') \cdot f(t,s',b)}{\sum_{s' \in G(t,s,b)} \text{Volume}(s')} \tag{2.8}$$

Reduction Coefficients

We now provide equations for computing the reduction coefficients. The full decrement is always applied to going off-air, that is $r_{t,s,\text{OFF}} = 1$.

The reduction coefficient for moving to HVHF is:

$$r_{t,s,\text{HVHF}} = \frac{p_{0,\text{HVHF}} \cdot \sqrt{V_{t,s,\text{UHF}}}}{(p_{0,\text{OFF}} - p_{0,\text{HVHF}}) \cdot \sqrt{V_{t,s,\text{HVHF}}} + p_{0,\text{HVHF}} \cdot \sqrt{V_{t,s,\text{UHF}}}} \tag{2.9}$$

Finally, the reduction coefficient for moving to LVHF is:

$$r_{t,s,\text{LVHF}} = \left(\frac{(p_{0,\text{LVHF}} - p_{0,\text{HVHF}}) \cdot \sqrt{V_{t,s,\text{HVHF}}}}{(p_{0,\text{OFF}} - p_{0,\text{LVHF}}) \cdot \sqrt{V_{t,s,\text{LVHF}}} + (p_{0,\text{LVHF}} - p_{0,\text{HVHF}}) \cdot \sqrt{V_{t,s,\text{HVHF}}}} \right) \cdot (1 - r_{t,s,\text{HVHF}}) + r_{t,s,\text{HVHF}}$$

2.2.2 Placing Bids

A station is presented with prices for each permissible option as described above, as well as the option to exit the auction and receive no compensation. It then submits its preferred option. If a station bids to move to a VHF band that it is not already in, the station must also submit a *fallback bid* to be used in case the feasibility checker is unable to pack the station into the preferred band. The fallback bid must be either to retain the currently held option or to exit the auction.

2.2.3 Bid Processing

In this section we describe how the algorithm processes the stations' bids. Bids are all placed in a queue, sorted in order of the ratio between the price reduction for the station's currently held option in this round and the station's volume, with ties

broken randomly. The auction system then begins a loop in which it finds the first station in the queue that is feasible in its home band, processes that station's bid, and removes it from the queue. This phase completes when the queue is empty or every station in the queue is frozen. When a station's bid is processed, if the bid requires a fallback option, the feasibility of the move is first tested: if the move is feasible, the station is moved, otherwise the station's fallback bid is processed. If a station moves bands, the tentative assignment is updated accordingly.

2.2.4 Provisional Winners

A station becomes a provisional winner once the auction determines that the station will be frozen for the remainder of the stage. A UHF station becomes a provisional winner as soon as it is frozen: being frozen for this UHF station means that it cannot be repacked along with the current set of exited UHF stations, and since exiting is a permanent move, this set will not lose any members throughout the auction. Therefore, if a UHF station becomes unpackable in UHF at any point during the auction, it will remain unpackable in UHF for the duration of the auction. However, this is not true for VHF stations: e.g., if an LVHF station bids for OFF, and a UHF station moves into LVHF, the LVHF station may become frozen. However, the UHF station may later move out of LVHF, which might unfreeze the LVHF station. Therefore, at the end of the bid processing step, the reverse auction checks each frozen station to see if it has become a provisional winner. For a VHF station s, this means using the feasibility checker to determine that it is not possible to pack s into its home band alongside all of the exited stations and provisional winners in s's home band.

2.2.5 Removing Stations That Can Never Become Winners

It may be possible to determine over the course of the reverse auction that a station will always remain repackable in its home band regardless of the actions taken by other stations. Such a station will never become a winner: the reverse auction can continue to lower the compensation offered to that station all the way down to zero, and at some point the station will be better off exiting. Therefore, when the reverse auction identifies such a station, it forces that station to exit in order to speed up the inevitable.

In order to determine whether a station s with home band b can never become a winner, the reverse auction first identifies the set of stations X that might finish the auction in b: these are stations that are either already in b or else are in a band below b, are not provisionally winning, and have home bands weakly greater than b. Once the set X has been identified, in order to prove that s can never become a winner, the reverse auction must show that no matter which stations in $Y \subseteq X$ wind up in b at the end of the auction, it will be feasible to repack $\{s\} \cup Y$ in b. This may become easier to show as the auction progresses, since X can shrink as stations that might have finished the auction in b become provisional winners on bands other than b or move to bands above b. The reverse auction relies on a sound but incomplete heuristic to identify these stations. The heuristic works by computing once at the start of the auction the maximum number of channels for each pair of stations s, s' that s' can prevent s from being assigned to in s's home band. If this sum taken over all $s' \in X$ is smaller than the number of channels on which s can be placed in its home band, then s can never become a winner.

2.2.6 Termination

A stage of the reverse auction terminates when a round concludes and every station has either exited the auction or is a provisional winner.

Chapter 3

Designing an Efficient Feasibility Checker

In this chapter, we provide a detailed description of feasibility checking and explain the design of our feasibility checker SATFC. We begin with a more formal description of the station repacking problem. Then we explain ways in which the problem can be encoded into SAT and Mixed-Integer Programming (MIP) form. We then turn to some domain-specific heuristics we use to solve problems, as well as ways in which problems can be simplified into smaller problems. We then discuss how algorithm configuration can be used to combine all of these ideas. Lastly, we describe a caching scheme for station repacking problems and explain how all of the aforementioned ideas combine to form our solver, SATFC. In what follows, we will sometimes say that a problem is SAT or unsatifiable (UNSAT) meaning it is feasible or infeasible respectively.

3.1 The Station Repacking Problem

Each television station in the US and Canada $s \in \mathcal{S}$ is currently assigned to a channel $c_s \in \mathcal{C} \subseteq \mathbb{N}$ that ensures that it will not excessively interfere with other, nearby stations. The FCC reasons about what interference would be harmful via a complex, grid-based physical simulation ("OET-69" [11]), but has also processed the results of this simulation to obtain a Constraint Satisfaction Problem (CSP) style

formulation listing forbidden pairs of stations and channels, which it has publicly released [14]. These constraints prohibit channel reassignments in which one station would reduce the interference-free population served by another station by more than 0.5%, so that after the incentive auction each station that remains on-the-air will serve roughly the same viewers that it served prior to the auction [13]. Let $\mathcal{I} \subseteq (\mathcal{S} \times \mathcal{C})^2$ denote a set of *forbidden station–channel pairs* $\{(s,c),(s',c')\}$, each representing the proposition that stations s and s' may not concurrently be assigned to channels c and c', respectively. The effect of the auction will be to remove some broadcasters from the airwaves completely, and to reassign channels to the remaining stations from a reduced set of channels. This reduced set is defined by a *clearing target*: some channel $\overline{c} \in \mathcal{C}$ such that all stations are only eligible to be assigned channels from $\overline{\mathcal{C}} = \{c \in \mathcal{C} \mid c < \overline{c}\}$. The clearing target is fixed for each stage of the reverse auction. Each station can only be assigned a channel on a subset of $\overline{\mathcal{C}}$, given by a *domain* function $\mathcal{D} : \mathcal{S} \to 2^{\overline{\mathcal{C}}}$ that maps from stations to these reduced sets. The *station repacking problem* is then the task of finding a repacking $\gamma : \mathcal{S} \to \overline{\mathcal{C}}$ that assigns each station a channel from its domain that satisfies the interference constraints, i.e., for which $\gamma(s) \in \mathcal{D}(s)$ for all $s \in S$, and $\gamma(s) = c \Rightarrow \gamma(s') \neq c'$ for all $\{(s,c),(s',c')\} \in \mathcal{I}$. A problem instance thus corresponds to a choice of stations $S \subseteq \mathcal{S}$ and channels $C \subseteq \overline{\mathcal{C}}$ to pack into, with domains $\mathcal{D}$ and interference constraints $\mathcal{I}$ implicitly being restricted to S and C; we call the resulting restrictions D and I.

3.2 Structure of Constraints

Interference constraints are more structured than in the general formulation. All constraints are of the form $\mathrm{ADJ}(s,s',c,i)$ with $i \in \{0,1,2\}$ prohibiting concurrent assignment of s to c and s' to any channel in $[c,c+i]$. Note that when $i = 0$ a constraint simply says that two stations may not be concurrently assigned to a particular channel; these are known as *co-channel constraints*. When $i > 0$ the constraint is known as an *adjacent-channel constraint*. Notice that it is very simple to translate the constraints into a large set of forbidden station–channel pairs of the form $\{(s,c),(s',c)\}$, $\{(s,c),(s',c+1)\}$, or $\{(s,c),(s',c+2)\}$. One notable structural feature of the constraints is that adjacent-channel constraints always im-

ply the existence of a set of related constraints: the property holds that for $i > 0$, $\text{ADJ}(s, s', c, i) \implies \text{ADJ}(s, s', c+1, i-1)$. This means, for example, that each $i = 1$ adjacent-constraint between two stations on c implies a co-channel constraint between the two stations on $c+1$. Other structural features include that constraints where $i = 2$ always involve a Canadian station and that no interference constraint involves channels in more than one frequency band.

Finally, it is interesting to visualize the constraint structure. Let us define the *interference graph* as an undirected graph in which there is one vertex per station and an edge exists between two vertices s and s' if the corresponding stations participate together in any interference constraint: i.e., if there exist $c, c' \in C$ such that $\{(s, c), (s', c')\} \in I$. If we were dealing exclusively with co-channel constraints, station repacking problems would be graph coloring problems, where stations are vertices and channels are colors, which is a simple way to see that the station packing problem is NP-complete. Figure 3.1 shows the US and Canada interference graph. Notice that the graph is very densely connected in areas such as New York. While this graph includes both UHF and VHF constraints, it is interesting to note that when restricting the graph to only showing UHF constraints, the eastern component of the graph is disconnected from the western components.

3.3 Structure of Station Repacking Problems in the Reverse Auction

Why should we hope that an NP-complete problem can be solved effectively in practice? First, we only need to be concerned with problems involving subsets of a fixed set of stations and a fixed set of interference constraints: those describing the television stations currently broadcasting in the United States and Canada. Furthermore, we are not interested in worst-case performance (i.e., the hardest possible problems within these limits), or even average case performance, but rather in good performance on the sort of instances generated by actual reverse auctions. For example, the repacking problems encountered depend on the order in which stations exit the auction, which depends on stations' valuations, which depend in turn (among many other factors) on the size and character of the population reached by their broadcasts. The distribution over repacking problems is hence far from

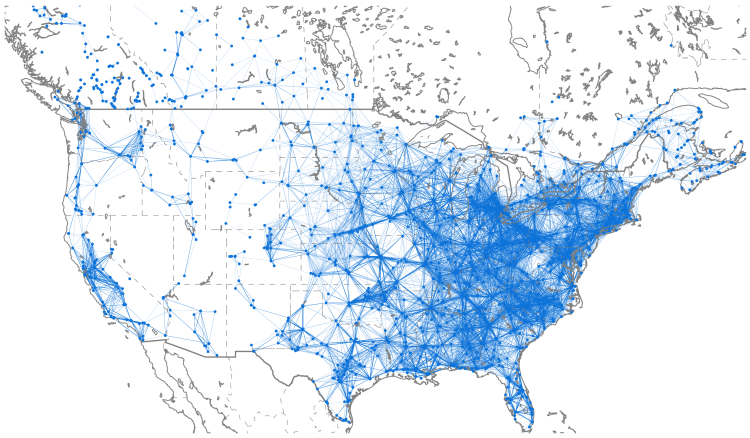


Figure 3.1: Interference graph derived from the FCC's November 2015 constraint data [14].

uniform. Second, descending clock auctions repeatedly generate station repacking problems by adding a single station s^+ to a set S^- of provably repackable stations. This means that every station repacking problem $(S^- \cup \{s^+\}, C)$ comes along with a partial assignment $\gamma^- : S^- \to C$ which we know is feasible on restricted station set S^-.

3.4 Complete and Local Search Solvers

Before heading into a discussion of problem solving techniques, we quickly review the differences between complete and local search SAT solvers, since some of our techniques only make sense for one or the other. Complete solvers are typically based on backtracking algorithms such as DPLL [7] in which variables are tentatively assigned and the consequences of the assignments are propagated until a conflict emerges, at which point the solver backtracks, or until a feasible assignment is found. These solvers are able to exhaust the search space and prove

that a problem has no feasible solution. Local search solvers, on the hand, work by searching a space of complete assignments and seeking a feasible point, typically following gradients to minimize an objective function that counts violated constraints, and periodically randomizing. Unlike complete solvers, local search solvers cannot prove that a problem is infeasible. A final important difference is that while local search solvers maintain a full assignment of the variables as part of their state, complete solvers do not.

3.5 Encodings

In this section we describe various ways of encoding the station repacking problem so that it can be solved using standard methods.

3.5.1 MIP Encoding

The FCC's initial investigations included modeling the station repacking problem as a MIP and using off-the-shelf solvers paired with problem-specific speedups. Unfortunately, the problem-specific elements of this solution were not publicly released, so we do not discuss them further in this document. The station repacking problem can be encoded as a MIP straightforwardly as follows:

$$x_{s,c} + x_{s',c'} \leq 1 \; \forall \left\{ (s,c), (s',c') \right\} \in I \tag{3.1}$$

$$\sum_{c \in D(s)} x_{s,c} = 1 \qquad \forall s \in S \tag{3.2}$$

$$x_{s,c} \in \{0,1\} \qquad \forall c \in D(s) \forall \, s \in S \tag{3.3}$$

Constraint 3.1 ensures that no interference constraints are violated, Constraint 3.2 ensures that each station gets assigned to exactly one channel, and lastly, Constraint 3.3 makes the variables integral.

3.5.2 SAT Encodings

We decided to instead consider SAT encodings: this formalism is well suited to station repacking, which is a pure feasibility problem (no objective function) with

only combinatorial constraints.[1] A SAT encoding has the advantage of making it possible to leverage the research community's vast investment into developing high-performance SAT solvers (see e.g., Järvisalo et al. (2012)). In all of the following encodings, we create a variable $x_{s,c} \in \{0,1\}$ for every station–channel pair, representing the proposition that station s is assigned to channel c. Many SAT encodings for our problem are possible (see e.g., Prestwich (2003) for a number of applicable encodings). Here we focus on two relatively straightforward encodings: the *direct* encoding, and the *multivalued* encoding.

Direct SAT Encoding

The direct SAT encoding is analogous to the MIP encoding. We create three kinds of clauses:

$$\bigvee_{d \in D(s)} x_{s,d} \qquad \forall s \in S \tag{3.4}$$

$$\neg x_{s,c} \vee \neg x_{s,c'} \qquad \forall s \in S, \forall c, c' \neq c \in D(s) \tag{3.5}$$

$$\neg x_{s,c} \vee \neg x_{s',c'} \; \forall \{(s,c),(s',c')\} \in I \; \forall c \in D(s) \; \forall \; s \in S \tag{3.6}$$

Clause 3.4 is an at least one (ALO) constraint, ensuring that every station gets assigned a channel. Clause 3.5 is an at most one (AMO) constraint, ensuring that every station gets assigned at most one channel (so together, these clauses ensure that every station gets exactly one channel). Finally Clause 3.6 ensures that interference constraints are not violated.

Multivalued SAT Encoding

The multivalued encoding is exactly the same as the direct encoding without Clause 3.5, the AMO constraint. It is trivial to transform a solution to the multivalued problem into a solution of the original problem: if a station is assigned more than one channel, we can simply pick one channel from these arbitrarily. This encoding significantly reduces the number of constraints (recall that there is an AMO

[1] Of course, it may nevertheless be possible to achieve good performance with MIP or other techniques; we did not investigate such alternatives in depth.

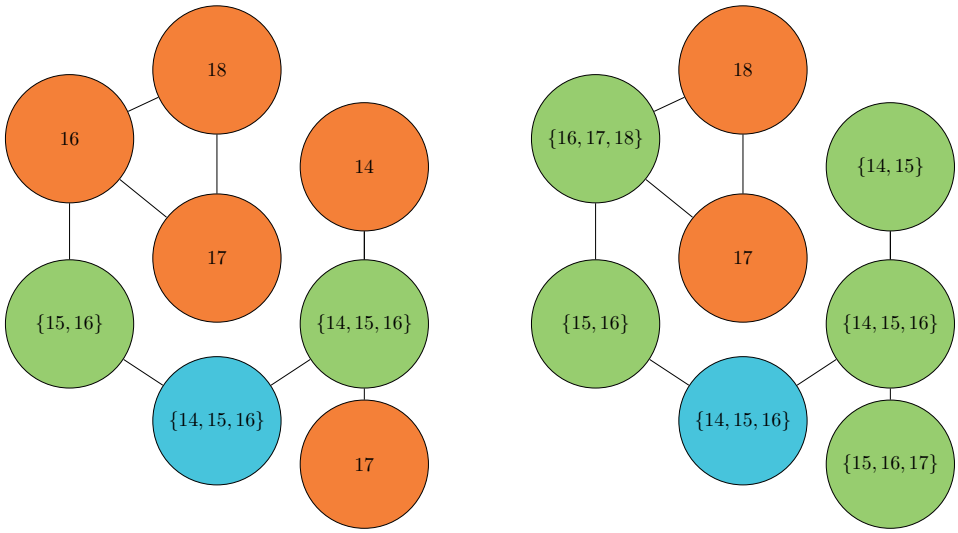
constraint for every pair of channels in a station's domain in the direct encoding), and it increases the size of the feasible region by allowing solutions where stations occupy more than one channel. This encoding is well suited to local search solvers, which cannot benefit from the constraint propagation of the AMO constraints in the same way that a complete solver can.

3.6 Using the Previous Solution

In this section we present two methods that leverage the partial assignment γ^- to quickly solve problems.

3.6.1 Locally Altering the Previous Solution

On a majority of problem instances, a simple transformation of γ^- is enough to yield a satisfiable repacking: we consider whether it is possible to assign s^+ to a channel and update the channel assignments of the stations in s^+'s neighborhood, holding the rest of γ^- fixed. Specifically, we find the set of stations $\Gamma(s^+) \subseteq S$ that neighbor s^+ in the interference graph, then solve the reduced repacking problem in which all non-neighbors $S \setminus \Gamma(s^+)$ are fixed to their assignments in γ^-. Observe that a feasible repacking for this reduced problem is also feasible on the full set; on the other hand, if we prove that the reduced problem is infeasible, we cannot conclude anything. The value of this approach is in its speed: often a station's immediate neighborhood in the induced problem's interference graph is very small, hence the reduced problem can be solved very quickly. Also note that if the reduced problem is infeasible, we can unfix additional stations and try again: a natural second step would be to also unfix stations in $\Gamma(s'), s' \in \Gamma(s^+)$, that is, neighbors of neighbors of s^+, though of course other expansion rules are possible. We proceed in this manner until a solution is found or we wind up with the initial problem. Figure 3.2 illustrates the approach. Note that it may happen that later subproblems in the series are easier to solve than earlier ones, so we impose a cutoff on each problem in the series to ensure the solver gets to attempt later subproblems even if it gets stuck on earlier ones. Also note that complete solvers have an advantage in applying this idea because they can prove problems are infeasible, whereas a local search solver would need to exhaust its entire cutoff before expanding.



(a) Locally altering the previous solution **(b)** Expanding the set of unfixed stations

Figure 3.2: The first two subproblems in a series of applying the "locally altering the previous solution" method. In 3.2a we allow the newly added station (blue) and its neighbors (green) to take on any channel in their domains, but hold the remaining stations (orange) fixed on their channels. If a feasible solution cannot be found in the first subproblem, we unfix neighbors of neighbors and try again (see 3.2b).

3.6.2 Starting Near the Previous Solution

When working with local search solvers, which maintain a full variable assignment in their state, we can leverage γ^- by assigning the stations in γ^- to their channels in γ^- and randomly assigning a channel for s^+. If a solution exists "near" this starting point, i.e., γ^- can be converted into a solution with only a few variable changes, such an initialization can help us to find it much more quickly (although there is no guarantee that the solver will not immediately randomize away to another part of the space), so we interleave initializations to the previous solution with random restarts. We observe that this approach does not generalize the "locally altering the previous solution" approach discussed in Section 3.6.1, as we do not constrain the local search algorithm to consider only s^+'s (extended) neighborhood.

3.7 Problem Simplification

In this section we provide three preprocessing methods that can be used to take a station repacking problem and transform it into a more manageable, smaller problem.

3.7.1 Arc Consistency

Station repacking problems are CSPs where all constraints are between pairs of variables. We can therefore use local consistency techniques to reduce the search space. In particular we can make a given station repacking problem *arc consistent* using the well known polynomial time Arc Consistency Algorithm #3 (AC-3) [32]. A station s is arc consistent with another station s' if whenever s is assigned to any channel $c \in D(s), \exists c' \in D(s')$ such that $\{(s,c),(s',c')\} \notin I$. If c does not satisfy this property, then no feasible assignment can ever assign s to c and so we can prune c from $D(s)$. Enforcing arc consistency therefore works as a preprocessing step to create smaller problems, though it is also possible for it to prune all of the channels in a station's domain thereby proving a problem is infeasible.

3.7.2 Unconstrained Station Removal

In a given repacking problem (S,C), there may exist stations with the following property: given any feasible assignment of all of the other stations in $S \setminus s$, there will always exist a channel in $D(s)$ into which s can be feasibly packed. We call these stations unconstrained, since we can remove them from the problem and solve a smaller problem $(S \setminus S_{\text{unconstrained}}, C)$. If the smaller problem is infeasible, so is the larger problem. If a feasible solution is found, we can sequentially add the unconstrained stations back by iterating over their domains until a feasible channel is found. Testing whether or not a station is unconstrained is a co-NP hard problem, so we rely on a sound but incomplete set of heuristics to identify unconstrained stations.

Consider that each neighbor $s' \in \Gamma(s)$ can be assigned to any channel $c' \in D(s')$, and in doing so would make any solution where s was assigned to a channel in $\{c \mid c \in D(s), \{(s,c),(s',c')\} \in I\}$ infeasible. A station is unconstrained if there is no feasible way for the neighbors to select channels in such a way that they block

out all of $D(s)$. We can relax the problem by removing the requirement that the neighbors' assignment must be feasible. If a station is unconstrained in the relaxed problem, it will also be unconstrained in the base problem since imposing a feasible assignment on the neighbors only makes blocking out channels more difficult.

Unblockable Channel Heuristic

Our first heuristic is simple: If there exists a channel on s's domain that no neighbor has the ability to block, then s is unconstrained since it can always be assigned to this channel. More formally, s is unconstrained if $\exists\ c \in D(s)$ such that $\forall s' \in \Gamma(s),\ \forall c' \in D(s'), \{(s,c),(s',c')\} \notin I$. This heuristic can be computed in time $\mathcal{O}(|\Gamma(s)| \cdot |D(s)| \cdot d)$, where d is the size of the largest neighbor's domain and checking whether or not a constraint exists in I is an $\mathcal{O}(1)$ operation.

Worst Case Selection Heuristic

Our second heuristic is based on upper bounding the number of channels that neighboring stations can block. If this upper bound is smaller than $|D(s)|$, s is unconstrained. To compute this upper bound, we observe that every neighboring station s' has at least one channel that it could be assigned to on which it blocks the maximum number of channels possible for s' to block on $D(s)$. Let $\text{Block}_{s'}^{s}$ be the maximum number of channels s' can block for s:

$$\text{Block}_{s'}^{s} = \max_{c' \in D(s')} \left| \left\{ c \in D(s) \mid \left\{ (s,c), \left(s',c'\right) \right\} \in I \right\} \right| \tag{3.7}$$

We compute an upper bound on the total number of channels that can be blocked for s by assigning all of the neighbors to such a maximum blocking channel and assuming that the blocked sets are non-overlapping. If $\sum_{s' \in \Gamma(s)} \text{Block}_{s'}^{s} \leq |D(s)|$, then s has more channels on its domain than can possibly be blocked by other stations, so s is unconstrained. This heuristic is also computable in time $\mathcal{O}(|\Gamma(s)| \cdot |D(s)| \cdot d)$.

Our two heuristics complement each other: the first heuristic only applies when a station has a constraint-free channel, whereas the second heuristic can fail to identify a station with a constraint-free channel as unconstrained but can identify

unconstrained stations that do not posses such a channel.

Recursively Unconstrained Stations

We use the above two heuristics to identify unconstrained stations. Notice however that in the smaller problem induced by removing unconstrained stations, new stations may become unconstrained that were not unconstrained in the original problem. Therefore, we recursively run our unconstrained station removal procedure until it can no longer remove any stations. Note that when adding back the unconstrained stations after solving the smallest induced problem, we must add stations back at same the level of recursion in which they were removed, since unconstrained stations at higher recursion depths are only unconstrained conditional on the removal of stations at shallower recursion levels.

3.7.3 Problem Decomposition

The interference graph induced by a problem may consist of multiple connected components; we can separate them in linear time. We only need to solve the one component that contains s^+, as we can directly leverage γ^- to fill in a feasible assignment for the other components. This is particularly useful for complete solvers, which cannot otherwise leverage γ^-. Note that arc consistency and unconstrained station removal can remove edges and links in the interference graph, possibly shrinking the size of the component containing s^+.

3.8 Meta-Algorithmic Techniques

In recent years, there has been increasing development of artificial intelligence techniques that reason about how existing heuristic algorithms can be modified or combined together to yield improved performance on specific problem domains of interest. These techniques are called *meta-algorithmic* because they consist of algorithms that take other algorithms as part of their input. For example, *algorithm configuration* consists of setting design decisions exposed as parameters to optimize an algorithm's average performance across an instance distribution. This approach has proven powerful in the SAT domain, as many SAT solvers expose parameters that can drastically modify their behavior, from the probability of random

restarts to the choice of search heuristics or data structures [22]. We performed configuration using Sequential Model-based Algorithm Configuration (SMAC) [21] in order to tune the performance of generic SAT solvers to station repacking problems. Many ideas in this chapter, such as the choice of what encoding to use discussed in Section 3.5, or the preprocessing ideas in Section 3.7 can be exposed as meta-parameters to an algorithm configurator; we did so whenever possible.

Unfortunately, even after performing algorithm configuration, it is rare to find a single algorithm that outperforms all others on instances of an NP-complete problem such as SAT. This inherent variability across solvers can be exploited by *algorithm portfolios* [19, 37]. Most straightforwardly, one selects a small set of algorithms with complementary performance on problems of interest and, when asked to solve a new instance, executes them in parallel.

Finally, algorithm configuration and portfolios can be combined. Hydra [41] is a technique for identifying sets of complementary solvers from highly parameterized design spaces via algorithm configuration. It operates by altering an algorithm configurator to optimize an algorithm's marginal gains over an existing portfolio: this allows for discovery of algorithms that might perform poorly where existing algorithms are known to do well but that can outperform known configurations on other problems. In each iteration, Hydra greedily adds configurations that make the greatest possible marginal contribution to an existing portfolio. Specifically, we create a (parallel) portfolio by greedily selecting the algorithm that most improves its performance.

3.9 Containment Caching

We know that every repacking problem we encounter will be derived from a restriction of the interference graph to some subset of $\mathcal{S}$. We know this graph in advance of the auction; this suggests the possibility of doing offline work to precompute solutions. However, this graph involves a total of $|\mathcal{S}| = 2990$ stations, and the number of possible subsets is exponential in this number. Thus, it is not possible to exhaustively range over all possible subsets prior to the auction.

One possibility would simply be to store the solution to every repacking problem we encounter and build a giant cache, but in practice we found that identical

problems were rarely encountered across auction simulations. However, observe that if we know whether or not it is possible to repack a particular set of stations S, we can also answer many different but related questions. Specifically, if we know that S was packable then we know the same is true for every $S' \subseteq S$ (and indeed, we know the packing itself—the packing for S restricted to the stations in S'). Similarly, if we know that S was unpackable then we know the same for every $S' \supseteq S$. This observation dramatically magnifies the usefulness of each cached entry S, as S can be used to answer queries about an exponential number of subsets or supersets (depending on the feasibility of repacking S). This is especially useful because sometimes it can be harder to find a repacking for subsets of S than it can be to find a repacking for S (it may be that the added interference constraints constrain the search space in a way that makes a solution easier to find). Analogously, it might sometimes be easier to prove a smaller subset of S is infeasible than S itself.

We call a cache meant to be used in this way a *containment cache*, because it is queried to determine whether one set contains another (i.e., whether the query contains the cache item or vice versa). To the best of our knowledge, containment caching is a novel idea. A likely reason why this scheme is not already common is that querying a containment cache is nontrivial, as we will explain below. We observe that containment caching is applicable to any family of feasibility testing problems generated as subsets of a master set of constraints, not just to spectrum repacking.

In more detail, containment caching works as follows. We maintain two caches, a *feasible cache* and an *infeasible cache*, and store each problem we solve in the appropriate cache. We store full instances for SAT problems and the smallest simplified component (created by applying all of the methods from Section 3.7 to a problem instance) for UNSAT problems, in order to be able to service the largest number of possible queries. When asked whether it is possible to repack station set S, we proceed as follows. First, we check whether a subset of S belongs to the infeasible cache, in which case the original problem is infeasible. If we find no matches, we decompose the problem into its smallest simplified component and check if the feasible cache contains a superset of those stations, in which case the original problem is feasible.

Containment caching is less straightforward than traditional caching, because

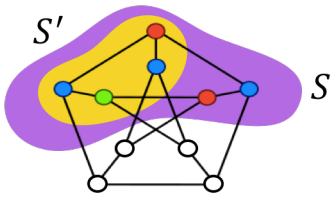


Figure 3.3: This figure shows a solution to a graph coloring problem S that can trivially be made into a solution for the graph coloring problem S' by restricting the solution for S to vertices in S'. This is analogous to how our SAT cache works.

we cannot simply index entries with a hash function. Instead, an exponential number of keys could potentially match a given query. We were nevertheless able to construct an algorithm that solved this problem quickly[2] Specifically, our approach proceeds as follows. Offline, we build (1) a traditional cache $\mathfrak{C}$ indexed by a hash function and—in the case of feasible problems—storing solutions along with each problem; and (2) a secondary cache $\mathfrak{C}_o$ containing only a list of station sets that appear in $\mathfrak{C}$. This secondary cache is defined by an ordering o over $\mathcal{S}$, which we choose uniformly at random. We represent each station set stored in $\mathfrak{C}_o$ as a bit vector, with the bit in position k set to 1 if and only if the k-th station in ordering o belongs to the given station set. We say that one station set is larger or smaller than another by interpreting both station set bit vectors as integers under the ordering o and then comparing the integers. Appealing to this ordering, we sort the entries of $\mathfrak{C}_o$ in descending order. Figure 3.4 illustrates a set of six subsets of the power set $2^{\{a,b,c,d,e\}}$ and a secondary cache constructed based on these sets along with a random ordering over their elements. As the figure suggests, secondary caches are very compact: we can thus afford to build multiple secondary caches $\mathfrak{C}_{o_1}, \ldots, \mathfrak{C}_{o_\ell}$, based on the same set of station sets but ℓ different random orderings.

We now explain how to query for a superset, as we do to test for feasible solutions; the algorithm for subsets is analogous. A sample execution of this algorithm is illustrated in Figure 3.5. Given a query S, we perform binary search on each of the ℓ secondary caches to find the index corresponding to S itself (if it is actually

[2]There is a literature on efficiently finding subsets and supersets of a query set [4, 20, 39]. However, our algorithm was fast in our setting and we did not explore alternatives; indeed, our approach may be of independent interest.

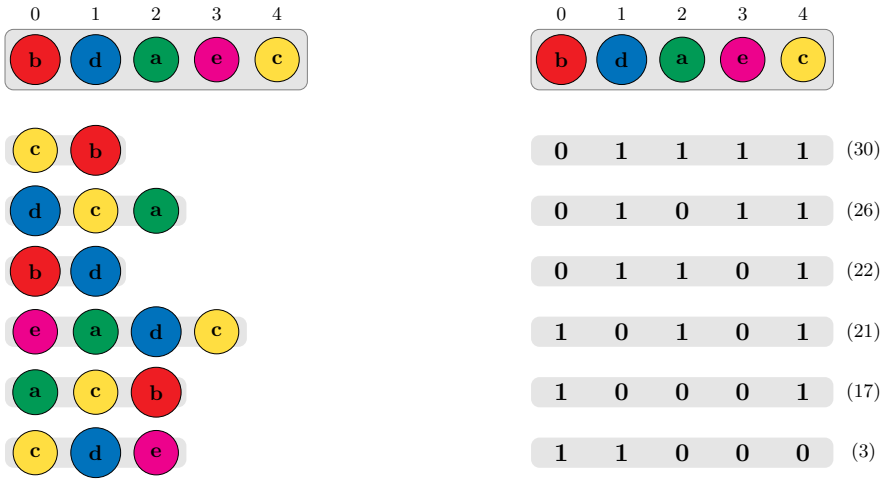


Figure 3.4: Translating six sets (left) into integers (right) according to a secondary cache ordering.

stored in the cache) or of the smallest entry larger than S (if not); denote the index returned for cache $\mathfrak{C}_{o_k}$ as i_k. If we find S, we are done: we retrieve its corresponding solution from the main cache. Otherwise, the first i_1 entries in cache $\mathfrak{C}_{o_1}$ contain a mix of supersets of S (if any exist) and non-supersets that contain one or more stations not in S that appear early in the ordering o_1. Likewise, the first i_2 entries in $\mathfrak{C}_{o_2}$ contain the same supersets of S (because $\mathfrak{C}_{o_1}$ and $\mathfrak{C}_{o_2}$ contain exactly the same elements) and a different set of non-supersets based on the ordering o_2, and so on. We have to search through the first i_k entries of some cache $\mathfrak{C}_{o_k}$; but it does not matter which $\mathfrak{C}_{o_k}$ we search. We thus choose the shortest list: $k = \arg\min_j i_j$. This protects us against unlucky situations where the secondary cache's ordering yields a large i_k: this is very unlikely to happen under all ℓ random orderings for large enough ℓ. The superset search itself can be performed efficiently by testing whether the cached bit vector is bitwise logically implied by the query bit vector. If we find a superset, we query the main cache to retrieve its solution.

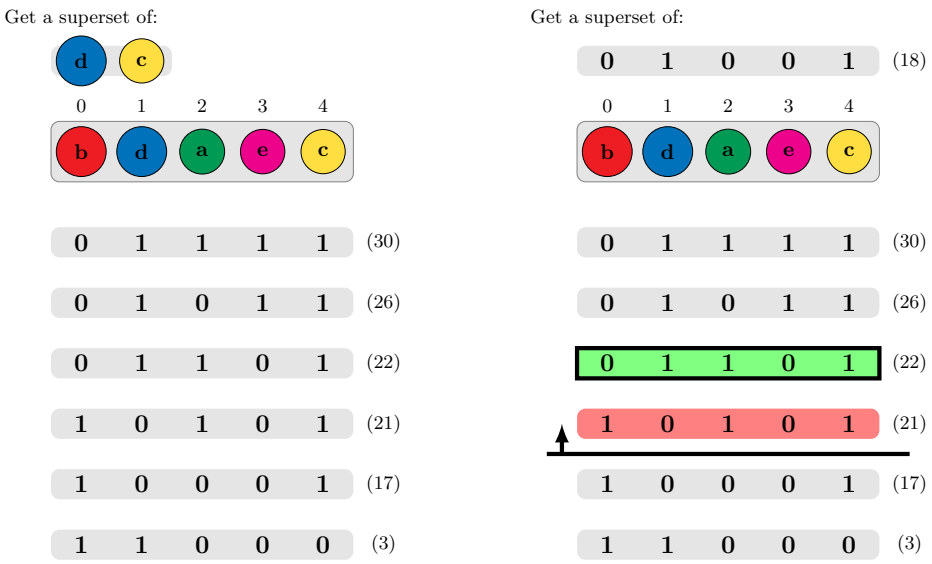


Figure 3.5: A visualization of looking up a superset in a secondary cache. Only one secondary cache is shown in these figures, but multiple secondary caches can be used to decrease the number of elements to search over.

3.10 SATFC

We combined all of the methods described in this chapter to create our feasibility checker, SATFC. Using problems from FCC simulations, we performed algorithm configuration as described in Section 3.8 on 18 different SAT solvers, obtained mainly from SAT solver competition entries collected in AClib [23]. We observed good performance from the solvers `Clasp` [18] and `SATenstein` [27]. `Clasp` is a highly-parameterized complete solver that relies on conflict-driven nogood learning, and `SATenstein` is a local search framework that can instantiate most existing high-performance local search algorithms for SAT in the literature, as well as novel solvers that have never been thought of before. We then used Hydra to create an eight-algorithm portfolio of various parameterizations of these two solvers. Together with the cache described in Section 3.9 these algorithms make up SATFC

Portfolio
`Clasp` (Direct encoding, Locally altering the previous solution)
`Clasp` (Direct encoding, Fully simplified problems)
`Clasp` (Direct encoding, Fully simplified problems)
`SATenstein` (Direct encoding, Previous solution restarts)
`SATenstein` (Multivalued encoding, Previous solution restarts)
`SATenstein` (Multivalued encoding, Previous solution restarts, Problem decomposition)
`SATenstein` (Multivalued encoding, Previous solution restarts, Problem decomposition)
`SATenstein` (Multivalued encoding, Previous solution restarts, Fully simplified problems)

Table 3.1: A breakdown of the different solvers composing SATFC showing the different ideas from this chapter that are used in each algorithm.

2.3.1, the version used by the FCC.[3] In particular, the portfolio contains three `Clasp` threads, all of which use the direct encoding. One is used for the locally altering the previous solution idea of Section 3.2a and the other two operate on fully simplified components. The remaining five threads are `SATenstein`, all of which sometimes restart to previous solutions, and four of which use the multivalued encoding. The portfolio composition is outlined in Table 3.1.

[3]The set of configuration experiments that produced this version's portfolio are unpublished. We acknowledge some limitations of these experiments here. Firstly, instead of merging the design spaces of `Clasp` and `SATenstein` and then running Hydra on the merged design space, we performed separate runs of Hydra on each of their design spaces in isolation. As a result, we were only able to exploit complementarity available within each of these solvers' design spaces—and not across them—during configuration. We assembled the final portfolio greedily by evaluating configurations from both Hydra runs on a validation set, so complementarity *could* be exploited between the two design spaces amongst the configurations that were found. A second issue with our configuration experiments is that we did not extract the "locally altering the previous solution" idea described in Section 3.6.1 as a meta-parameter, and instead tuned it manually. This work does not contain any new configuration experiments, and hence does not address either of these flaws.

Chapter 4

Designing a Reverse Auction Simulator

One of our main contributions is a simulator for the reverse auction algorithm. Much of the work involved in building a simulator is implementing all of the auction rules, which we explained in Chapter 2 and do not repeat here. In this chapter, we focus on what still needs to be specified in order to actually run a simulation. First, we need a model of how stations will bid in the auction, which we give in Section 4.1. Next, we need to come up with value profiles for these stations; we describe how we generate these in Section 4.2. We also need to select what clearing target to use; we go over how we select clearing targets in Section 4.3. Another decision is what feasibility checker to use: SATFC is one option; we describe another in Section 4.4. Section 4.5 explains two methodologies for dealing with timeouts. Lastly, Section 4.6 describes how we can save calls to the feasibility checker by reasoning about cases where a station must have remained feasible between checks.

4.1 Station Bidding Model

In order to actually run a simulation, we need a model of how stations bid. We assume that each station s has a private value for broadcasting on each of its permissible bands, $v_{s,b}$, and that in each bidding round, a station selects the offer that

myopically maximizes its utility $P_{b,s,t} + v_{b,s}$. Fallback options (see Section 2.2.2), when required, are also selected in this manner.

4.2 Station Valuation Model

In this section we describe how we generate value profiles for each station. Let $v_{s,\text{OFF}} = 0$, since a station gets no value if it gives up its license. To assign the remaining values, we turned to the valuation model developed by Doraszelski et al. (2016). Their work uses data from the MEDIA Access Pro Database from BIA Kelsey (BIA) and the Television Financial Report from the National Association of Broadcasters (NAB) to estimate station valuations for UHF stations. In their model, a station's value is the maximum of its cash-flow value and its stick value, where the former reflects the "the price a TV station expects if it sells itself on the private market as a going concern" and the latter purely reflects the value of the broadcast license.

$$v_{s,\text{UHF}} = \max\{v_s^{\text{CF}}, v_s^{\text{Stick}}\} \tag{4.1}$$

The cash flow value is computed by multiplying a station's cash flow CF_s with a *cash flow multiple* $\text{Multiple}_s^{\text{CF}}$.

$$v_s^{\text{CF}} = \text{CF}_s \cdot \text{Multiple}_s^{\text{CF}} \tag{4.2}$$

The stick value is proportional to the interference-free population that a station reaches and a *stick multiple* $\text{Multiple}_s^{\text{Stick}}$.

$$v_s^{\text{Stick}} = \text{Multiple}_s^{\text{CF}} \cdot 6\text{Mhz} \cdot \text{Population}\,(s) \tag{4.3}$$

where 6 Mhz is the size of a channel. While we know the population values for each station, we need to estimate cash flows and both of the multiples in order to apply this model. The authors of Doraszelski et al. (2016) shared with us distributions that we can use to sample estimates of these unknown values. This allows us to apply the above equations to produce samples of $v_{s,\text{UHF}}$. Now that we have entries for $v_{s,\text{UHF}}$, we fill in the VHF values by making the simplifying assumption that $v_{s,\text{HVHF}} = \frac{2}{3} v_{s,\text{UHF}}$ and that $v_{s,\text{LVHF}} = \frac{1}{3} v_{s,\text{UHF}}$.

4.3 Determining Clearing Target

In order to simulate a reverse auction, we need to select a clearing target. Prior to the auction, the FCC announced a series of ten clearing targets, ranging from 126–42 Mhz (corresponding to a maximum allowable channel of 29–44). In the incentive auction, in order to avoid limiting the clearing target by the most constrained regions of the country, some *impairments* are allowed, meaning that some stations can be placed on channels otherwise disallowed by the clearing target. Section 3 of an FCC public notice [15] details the *initial clearing target determination procedure*, which involves optimizing a tentative assignment for every clearing target and selecting the most aggressive clearing target that meets a standard set for tolerable impairments. One other role of the initial clearing target determination procedure is to set the starting band of each participating station. Prior to the auction, stations select all of the options that they would be willing to consider at the opening prices. While there is unlimited room in OFF, if too many stations restrict their willingness to participate only to VHF bands it may be impossible to find an initial feasible assignment that accommodates all of them in their preferred options. The initial clearing target determination procedure decides which stations to initially place on each band. If a station did not indicate OFF as an allowable option, it may place the station on its home band, causing it to exit.

The initial clearing target determination procedure is somewhat complex, and we do not simulate it. Instead, we use the following procedure to determine the clearing target and find an initial assignment. In our simulator, only those stations that are offered more to go off-air than their values for broadcasting in their home bands (i.e., those stations for which $P_{0,s,\text{OFF}} > v_{s,\text{HOME-BAND}(s)}$) will participate in the auction. Our simulator places all participating stations in OFF at the beginning, even if their most preferred offer would have led them to starting in a VHF band (though they are free to bid for their preferred option in the first round). As a consequence of this restriction, we do not allow stations that would have had positive utility for starting in one of the VHF bands but negative utility from starting in OFF to participate in the auction in our simulator. We make this restriction to ensure that stations always have positive utilities for participating: our concern is that a station might freeze in the first round before it is able to move to its preferred option, and

then it would have a negative utility for participating.

After the set of participating stations is determined, we select a clearing target by iterating through all possible clearing targets in order, starting with the most aggressive, and selecting the first clearing target for which we are able to find a feasible assignment for the non-participating stations. We do not simulate the possible multi-stage nature of the incentive auction, so this clearing target will be used throughout the entire simulation.

4.4 Feasibility Checker

In order to run a reverse auction simulation, we need to select a feasibility checker. Since we want to investigate the impact that the quality of a feasibility checker has on the efficiency and cost of the reverse auction, we need alternative feasibility checkers to compare with SATFC. We propose what is perhaps the simplest reasonable feasibility checker, which we call the *greedy feasibility checker*. It simply iterates through the domain of s^+ checking to see if any channel results in a feasible assignment when augmented with γ^- (as shown in Figure 4.1). It does not alter γ^- in any way. If it cannot find a feasible channel for s^+, the greedy feasibility checker reports a timeout. We use the greedy feasibility checker as a zero reference point for feasibility checker quality in order to measure the impact of feasibility checker quality on auction efficiency and cost (although we do not know that either of these two metrics monotonically improve with feasibility checker quality).

4.5 Revisiting Timeouts

Recall that the bid processing step of the reverse auction (see Section 4.5) loops over a queue of stations, processing the first station found to be feasible, removing it form the queue, and continuing in this manner until an iteration in which it cannot prove any remaining stations in the queue are feasible. We need to decide how to handle timeouts during this step of the reverse auction. If a station is found infeasible in UHF, it will remain that way until the end of the auction (because no stations leave UHF). Therefore, when we find a station to be infeasible in UHF in one loop iteration of the bid processing step, then in subsequent loops over the queue we can skip the call to the feasibility checker for that station because that station must still

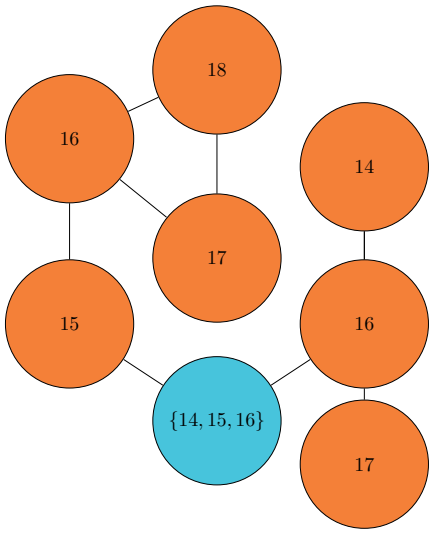


Figure 4.1: An illustration of the greedy feasibility checker. Only the newly added station (blue) can vary on its domain, other stations are all fixed on their channels.

be infeasible. However, if instead the result of a UHF feasibility check is a time-out, a future call to the feasibility checker in that same bid processing step may be able to prove that the station is actually feasible in UHF (as more stations exit, the problem of finding a feasible repacking for that station could become easier). In other words, just because a station is skipped over the first time it is examined in the queue does not mean it should necessarily be skipped over again during the next iteration. We therefore need to decide whether to run our feasibility checker again the next time the station is processed in bid processing and other stations have exited to UHF since the last check (we refer to this as *revisiting timeouts*), or to simply declare that the station is infeasible because it timed out once and will likely time out again. The latter approach can save computational resources, but may harm the reverse auction's cost and efficiency. Our simulator has supports both approaches, with the default behavior being to not revisit timeouts.

4.6 Reusing Solutions Within a Simulation

Within a given auction simulation, we can reuse feasible solutions to UHF problems in order to reduce the number of queries to the feasibility checker. The idea is that once a station s is proven feasible to pack in UHF, it will remain so as long as no other station s' exits into UHF where s and s' are in the same connected component of the interference graph. Therefore, once we find a feasible assignment for s in UHF, we mark that s is feasible and skip future calls to the feasibility checker regarding placing s in UHF (since we know that s remains feasible). Whenever a station exits to UHF, we clear our stored results for any stations in the same component of the interference graph as the exited station.

Chapter 5

Experiments

In this chapter, we perform experiments using the reverse auction simulator described in Chapter 4 in order to evaluate the performance of SATFC and to investigate the feasibility checker's impact on reverse auction outcomes. First, we explain how we ran our simulations (Section 5.1). Next, we create a test set of problems (Section 5.2) and use it to evaluate solvers (Section 5.3). We then evaluate the containment cache (Section 5.4). Lastly, we compare the reverse auction mechanism with VCG and observe how replacing SATFC 2.3.1 in our simulator with alternate feasibility checkers affects reverse auction outcomes (Section 5.5).

5.1 Simulations

We ran 20 reverse auction simulations. For each simulation, we used a different random seed to create a value profile as described in Section 4.1. The FCC released a document announcing the opening prices for each station on November 2015 [16]. To initialize benchmark prices (see Section 2.2.1) we followed this document and set $p_{0,\text{UHF}} = 0$, $p_{0,\text{HVHF}} = 360$, $p_{0,\text{LVHF}} = 675$, and $p_{0,\text{OFF}} = 900$. We also used the volumes for each station that were listed in this document, so that our opening prices matched up with the incentive auction's opening prices. We furthermore used the opening price document as our eligibility criteria for the simulator: we allowed any station that was offered an opening price to participate in the reverse auction in our simulations. The document lists stations for which "the auction

system has determined that this station will always have a feasible channel assignment in its pre-auction band at all of the possible auction clearing targets"—we simply dropped such stations from our simulations. We also dropped stations for which our valuation model contains no information, which can happen due to any of three reasons: Firstly, Doraszelski et al. only considered stations in mainland US and Hawaii, so we did not have valuations for stations outsides of these areas. Secondly, Doraszelski et al. did not model VHF station values, so we dropped all 39 eligible US LVHF stations and all 378 eligible US HVHF stations from our simulations. Lastly, since Doraszelski et al. used a different source to determine eligibility, we were unable to use their model to sample valuations for some of the stations that were eligible in our source but not in theirs[1]. After dropping these stations, we were left with 1,638 eligible US UHF stations.

Canadian stations are also involved in the reverse auction. They act exactly like stations that decide not to participate: they cause interference with other stations and must be repacked. Since Canadian stations are not bidders in the reverse auction, we did not need to model their valuations and hence we included all of the Canadian stations in all our simulator. In total, we included 113 Canadian LVHF stations, 332 Canadian HVHF stations, and 348 Canadian UHF stations in our simulations.

We downloaded the interference constraints and station domains from the FCC's website [14] posted November 12, 2015. For each simulation, we determined participation and set the clearing target using the method described in Section 4.3. In all cases this led to a clearing target of 84 Mhz, corresponding to a maximum allowable channel of 36 for US stations and 35 for Canadian stations.[2] We used SATFC 2.3.1 as our feasibility checker, with a cutoff of 60 seconds and did not revisit timeouts. All of our experiments were run on Intel Xeon E5-2640 v2 processors on nodes with 96 GB of RAM running Red Hat Enterprise Linux Server release 6.7. We allocated 8 cores to each simulation, so that all 8 configurations in SATFC 2.3.1 could run in parallel. The simulations took between 3.52 and 4.02

[1]Specifically we dropped 25 UHF stations with the following facility IDs: 4353, 259, 71425, 70414, 70415, 70423, 70426, 70428, 168094, 41375, 38562, 38437, 17830, 57905, 8500, 68406, 51656, 34894, 34341, 34342, 14315, 27501, 57456, 66549, 69753.

[2]At every clearing target, the maximum allowable channel for Canadian stations is always 1 channel below the maximum US channel.

hours to run (wall time), of which the majority of the time—between 2.52 and 3.07 hours—was spent within SATFC solving UHF problems. In each simulation between 1,553 and 1,571 stations participated at the opening prices.

5.2 Creating a Test Set of Non-Trivial Problems

The feasibility checker must solve tens of thousands of station repacking problems in a single auction simulation. However, the vast majority of these problems are not very difficult to solve. For example, even in our earliest official release, SATFC 1.3, we never observed any timeouts on VHF problems, likely because these bands contain at most 7 channels and are home to fewer stations. In addition, most UHF problems, especially earlier in the auction, can be solved by the simple greedy feasibility checker described in Section 4.4; we observed that across all of our simulations 97.39% of UHF problems can be solved this way. When creating a test set of station repacking problems, we first filter out these "trivial" problems. Across all of our simulations, this left us with a pool of 60,057 problems. We then sampled 10,000 of these problems uniformly at random to use as our test set.[3] This test set consists of 9,482 feasible problems, 121 infeasible problems, and 397 problems that timed out at our one minute cutoff and therefore have unknown feasibility.

5.3 Evaluating Test Set Performance

As mentioned in Section 3.10, in our initial experiments [17] we configured SAT solvers from AClib [23] in order to determine which SAT solvers to include in SATFC. We do not repeat these configuration experiments here (i.e., try to build a newer, better SATFC) and instead study the version of SATFC that is actually being used in the reverse auction.[4] Figure 5.1 shows the performance of the default

[3] We acknowledge that a test set is an imperfect way to evaluate a feasibility checker, since unless the test set is generated by a perfect feasibility checker that never times out, scoring well on a test set does not necessarily imply good performance in practice. We generated our test set using SATFC 2.3.1 as our feasibility checker, which encounters relatively few timeouts, and for which we have observed that most unsolved problems turn out to be infeasible (and hence would not change the problem trajectory of an auction). Nevertheless, it is difficult to reason about the true ceiling in this setting, and there is certainly a larger gap than that implied merely by the fraction of unsolved problems in our test set.

[4] This is not to say that running new configuration experiments would not be very interesting—on the contrary, we view this as promising future work!

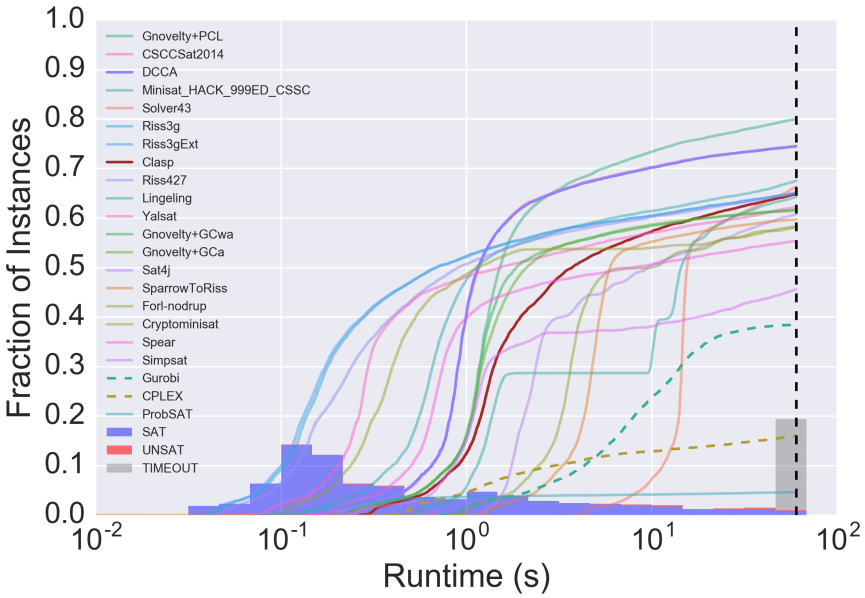


Figure 5.1: ECDF of runtimes for default configurations of MIP and SAT solvers on our test set. The legend is ordered by percentage of problems solved before the cutoff. The bars show fraction of SAT and UNSAT instances binned by their (fastest) runtime. Although present, unsatisfiable instances form an insignificant portion of instances solved.

configurations of 20 SAT solvers from AClib[5] as well as what are arguably the two best-performing MIP solvers—`CPLEX` and `Gurobi`.[6]

Note that with one exception, the SAT solvers outperformed the MIP solvers. We observed that one solver, `Gnovelty+PCL`, was able to solve the largest number of problems in our test set within the cutoff—79.96% of the problems in 41.74 hours. The parallel portfolio of all of these 22 solvers together was able to solve 81.58% of the problems in 38.08 hours. In our previous experiments [17] we observed the best performance (post algorithm configuration) from `clasp` and `DCCA` [31]. `DCAA` is a local search solver with no exposed parameters. We recruited an undergraduate student, Paul Cernek, who integrated `DCAA` into `SATenstein` and

[5]At the time of writing, there are 22 SAT solvers in AClib. In two cases (`clasp` and `lingeling`) AClib contains submissions from multiple years; we use only the latest submissions.

[6]We used `CPLEX` 12.6.2 and `Gurobi` 6.0.0.

#	Solved %	Δ %	Time (s)
1	92.91	-	53, 874
2	94.42	1.51	39, 415
3	95.14	0.72	37, 751
4	95.59	0.45	34, 967
5	95.78	0.19	34, 329
6	95.92	0.14	33, 650
7	96.01	0.09	32, 569
8	96.03	0.02	32, 503

Table 5.1: A breakdown of the marginal value of adding each new configuration to our portfolio by building the portfolio greedily.

exposed 3 parameters.

As was described previously in Section 3.10, SATFC is composed of 8 configurations of `clasp` and `SATenstein`. Figure 5.2 shows the performance of SATFC 2.3.1 and each individual configuration in SATFC 2.3.1 on the test set. SATFC 2.3.1 was able to solve 96.03% of the test set's problems in 9.03 hours (wall time), solving 87.73% of the problems in under one second. The figure shows five individual configurations in SATFC 2.3.1 which outperform any of the default solver configurations.

Another result that stands out from the figure is that the gap between the portfolio performance and the performance of the single best configuration is quite small. Table 5.1 quantifies this gap, breaking down the marginal gains of each configuration to SATFC by constructing its portfolio greedily. From this table we conclude that if SATFC 2.3.1 only ran a single configuration, it would still be able to solve 92.92% of the problems in the test set, only 3.12% fewer than the entire portfolio was able to solve. How much are the extra seven configurations contributing to the reverse auction? To better understand the value of the extra configurations, we define a new feasibility checker, the *best single configuration*, consisting solely of the top `SATenstein` configuration; we will have more to say about it in Section 5.5. We conclude by noting that the returns on both problems solved and runtime diminish fairly quickly after the fourth configuration is added.

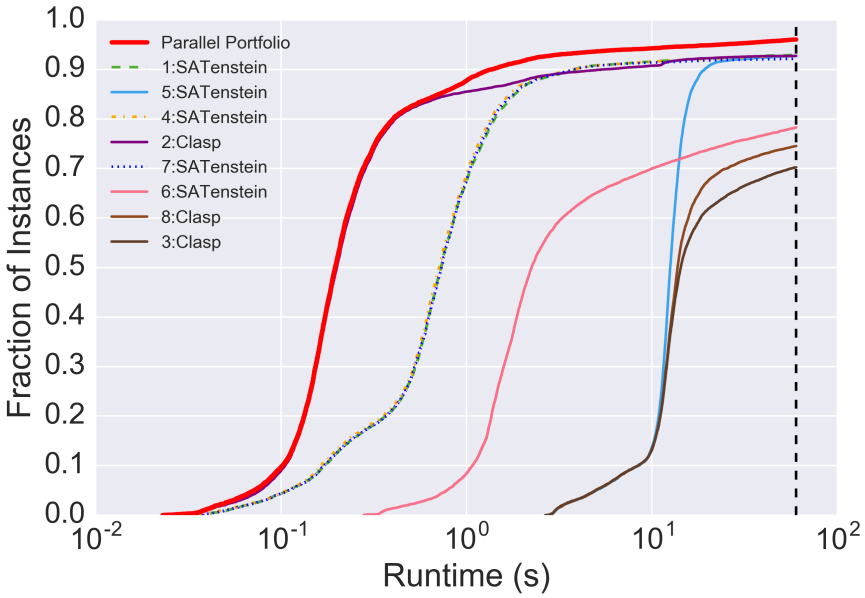


Figure 5.2: SATFC performance on test set broken down by configuration (numbers in names correspond to Table 5.1). The bold red line is the parallel portfolio performance.

5.4 Containment Cache Evaluation

We cannot evaluate our caching scheme from Section 3.9 by looking at its test set performance as we do the rest of SATFC's portfolio because we would first need to decide with what solutions the cache should be initialized. One idea might be to initialize the cache with all of the problems that we did not include in the test set, but doing so could overstate cache performance because related instances from the same auction simulation could appear both in the initialization set and the test set. We avoided this concern by performing a "leave-one-out" analysis: For each auction we built a cache composed of problems from all of the other auctions, and then computed what fraction of the non-trivial problems this cache would have been able to solve in that auction. Before performing this analysis, we checked to make sure that the problems across each simulation did not overlap (i.e., that a conventional cache would not have sufficed). We observed no auction shared

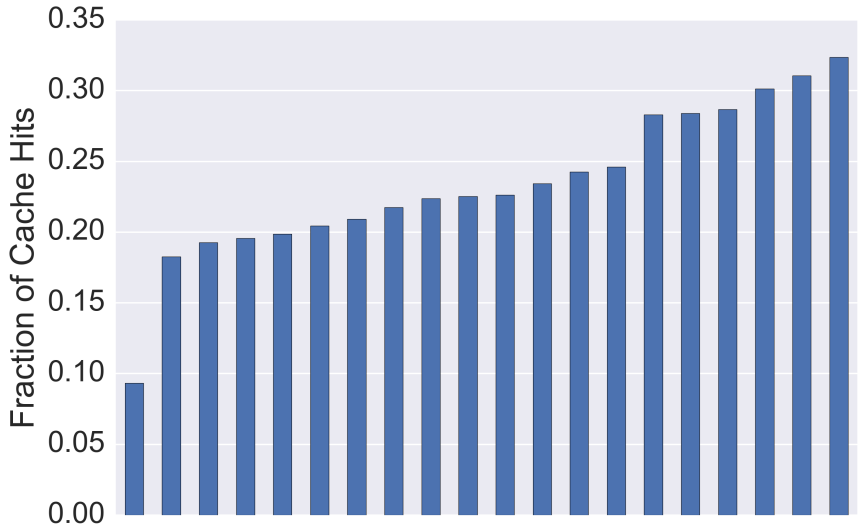


Figure 5.3: Bar chart where each bar represents the fraction of instances solvable by a cache filled with problems from the other auctions.

any non-trivial problems in common with any other auction (where we define a problem for this purpose by the set of stations to repack, ignoring the previous assignment), so a conventional cache would have a hit rate of 0% in our setting. Our results for the containment cache are shown in Figure 5.3. At worst the cache was able to solve 9.3% of the problems, and at best 32.4%, with a median of 22.6% problems solved. We note that these numbers would only improve if we added more simulations to grow the cache.

5.5 Simulations with Different Feasibility Checkers

5.5.1 Comparing Reverse Auction Outcomes

The outcome of the reverse auction is a channel assignment γ for stations that remain on-the-air and a set of prices that winning stations will be paid to go off-air. We compare reverse auction outcomes based on their efficiency and cost. A natural

way to measure efficiency is by the value preserved by the auction, in other words by defining an efficient repacking as one that maximizes the total value of the participating stations that remain on-the-air. We instead choose to measure efficiency by the total value *lost* by the auction, defining an efficient repacking as one that minimizes the total value loss of winning stations, which for any given station is measured as the difference between that station's value for broadcasting in its home band and its post-auction band. Note that an efficient repacking satisfying one definition also satisfies the other. We prefer the latter definition because it is only influenced by stations that a feasibility checker is unable to repack in their home bands and therefore does not assign credit for easy to repack stations. For example, holding everything else constant, increasing the value of an easy to repack station arbitrarily high would make the value preserved arbitrarily high as well, but value loss would be unaffected. We therefore feel that the value loss definition allows for more meaningful comparisons between different feasibility checkers. If we can find an efficient repacking γ^*, then we have an upper bound on efficiency. We can use γ^* to relate the total value loss of another repacking to the optimal total value loss through a value loss ratio:

$$\text{Value Loss Ratio} = \frac{\sum_{s \in \mathcal{S}} v_{s,\text{HOME-BAND}(s)} - v_{s,\text{POST-AUCTION-BAND}(\gamma,s)}}{\sum_{s \in \mathcal{S}} v_{s,\text{HOME-BAND}(s)} - v_{s,\text{POST-AUCTION-BAND}(\gamma^*,s)}} \tag{5.1}$$

where POST-AUCTION-BAND is a function that takes in a channel assignment and a station and returns the band that the station is assigned to, or OFF if it is not assigned to a band. Since γ^* is optimal, a value loss ratio will always be weakly greater than 1.

To get a feel for what this ratio means, imagine a setting containing two stations each with values of \$25 and one station with a value of \$100 for broadcasting in their home bands. Furthermore, imagine that the optimal repacking in this setting causes the two \$25 stations to go off-air for a total value loss of \$50. If we have a non-optimal repacking that instead causes the \$100 station to go off-air for a total value loss of \$100, then the value loss ratio of this solution would be 2, since twice as much value has been lost as in the optimal solution.

Unfortunately we cannot compute γ^* at a national scale, so we cannot compare to an optimal repacking in our national simulations. However, we can still compare

the value loss between two assignments to see which is more efficient.

For computing cost, we define a price function $\mathcal{P} : S \to \mathbb{R}$ that maps from a station to its payment (returning 0 for stations that are not winners). Then an outcome's cost is:

$$\text{Cost} = \sum_{s \in \mathcal{S}} \mathcal{P}(s). \tag{5.2}$$

We prefer outcomes with value loss ratios close to 1 and low cost. It is straightforward to compare two outcomes if one Pareto dominates the other with respect to efficiency and cost, otherwise any comparison is at least partially subjective.

5.5.2 Comparing the Reverse Auction and VCG

In this section we compare the FCC's reverse auction mechanism to a VCG mechanism. We chose to compare to VCG because it computes a globally efficient outcome and is truthful for bidders.

VCG Encoding

We now specialize VCG to our setting to code for the relevant constraints. We encode the global repacking problem as a MIP. We differentiate between non-participating stations that must be repacked in their home bands, $S_{\text{non-participating}}$, and participating stations, $S_{\text{participating}}$, that can be repacked on any band weakly below their homes or placed off-air. For simplicity, we use an objective function that maximizes the value preserved, noting that a solution to this maximization problem also minimizes the total value loss.

$$\text{maximize} \sum_{s \in S_{\text{participating}}} \sum_{c \in D(s)} x_{s,c} \cdot v_{s,\text{CHANNEL-TO-BAND}(c)}$$

$$\text{subject to } x_{s,c} + x_{s',c'} \leq 1 \qquad \forall \{(s,c),(s',c')\} \in I \tag{5.3}$$

$$\sum_{c \in D(s)} x_{s,c} \leq 1 \qquad \forall s \in S_{\text{participating}} \tag{5.4}$$

$$\sum_{c \in D(s)} x_{s,c} = 1 \qquad \forall s \in S_{\text{non-participating}} \tag{5.5}$$

$$x_{s,c} \in \{0,1\} \qquad \forall c \in D(s) \forall\, s \in S \tag{5.6}$$

The objective function maximizes the aggregate on-the-air value of the participating stations by summing over the channel indicator variables for each participating station multiplied by its value for broadcasting in that channel's band. Note that if a station is placed off-air, it does not contribute to the objective function, and that a station contributes most to the objective function when it is placed where it has the highest value, i.e., in its home band. The rest of our encoding is very similar to the MIP encoding of the station repacking problem presented in Section 3.5.1. In fact, besides the objective function there is only one difference, which is that here we differentiate between participating stations, which Constraint (5.4) says may be assigned or unassigned, and non-participating stations, which Constraint (5.5) ensures are assigned to a channel in their home band.[7]

The VCG price for a winning station s is calculated as the difference (excluding the value of s) in the value of the optimal packing γ^* and the value of the optimal packing γ^*_{-s} when s is added to $S_{\text{non-participating}}$. The price paid to any losing station is zero. The $|S_{\text{winners}}|$ pricing problems can be solved in parallel once the initial allocation has been found.

VCG Simulations

We were unable to compute the VCG allocation with all of the eligible stations even after several days of computing time. To make the problem tractable, we restricted ourselves in the following manner: We dropped all of the Canadian stations and restricted ourselves to the UHF band using the smallest possible clearing target, 126 MHz, corresponding to a maximum allowable of channel of 29. Using the interference graph induced by these restrictions, we furthermore dropped every station that was not reachable by following at most two interference constraints from stations in New York City. We chose New York City as it is one of the most densely connected areas of the interference graph. The induced constraint graph contains 218 stations and is shown in Figure 5.4.

We computed the optimal repackings and pricing problems using CPLEX 12.6.2, with 8 cores available, solving all MIPs optimally to within 10^{-6} absolute MIP gap tolerance. Since the VCG experiments were computationally very expensive (it

[7] We implicitly restrict $D(s)$ to the home band for each non-participating station.

took a mean of 93.92 days CPU time to compute the allocation and prices for each value profile), we only ran them for our first five value profiles. We then ran reverse auction simulations in the same New York setting using the following feasibility checkers: SATFC 2.3.1, the greedy feasibility checker, and the best single configuration. Figure 5.5 plots the cost and value loss normalized by the corresponding VCG cost and value loss for each of these simulations. The SATFC 2.3.1 simulations had a mean value loss ratio of 1.047, so the reverse auction achieved outcomes with value losses very near to optimal in this setting and at lower cost than the VCG prices. Both the SATFC 2.3.1 and best single configuration runs Pareto dominated the greedy runs. Averaging across all results, greedy runs cost 1.732 times more and had value loss ratios 1.424 times higher than the SATFC 2.3.1 runs. SATFC 2.3.1 is a strictly more powerful feasibility checker than the best single configuration because SATFC 2.3.1 contains the best single configuration as part of its portfolio. The performance of SATFC 2.3.1 and the best single configuration were very similar in this setting, though we observed in three cases that the best single configuration actually led to a slightly higher value loss ratio than using SATFC 2.3.1. These results show that reverse auction performance does not always improve monotonically with a better feasibility checker.

5.5.3 National Simulation Results

While we could not simulate VCG at the national scale, we were still able to compare the cost and value loss resulting from substituting different feasibility checkers into our national simulations. Using the same value profiles as for our original national simulations, we ran new two new sets of simulations replacing SATFC 2.3.1 with the greedy feasibility checker and the best single configuration. We observed how the cost and efficiency of each auction varied with the new feasibility checkers. The results are shown in Figure 5.6, which plots the value loss and cost of each simulation normalized by the corresponding SATFC 2.3.1 simulation. Similarly to the VCG results, we observed that each SATFC 2.3.1 simulation and best single configuration simulation Pareto dominated their greedy counterparts. Averaging over all of our observations, the greedy simulations cost 3.550 times ($5.114 billion) more and had value losses 2.850 times ($2.030 billion) higher than the SATFC 2.3.1 sim-

Figure 5.4: Interference graph of the set of 218 UHF stations within two edges of a New York station. Each edge represents the existence of at least one pairwise binary constraint between two stations under a 126 MHz clearing target.

ulations. The gaps between SATFC 2.3.1 and the greedy feasibility checker in both the cost and efficiency dimensions were significantly larger at the national scale than in the smaller New York setting. The best single configuration simulations had values losses on average 1.101 times ($110 million) higher and were 1.142 times ($284 million) more expensive than their SATFC counterparts. The SATFC 2.3.1 simulations Pareto dominated the best single configuration simulations in all but one sample, in which the best single configuration simulation had a value loss 0.05% ($569 thousand) smaller than its corresponding SATFC 2.3.1 simulation, but was 6.8% ($137 million) more expensive. This is quite different from how these two feasibility checkers compared in our VCG simulations, where the best single configuration performed more favorably. These results strengthen our intuition that while the efficiency and cost of a reverse auction may not improve monotonically with a better feasibility checker, in general we expect that improvements to the performance of the feasibility checker will improve both of these metrics. Another intuition, also validated by these results, is that in larger settings with harder repacking problems, the importance of a strong feasibility checker is amplified.

Finally, recall that Section 4.5 described how a feasibility checker could revisit

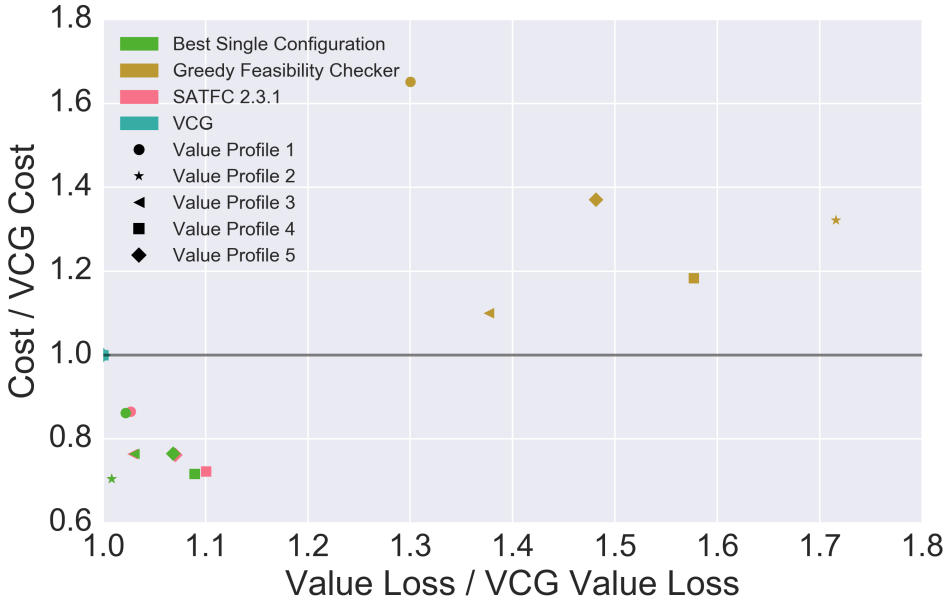


Figure 5.5: Fraction of VCG cost versus fraction of VCG value loss plotted for three different feasibility checkers (indicated by colors) for five different value profiles (indicated by markers). All VCG points lie at (1,1). SATFC 2.3.1 and the best single configuration had equivalent outcomes on the second value profile (the markers coincide).

timeouts within a bid processing round. We ran SATFC 2.3.1 simulations that revisited timeouts for each of the 20 simulations. These simulations took much longer than the runs that did not revisit timeouts, spending on average between 24.4 and 54.5 hours on solving UHF problems. We observed that this extra computation time on average led to better outcomes. The runs that did not revisit timeouts cost 1.011 times ($21 million) more and had value losses that were 1.010 times ($10 million) higher on average than the runs that did revisit timeouts.

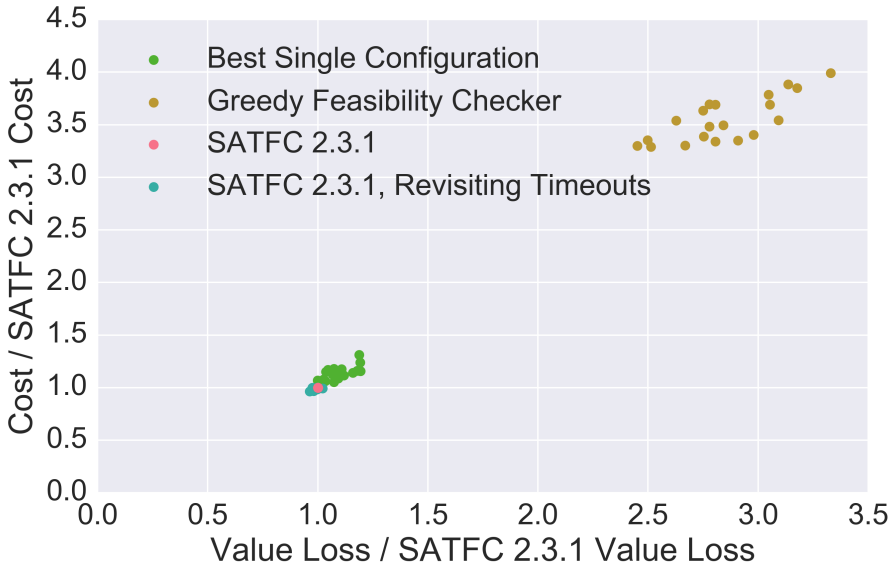


Figure 5.6: Value loss and cost of national simulations using three different feasibility checkers for 20 value profiles. Both axes are normalized by the cost and value loss of the corresponding SATFC 2.3.1 simulation. The figure also contains a second SATFC 2.3.1 series which revisit timeouts.

Chapter 6

Discussion and Conclusions

Station repacking in the reverse auction of the "incentive auction" is a difficult problem, but one in which progress is translatable into both large monetary sums and social welfare. We presented our feasibility checker, SATFC, which combines state-of-the-art SAT solvers, recent meta-algorithmic techniques, and further speedups based on domain-specific insights to yield a feasibility checker that meets the performance needs of the real incentive auction. We also presented our reverse auction simulator, which we used to explore the effect that the reverse auction's choice of feasibility checker has on the reverse auction's efficiency and cost. Specifically we showed that using SATFC vastly outperforms a very naïve feasibility checker, and that even substituting SATFC with a feasibility checker that is nearly as good as SATFC still results in significant losses in efficiency and increased cost. Pointers to source code and documentation for both SATFC and our reverse auction simulator are available at http://www.cs.ubc.ca/labs/beta/Projects/SATFC/.

There are many possible directions for future work. First, it would be interesting to continue to work towards a feasibility checker that can perform perfectly in this domain. This would involve further improvements to SATFC, such as adding new encodings and heuristics, or performing another round of configuration experiments. It would also be interesting to be able to compare the value loss from repackings generated by the reverse auction to the value loss of globally optimal repackings in larger settings (i.e., beyond just stations near New York City). This could be achieved by improving the performance of computing optimal repackings

(e.g., by applying algorithm configuration and new encodings to our VCG MIP) so that optimal repackings can be computed over larger sets of stations, or by bounding the value loss of optimal repackings if they cannot be computed exactly. There are other research questions that could be answered with some additions to our simulator: For example, we could enhance the valuation model to include values for VHF stations. This would allow for more realistic simulation results and could answer questions such as how much efficiency is gained by adding the complexity of the VHF bands. Other improvements that we could make to our simulator would be to enhance it to handle the initial clearing target optimization and the multi-stage nature of the incentive auction, which would allow us to answer questions such as whether the multi-stage nature of the auction decreases efficiency. We close by adding that even with the current simulator, there are many more interesting experimental questions that can be answered. For example, we can investigate alternative scoring rules, or better understand the tradeoffs involved in using fixed timeouts of one minute for each feasibility checking problem, as compared to longer timeouts or dynamic alternatives.

Bibliography

[1] K. I. Aardal, S. P. Van Hoesel, A. M. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007. → pages 5

[2] C. Bazelon, C. L. Jackson, and G. McHenry. An engineering and economic analysis of the prospects of reallocating radio spectrum from the broadcast band through the use of voluntary incentive auctions. TPRC, 2011. → pages 5

[3] M. Calamari, O. Kharkar, C. Kochard, J. Lindsay, B. Mulamba, and C. B. Scherer. Experimental evaluation of auction designs for spectrum allocation under interference constraints. In *Systems and Information Design Symposium*, pages 7–12. IEEE, 2012. → pages 4

[4] M. Charikar, P. Indyk, and R. Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *Automata, Languages and Programming*, pages 451–462. Springer, 2002. → pages 27

[5] Congressional Budget Office. Proceeds from auctions held by the Federal Communications Commission. https://www.cbo.gov/publication/50128, 2015. Accessed 2015-04-21. → pages 4

[6] P. Cramton, H. Lopez, D. Malec, and P. Sujarittanonta. Design of the reverse auction in the broadcast incentive auction. 2015. → pages 5

[7] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. → pages 17

[8] U. Doraszelski, K. Seim, M. Sinkinson, and P. Wang. Ownership concentration and strategic supply reduction. 2016. → pages 32, 38

[9] P. Dütting, V. Gkatzelis, and T. Roughgarden. The performance of deferred-acceptance auctions. In *Proc. of EC*, EC '14, pages 187–204, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2565-3. doi:10.1145/2600057.2602861. → pages 4

[10] FCC. In the matter of expanding the economic and innovation opportunities of spectrum through incentive auctions. *FCC Notice of Proposed Rulemaking*, FCC 12-118, September 2012. → pages 1

[11] FCC. Office of engineering and technology releases and seeks comment on updated OET-69 software. *FCC Public Notice*, DA 13-138, February 2013. → pages 14

[12] FCC. Comment sought on competitive bidding procedures for broadcast incentive auction 1000, including auctions 1001 and 1002. *FCC Public Notice*, 14-191, December 2014. → pages 6

[13] FCC. In the matter of expanding the economic and innovation opportunities of spectrum through incentive auctions. *FCC Report & Order*, FCC 14-50, June 2014. particularly section IIIB. → pages 15

[14] FCC. Repacking constraint files. http://data.fcc.gov/download/incentive-auctions/Constraint_Files/, 2015. Accessed 2015-11-20. → pages viii, 15, 17, 38

[15] FCC. Procedures for competitive bidding in auction 1000, including bidding in auction 1000, including initial clearing target determination, qualifying to bid, and bidding in auctions 1001 (reverse) and 1002 (forward). *FCC Public Notice*, FCC 15-78, August 2015. section III. → pages 33

[16] FCC. Reverse auction opening prices. http://wireless.fcc.gov/auctions/incentive-auctions/Reverse_Auction_Opening_Prices_111215.xlsx, 2015. Accessed 2015-11-15. → pages 37

[17] A. Fréchette, N. Newman, and K. Leyton-Brown. Solving the station repacking problem. 2016. → pages iii, 39, 40

[18] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer, 2007. → pages 29

[19] C. P. Gomes and B. Selman. Algorithm portfolios. *AIJ*, 126(1):43–62, 2001. → pages 25

[20] J. Hoffmann and J. Koehler. A new method to index and query sets. 1999. → pages 27

[21] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION*, page 507523, 2011. → pages 25

[22] F. Hutter, M. Lindauer, S. Bayless, H. Hoos, and K. Leyton-Brown. Configurable SAT solver challenge (CSSC) (2014). 2014. Accessed 2015-11-01. → pages 25

[23] F. Hutter, M. López-Ibáñez, C. Fawcett, M. Lindauer, H. H. Hoos, K. Leyton-Brown, and T. Stützle. AClib: A benchmark library for algorithm configuration. In *LION*, pages 36–40. Springer, 2014. → pages 29, 39

[24] M. Järvisalo, D. Le Berre, O. Roussel, and L. Simon. The international SAT solver competitions. *AI Magazine*, 33(1):89–92, 2012. → pages 19

[25] E. Kazumori. Generalizing deferred acceptance auctions to allow multiple relinquishment options. *SIGMETRICS Performance Evaluation Review*, 42 (3):41–41, 2014. → pages 4

[26] M. Kearns and L. Dworkin. A computational study of feasible repackings in the FCC incentive auctions. *CoRR*, abs/1406.4837, 2014. URL http://arxiv.org/abs/1406.4837. Accessed 2016-05-09. → pages 5

[27] A. R. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown. Satenstein: Automatically building local search sat solvers from components. → pages 29

[28] R. Knutson and T. Gryta. Verizon, AT&T May Face Bidding Limits in Spectrum Auction. *Wall Street Journal*, Apr. 2014. ISSN 0099-9660. URL http://www.wsj.com/articles/SB10001424052702304626304579510154106120342. Accessed 2016-12-12. → pages 1

[29] E. Kwerel, P. LaFontaine, and M. Schwartz. Economics at the FCC, 2011–2012: Spectrum incentive auctions, universal service and intercarrier compensation reform, and mergers. *Review of Industrial Organization*, 41 (4):271–302, 2012. → pages 4

[30] S. Li. Obviously strategy-proof mechanisms. *Available at SSRN 2560028*, 2015. → pages 4

[31] C. Luo, S. Cai, W. Wu, and K. Su. Double configuration checking in stochastic local search for satisfiability. In *AAAI*, 2014. → pages 40

[32] A. K. Mackworth. Consistency in networks of relations. *Artificial intelligence*, 8(1):99–118, 1977. → pages 22

[33] M. J. Marcus. Incentive auction: a proposed mechanism to rebalance spectrum between broadcast television and mobile broadband [spectrum policy and regulatory issues]. *Wireless Communications*, 20(2):4–5, 2013. → pages 4

[34] P. Milgrom and I. Segal. Deferred-acceptance auctions and radio spectrum reallocation. In *Proc. of EC*. ACM, 2014. → pages 2, 4

[35] P. Milgrom, L. Ausubel, J. Levin, and I. Segal. Incentive auction rules option and discussion. *Report for Federal Communications Commission. September*, 12, 2012. → pages 4

[36] T.-D. Nguyen and T. Sandholm. Optimizing prices in descending clock auctions. In *Proc. of EC*, pages 93–110. ACM, 2014. → pages 5

[37] E. Nudelman, K. Leyton-Brown, G. Andrew, C. Gomes, J. McFadden, B. Selman, and Y. Shoham. Satzilla 0.9. Solver description, International SAT Competition, 2003. → pages 25

[38] S. Prestwich. Local search on sat-encoded colouring problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 105–119. Springer, 2003. → pages 19

[39] I. Savnik. Index data structure for fast subset and superset queries. In *Availability, Reliability, and Security in Information Systems and HCI*, pages 134–148. Springer, 2013. → pages 27

[40] A. Vohra. On the near-optimality of the reverse deferred acceptance algorithm. 2014. → pages 4

[41] L. Xu, H. H. Hoos, and K. Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *AAAI*, pages 210–216, 2010. → pages 25