Solving the Station Repacking Problem

Alexandre Fréchette, Neil Newman, Kevin Leyton-Brown

Agenda

- Background
- Problem
- Novel Approach
- Experimental Results

Background

A Brief History

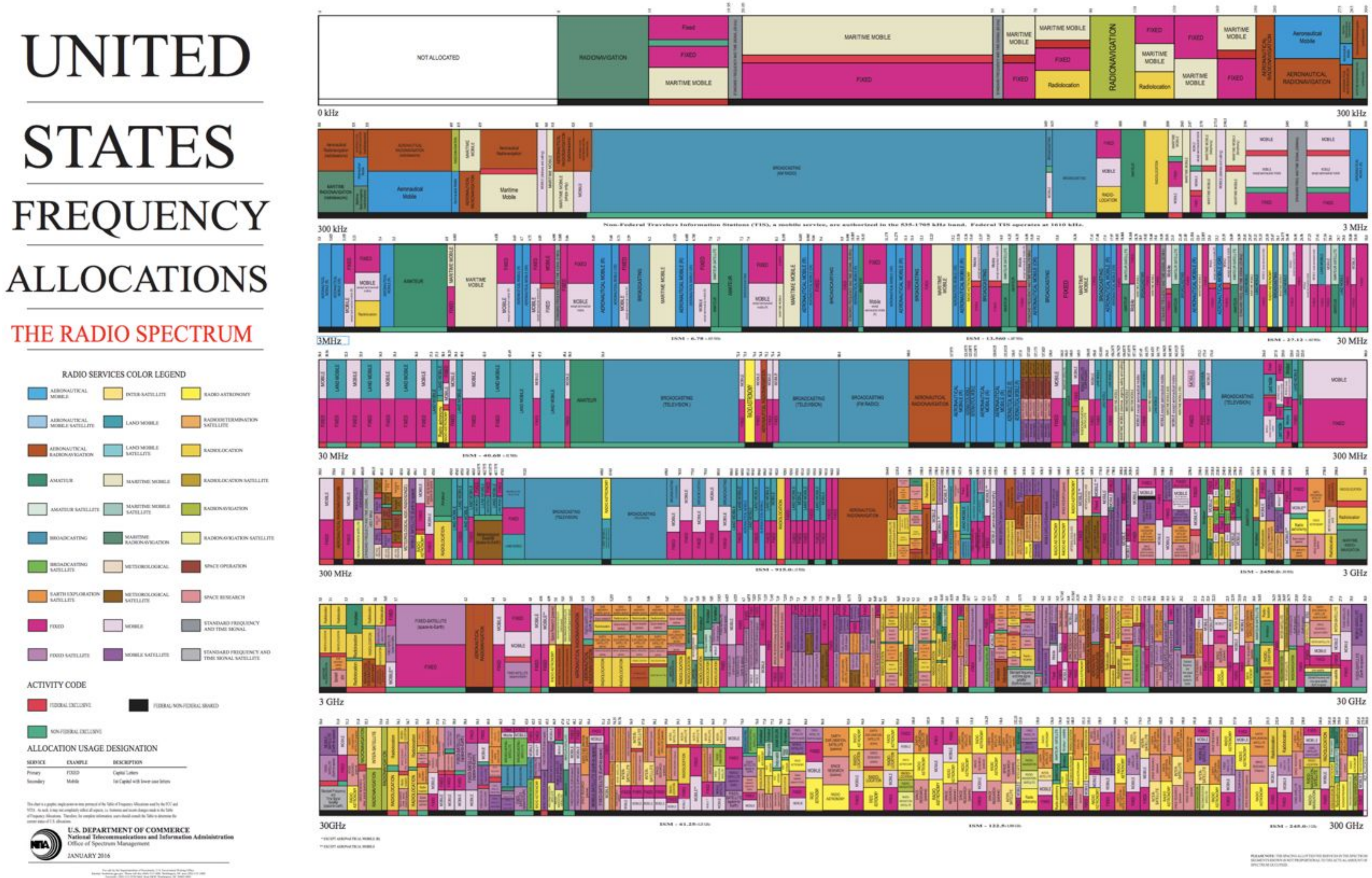
Spectrum rights have historically been a mess. Licenses were given away after public or private hearings.

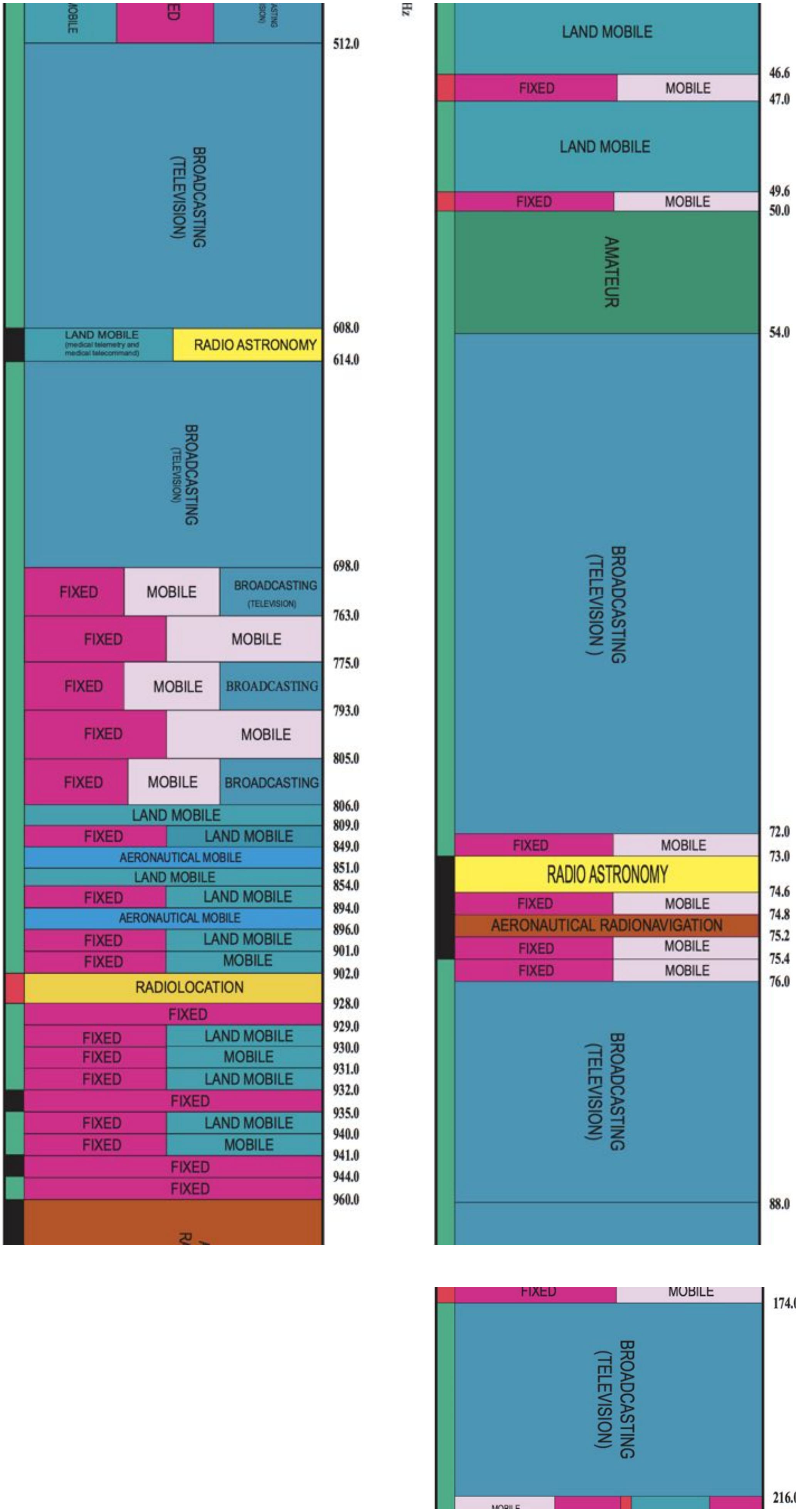
Then, the FCC pioneered selling spectrum through auctions.

Broadcast TV viewership (demand) has declined over the years.

Mobile demand for spectrum has increased.

It would be great to “clear” TV spectrum for mobile use.


UNITED
STATES
FREQUENCY
ALLOCATIONS
THE RADIO SPECTRUM
RADIO SERVICES COLOR LEGEND
AERONAUTICAL MOBILE
AERONAUTICAL MOBILE SATELLITE
AERONAUTICAL RADIONAVIGATION
AMATEUR
AMATEUR SATELLITE
BROADCASTING
BROADCASTING SATELLITE
EARTH EXPLORATION SATELLITE
FIXED
FIXED SATELLITE
INTER-SATELLITE
LAND MOBILE
LAND MOBILE SATELLITE
MARITIME MOBILE
MARITIME MOBILE SATELLITE
MARITIME RADIONAVIGATION
METEOROLOGICAL
METEOROLOGICAL SATELLITE
MOBILE
MOBILE SATELLITE
RADIO ASTRONOMY
RADIODETERMINATION SATELLITE
RADIOLOCATION
RADIOLOCATION SATELLITE
RADIONAVIGATION
RADIONAVIGATION SATELLITE
SPACE OPERATION
SPACE RESEARCH
STANDARD FREQUENCY AND TIME SIGNAL
STANDARD FREQUENCY AND TIME SIGNAL SATELLITE
ACTIVITY CODE
FEDERAL EXCLUSIVE
FEDERAL/NON-FEDERAL SHARED
NON-FEDERAL EXCLUSIVE
ALLOCATION USAGE DESIGNATION
SERVICE
EXAMPLE
DESCRIPTION
Primary
FIXED
Capital Letters
Secondary
Mobile
1st Capital with lower case letters
U.S. DEPARTMENT OF COMMERCE
National Telecommunications and Information Administration
Office of Spectrum Management
JANUARY 2016
NOT ALLOCATED
RADIONAVIGATION
MARITIME MOBILE
FIXED
0 kHz
300 kHz
BROADCASTING (AM RADIO)
Non-Federal Travelers Information Stations (TIS), a mobile service, are authorized in the 535-1705 kHz band. Federal TIS operates at 1610 kHz.
3 MHz
AMATEUR
BROADCASTING
3MHz
30 MHz
BROADCASTING (TELEVISION)
RADIO ASTRONOMY
BROADCASTING (FM RADIO)
AERONAUTICAL RADIONAVIGATION
300 MHz
3 GHz
FIXED-SATELLITE (space-to-Earth)
30 GHz
30GHz
300 GHz



512.0
BROADCASTING (TELEVISION)
608.0
LAND MOBILE (medical telemetry and medical telecommand)
RADIO ASTRONOMY
614.0
BROADCASTING (TELEVISION)
698.0
FIXED
MOBILE
BROADCASTING (TELEVISION)
763.0
FIXED
MOBILE
775.0
FIXED
MOBILE
BROADCASTING
793.0
FIXED
MOBILE
805.0
FIXED
MOBILE
BROADCASTING
806.0
LAND MOBILE
809.0
FIXED
LAND MOBILE
849.0
AERONAUTICAL MOBILE
851.0
LAND MOBILE
854.0
FIXED
LAND MOBILE
894.0
AERONAUTICAL MOBILE
896.0
FIXED
LAND MOBILE
901.0
FIXED
MOBILE
902.0
RADIOLOCATION
928.0
FIXED
929.0
FIXED
LAND MOBILE
930.0
FIXED
MOBILE
931.0
FIXED
LAND MOBILE
932.0
FIXED
935.0
FIXED
LAND MOBILE
940.0
FIXED
MOBILE
941.0
FIXED
944.0
FIXED
960.0
Hz
LAND MOBILE
46.6
FIXED
MOBILE
47.0
LAND MOBILE
49.6
FIXED
MOBILE
50.0
AMATEUR
54.0
BROADCASTING (TELEVISION)
72.0
FIXED
MOBILE
73.0
RADIO ASTRONOMY
74.6
FIXED
MOBILE
74.8
AERONAUTICAL RADIONAVIGATION
75.2
FIXED
MOBILE
75.4
FIXED
MOBILE
76.0
BROADCASTING (TELEVISION)
88.0
FIXED
MOBILE
174.0
BROADCASTING (TELEVISION)
216.0

Spectrum Incentive Auction

Goal: Free up contiguous spectrum for mobile use

Let's buy back the UHF radio spectrum and then sell it to the mobile companies.

Stations can...

1. Take our money and close up shop
2. Take a portion of our offer to voluntarily move down the spectrum
3. Not participate but may be forced to move down

This will be through a reverse auction followed by a forward auction.

Process cancelled if government can't break even or make money.

Reverse Auction

Multiple round descending price countdown auction.

Initial offer depends on local competition, national clearing target, etc. Prices should motivate stations to sell.

Stations are considered in a round-robin style during the auction.

Price/offer for a given station will decrease each round assuming they can be “repacked” at a lower frequency.

Forward Auction

Step 1: Sell spectrum to mobile companies

Step 2: Profit

Problem Definition

Problem

We need to be able to determine if it is feasible to move (repack) a channel during the reverse auction.

Stations can only use certain channels.

Stations cannot interfere with one another.

We will be given hundreds of thousands of repacking problems throughout the auction.

This is a NP-Complete problem we will need to solve quickly during the auction.

Standard solutions like MIP are too slow.

Considerations

What is involved in this repacking problem?

Domain assignments

Interference constraints

Performance

Domain Constraints

Not all stations can use all channels.

A domain file is provided which lists the possible channels each station could be assigned in the repacking process.

Station ID

Available Channels

```
DOMAIN, 10001, 2, 3, 4, 5, 6, 19, 20, 21, 48, 49, 50, 51
DOMAIN, 10002, 2, 3, 4, 5, 6, 7, 8, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40
DOMAIN, 10003, 2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 34, 35, 40, 41, 42, 43, 44, 45
```

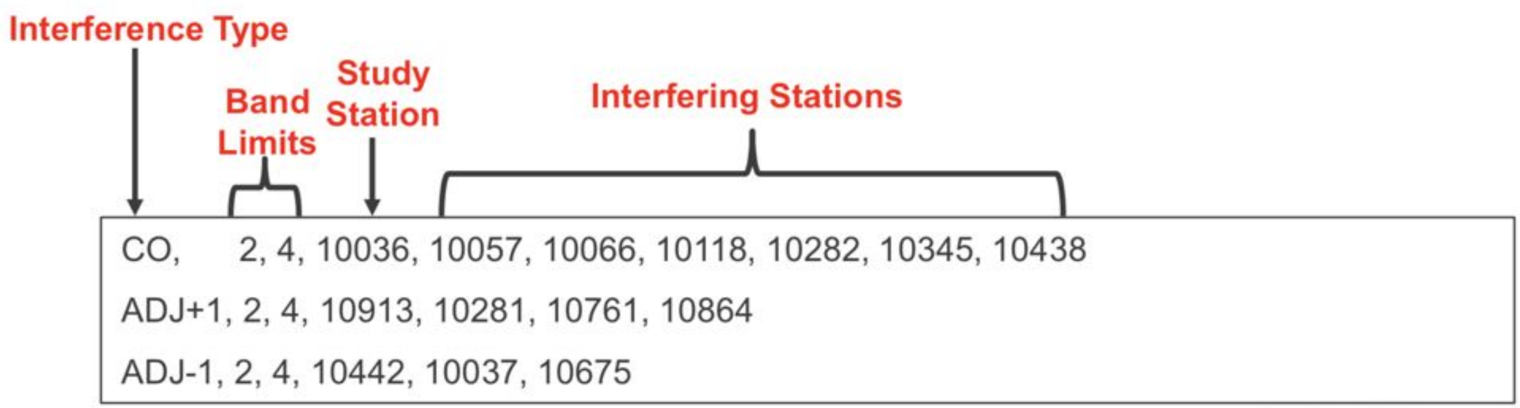
Interference Constraints

We must make sure repacking does not introduce interference.

Co-channel constraints: 2+ stations cannot be assigned the same channel

Adjacent channel constraints: Specific stations cannot be assigned adjacent channels

An interference file is provided which enumerate these constraints.

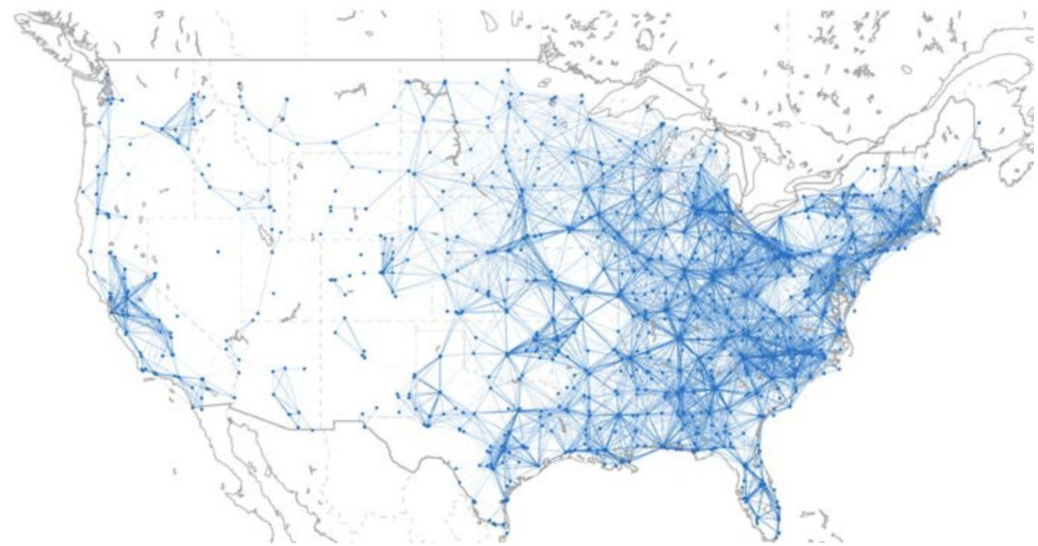


Interference Graph

The undirected graph of interference constraints.

Roughly 2000 channels.

Because of adjacency constraints, we cannot solve as a graph coloring problem.



Performance

We need to solve these repacking problems quickly...

So auction designers and economists can experiment and study auction behavior.

If we can’t solve a particular station in time we cannot lower the bid which could leave money on the table.

The auction is expected to have several rounds per day and take weeks overall.

Novel Approach

Approach: SATFC 2.0

SAT encoding and SAT solvers

Algorithm Configuration using SMAC

Algorithm Portfolio

Incremental Station Repacking

Problem Simplification

Hydra technique (AC + AP)

Containment Cache

SAT Encoding

Encode as a propositional satisfiability problem

Encode as Satisfiability Problem

Repacking is well suited as a feasibility problem with combinatorial constraints.

Leverage open-source, high performance solvers.

S = {all stations}

C = {all channels}

D = {domain allowable station/channel mappings}

I = {invalid station/channel mappings due to co-channel or co-adjacency}

Basic form look like $\neg x_{s,c} \vee \neg x_{s',c'}$

SAT Encoding Clauses

1. Each station is assigned at least one channel

$$\bigvee_{d \in D(s)} x_{s,d} \; \forall s \in S$$

2. Each station is assigned at most one channel

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

3. Interference constraints are respected

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

SAT Encoding

- $x_{s,c}$: the proposition that **station s is assigned to channel c**
  - one such variable for every station s and channel c
- Station s **must broadcast on one** of its allowable channels
  - For every station s and set of allowable channels $\{c_1, \dots, c_n\}$, create a clause $(x_{s,c_1} \vee \cdots \vee x_{s,c_n})$
- Station s **may broadcast on at most one** of these channels
  - For every pair of channels c_1 and c_2 allowed for station s, create a clause $\left(\neg x_{s,c_1} \vee \neg x_{s,c_2}\right)$
- The repacking **does not cause harmful interference**
  - For every interference rule stating that s_1 cannot broadcast on c_1 while s_2 broadcasts on c_2, create a clause $(\neg x_{s_1,c_1} \vee \neg x_{s_2,c_2})$
- Note: mostly 2-clauses
  - good for unit propagation: implies clique constraints

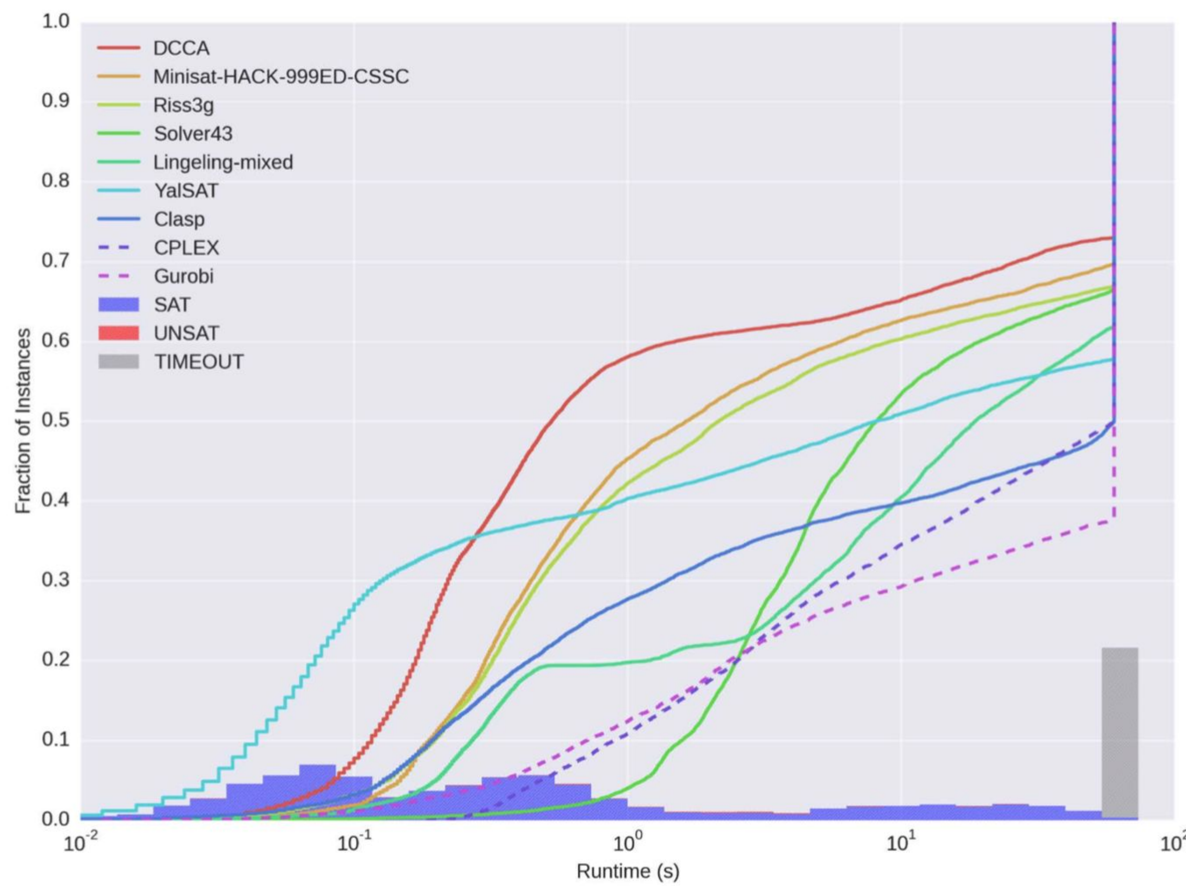
https://simons.berkeley.edu/talks/kevin-leyton-brown-2015-11-19

SAT Solvers

Outperformed initial mixed integer program solvers (CPLEX, Gurobi).

18 SAT solvers initially evaluated, none were able to solve all problems in target time of 60 seconds.

Most problems are actually solvable (~99%) but not before timeout.



Algorithm Configuration

Use machine learning to find optimal algorithm parameters

Algorithm Configuration

Some solvers like CLASP have lots of parameters allowing for fine-tuning.

We can view this as an optimization problem and use an automated approach to find the best parameters for our problem.

Leyton-Brown’s research group previously developed software for algorithm configuration: Sequential Model-based Algorithm Configuration (SMAC)

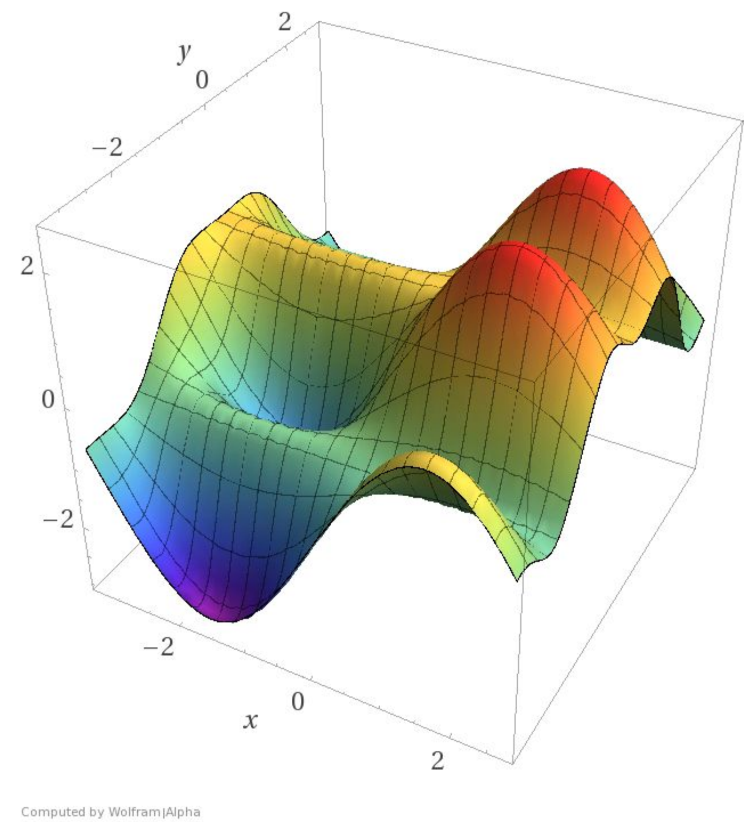
SMAC

Step 1: Configure SMAC with the solver and parameters.

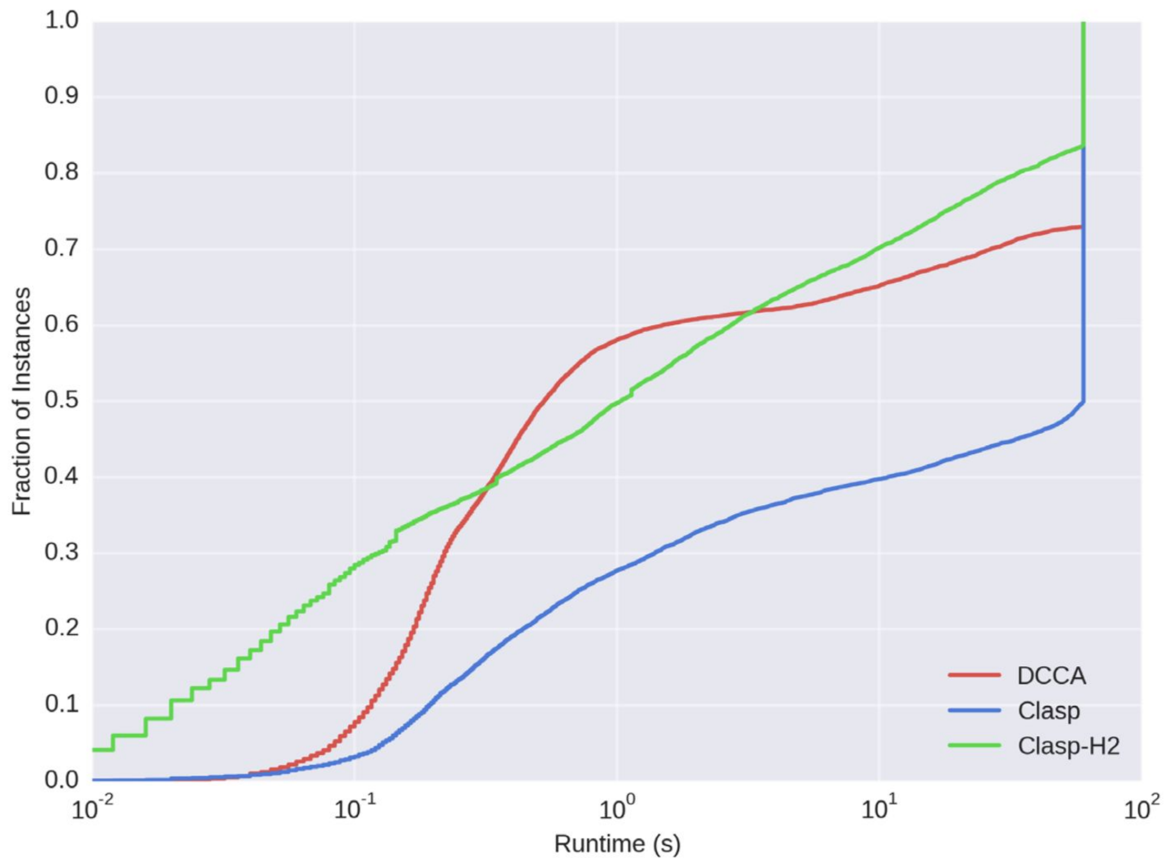
Step 2: Configure run period (ex: one day) for SMAC to find best parameter values.

SMAC will build a response surface starting with random(ish) initial parameters and hone in on the best values for a given problem.

(uses random forest regression trees)



Sample Algorithm Configuration Results



Algorithm Portfolio

#teamwork

Algorithm Portfolio

Rarely can we find a single solver to resolve all problems for an NP-Hard problem.

Instead, select a set of complementary solvers.

Attack the problem in parallel with the solvers.

This is also an active area of study for Leyton-Brown's research group.

Incremental Repacking

leveraging current assignments

Local Augmenting

Starting Assignment for Local Search Solvers

Local Augmenting

When checking feasibility of repacking a station, only consider the station and its immediate neighbors.

Hold all stations outside of this neighborhood fixed.

If we can quickly determine the local repack is feasible than we are done.

A modified *DCCA-preSAT* improved over DCCA by solving 78.5% of test instances in .1 seconds before stagnating.

Starting Assignment for Local Search Solvers

“Local search solvers such as DCCA 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.”

Similar to Local Augmenting, we can start with current assignments for a repack and then give s^+ a random channel to start with.

This approach does not constrain the problem to the neighborhood of s^+.

A modified *DCCA+* improved over DCCA by solving 85.4% of the sample problems before the timeout.

Problem Simplification

Making smaller problems out of bigger problems

Graph decomposition

Station Removal

Graph Decomposition

A set of related stations will usually results in a disconnected subgraph of interference constraints.

We can often break a problem down into multiple subgraphs / components.

Each subgraph is a computationally easier problems to solve.

If we can prove one of the smaller problems is infeasible then the whole problem is infeasible.

The largest component is often significantly smaller than the original problem.

Underconstrained Station Removal

There are some stations that can always be repacked due to less local competition for channels.

Removing these stations from the original feasibility problem makes it easier to compute.

This also improves graph decomposition.

Hydra

Iteratively build our portfolio:
Algorithm Configuration +
Algorithm Portfolio

"Which solver will offer the greatest marginal contribution to the existing portfolio?"

Hydra

Problem simplification lowers correlation between solvers making them “more different”.

SATzilla is an algorithm portfolio builder that iteratively adds a solver/algorithm that adds the most value.

This is, of course, an active area of study by Leyton-Brown’s research group and their SATzilla software has won numerous SAT competitions.

http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/

Containment Caching

Feasible cache

Infeasible cache

Fast cache queries

Subsets and Supersets

We know all of the constraints ahead of time.

We have lots of time to prepare.

But pre-cached problem solutions were found to RARELY be directly applicable for new problems.

However… if a set S is packable, then every subset $S' \subseteq S$ is also packable (and we have the packing)

Similarly, if set S is NOT packable, then every superset $S' \supseteq S$ is also NOT packable

We can build caches which tell us wither one set contains another.

Let's Build Caches

We will build feasible and infeasible caches for each problem we solve.

When faced with a new problem to repack station set S...

Check whether the feasible cache contains a superset of S. It's feasible!

Check whether the infeasible cache contains a subset of S. It's infeasible!

Else, simplify and decompose the problem. Check to see if each component can be found in the feasibility cache.

This becomes a cache querying problem.

Primary (traditional) Caches

Contains a full solution mapping (if exists) for a given problem along with the problem instance and simplified components.

Indexed by a hash function.

Not useful to answer feasibility question directly - see secondary caches.

Secondary Caches

Contain lists of station sets that correspond to entries in the primary cache.

We use these for querying and then “hash” into the primary cache when needed.

Each station set is represented by a bit string {1101..000101101} which can be interpreted as a large integer.

Very compact/efficient: “a cache of 200,000 entries, each consisting of 2,000 stations/bits, occupies only 50 MB”

We can have ℓ multiple secondary caches (descending order by integer value) with different random bit orders to search over.

Superset Cache Querying

Given a query S, we perform binary search on each of the ℓ secondary caches to find the primary cache index corresponding to S (if it is in the cache) or of the smallest entry larger than S (if not in cache)

If we find S, that is a direct hit on a solution.

If we don't find S, but we find a superset (larger entry), then we know the repack is feasible as part of a larger feasible repack.

Testing showed query execution within an average time of 30 ms on a cache of nearly 200,000 entries.

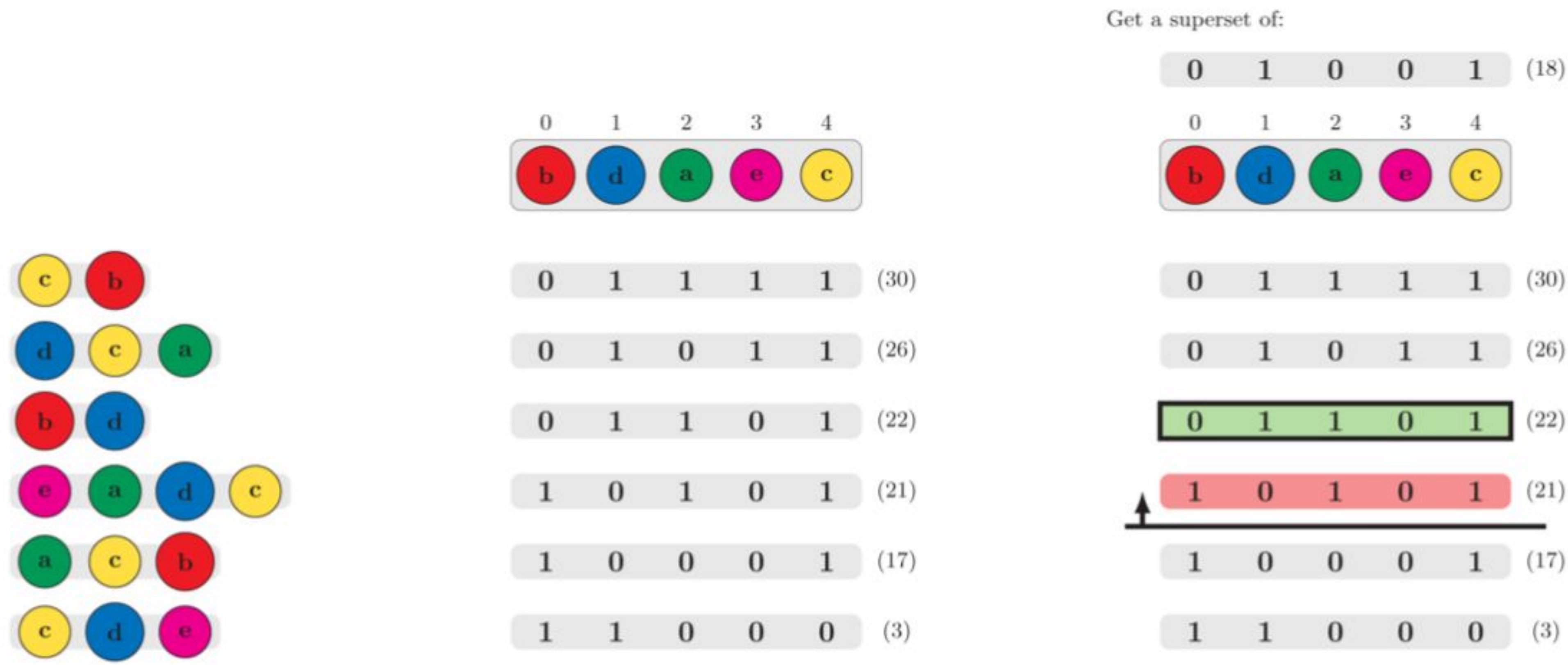


Figure 3: Containment caching example. Left: six elements of the power set $2^{\{a,b,c,d,e\}}$. Center: a secondary cache defined by a random ordering over the five elements, with each of the sets interpreted as a bit string and sorted in descending order. Right: the result of querying the containment cache for supersets of $\{c, d\}$. The query (18) does not exist in the cache directly; the next largest entry (21) is not a superset (i.e., 01001 does not bitwise logically imply 10101); the cache returns $\{a, c, d\}$ (22).

Containment Cache Evaluation

Used a 4 solver portfolio on all FCC supplied instances for 24 hours.

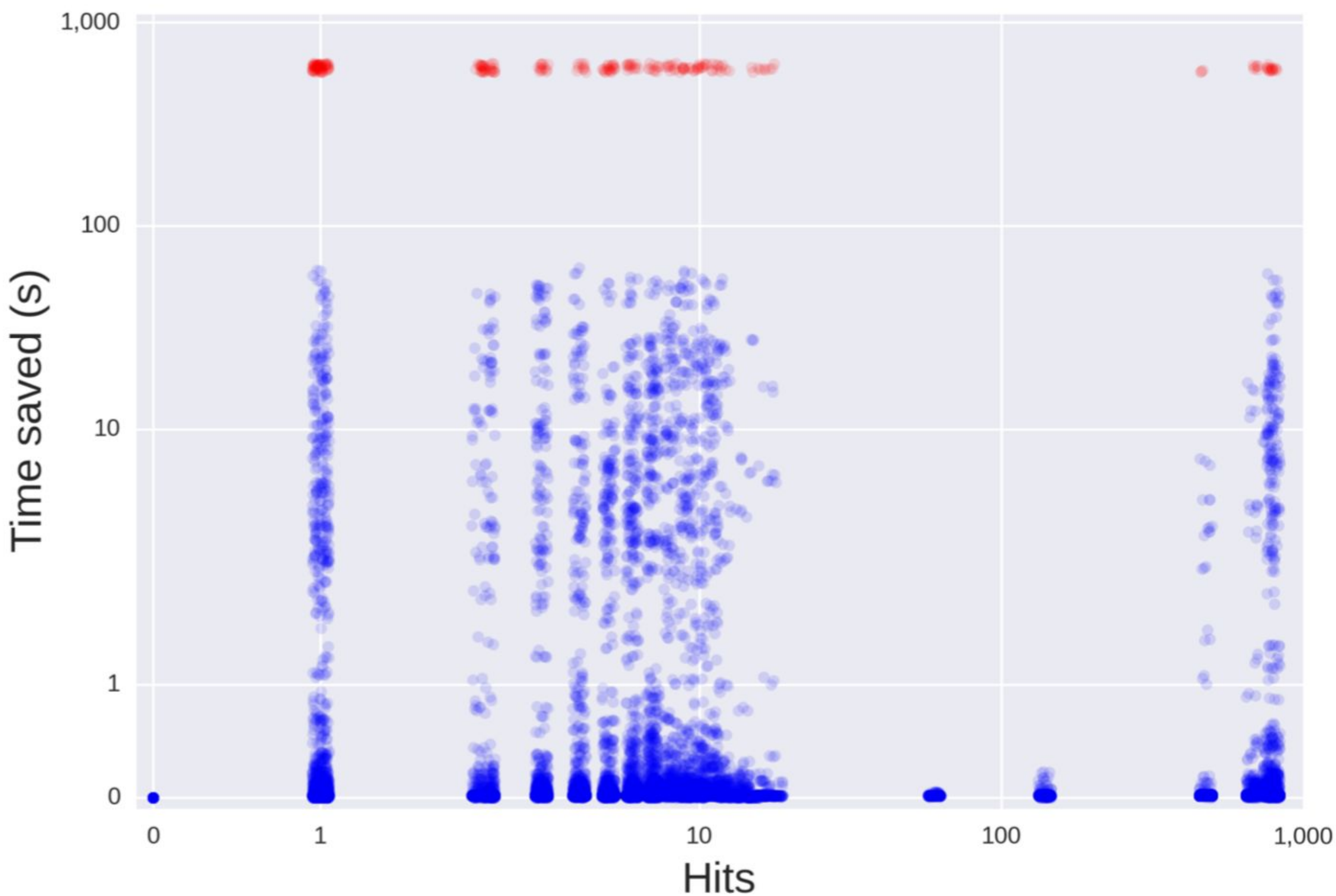
Solvers used the cache for lookups as they also built it up.

Afterwards, had a cache of 185,750 entries.

Largest problem in feasible cache had 1170 stations while smallest problem in infeasible cache had 2 stations. (remember superset vs subset lookups).

They built 5 secondary caches each with different bit orderings ($\ell = 5$).

When viewed as a “solver” it outperformed all other algorithms, solving 98.2% of problems.


Time saved (s)
1,000
100
10
1
0
0
1
10
100
1,000
Hits

Experimental Results

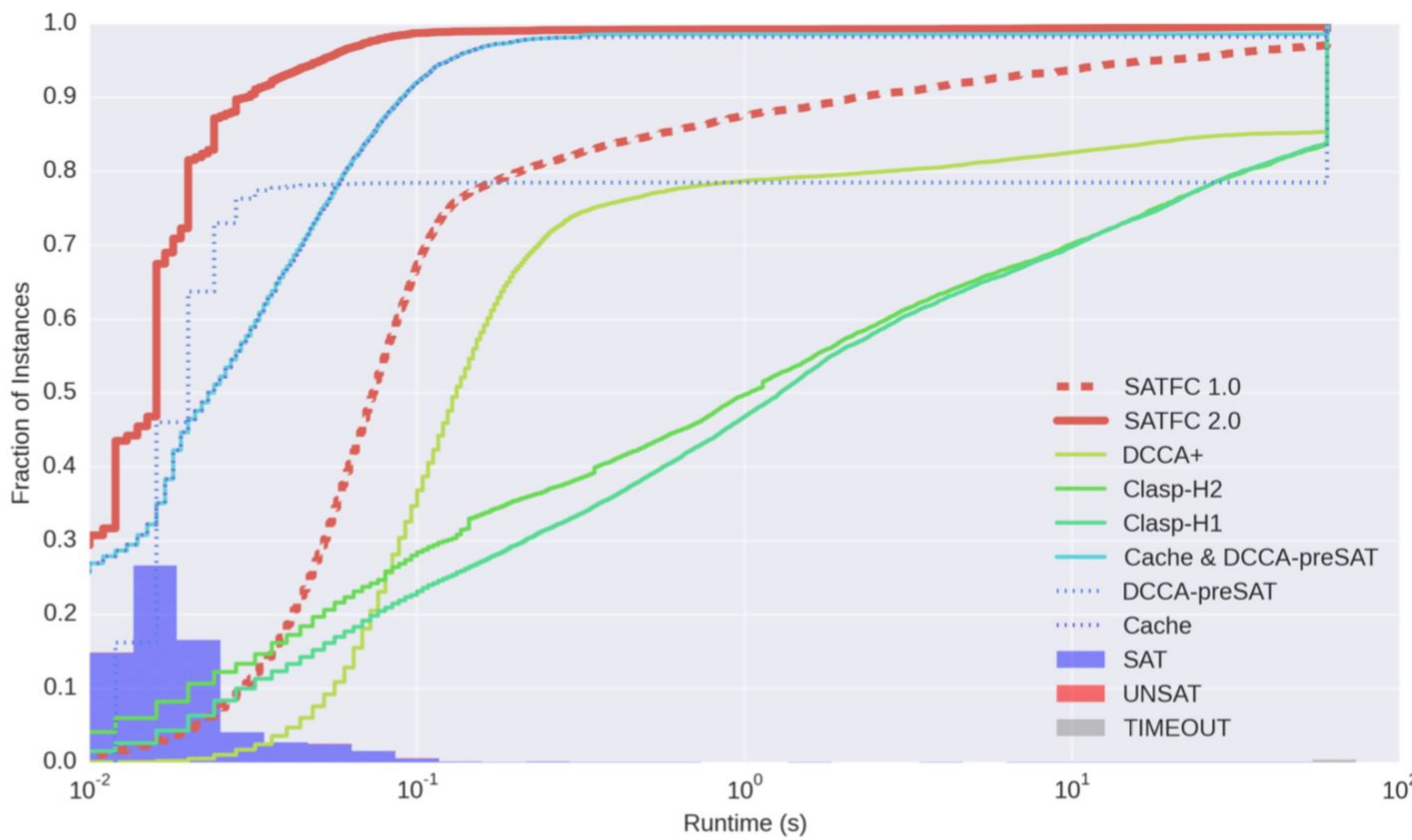
Results

This research produced a 4 solver portfolio plus the containment cache for addressing the repacking problem named SATFC 2.0.

Solvers: DCCA-preSAT, DCCA+, clasp-h1, and clasp-h2

In evaluation, this solution was able to solve 99.0% of test instances in under 0.2 seconds, and 99.6% in under a minute.

Final results on test data



Questions?