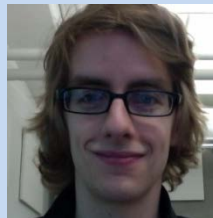


Spectrum Repacking in the Incentive Auction



Kevin Leyton-Brown
Computer Science Department
University of British Columbia

Joint work with Alexandre Fréchette and Neil Newman



FCC's "Incentive Auction"

Federal Communications Commission

Display Options

The FCC Our Work Tools & Data Business & Licensing Bureaus & Offices

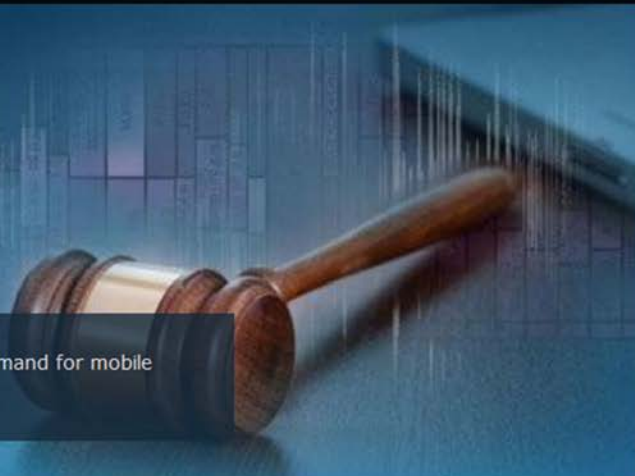
FC

Search

Take Action Comment, Complain, Discuss

Transition.FCC.gov

Home / Our Work / Incentive Auctions



Incentive Auctions

Unleashing spectrum to meet America's demand for mobile broadband

Find out about business opportunities and the Incentive Auctions process

'LEARN' PROGRAM

The United States leads the world in key areas of wireless infrastructure and innovation, including being the first country to have 4G Long-Term Evolution (LTE) technology networks at scale and to enable unlicensed use of white space spectrum. Meanwhile, demands on both licensed and unlicensed spectrum are increasing dramatically.

Explore the Broadcast Television Spectrum Incentive Auction Rulemaking

Quick Links

- Learning Everything About Reverse-Auctions Now (LEARN)
- Report and Order Staff Summary
- Incentive Auctions NPRM
- Incentive Auctions Resources

Anticipated Revenue: Tens of Billions



CONGRESSIONAL BUDGET OFFICE
U.S. Congress
Washington, DC 20515

Keith Hall, Director

Honorable Dean Heller
United States Senate
Washington, DC 20510

April 21, 2015

Dear Senator:

This letter responds to several questions that you posed about proceeds from certain auctions that have been held or will be held by the Federal Communications Commission (FCC).

To briefly summarize some of the major points mentioned below:

- The FCC is planning to hold a two-sided auction—known as an incentive auction—that will provide an opportunity for television broadcasters to voluntarily sell their spectrum rights and for wireless firms to buy licenses to use those frequencies.
- Because the FCC has not conducted such an auction before, it is difficult to predict what the net proceeds of the auction will be. The Congressional Budget Office estimates that the net proceeds will probably be between \$10 billion and \$40 billion, with an expected value of \$25 billion, the middle of that range.

Auction Tentatively Set for Next Year

Topics: [Regulatory News](#) | [Spectrum News](#)

FCC tentatively sets March 29, 2016, as start date for 600 MHz incentive auction

July 20, 2015 | By [Phil Goldstein](#)

SHARE

Email

28

Tweet

3

Share

0

Like

1

g+1

The FCC currently plans to start next year's incentive auction of 600 MHz broadcast TV spectrum on March 29, 2016, according to an FCC website. That would keep the agency just on target to meet FCC Chairman Tom Wheeler's goal of starting the complex auction in the first quarter of next year, which ends March 31.

Last week the FCC postponed its vote on rules for the incentive auction until Aug. 6 after pressure from Congress to push back the vote that had been planned for the agency's July 16 meeting. The FCC is going to allow more time for stakeholders, including broadcasters and carriers, to review information to the FCC and submit filings on the rules.

Accordingly, the FCC moved that agenda item onto its list of items that are currently "on circulation," meaning that they are being reviewed by the FCC's five commissioners before being voted on before the full commission. The item is titled: "Broadcast Incentive Auction to Begin March 29, 2016; Procedures for Competitive Bidding in Auction 1000, Including Initial Clearing Target Determination, Qualifying to Bid, and Bidding in Auctions 1001 (Reverse) and 1002 (Forward)."

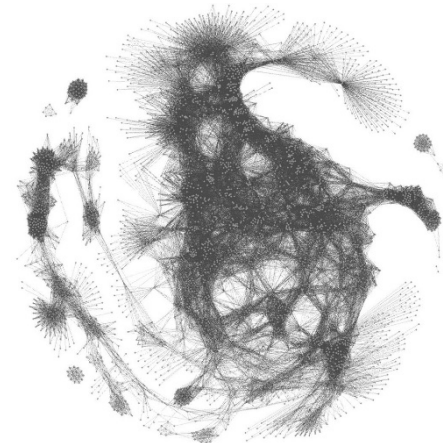
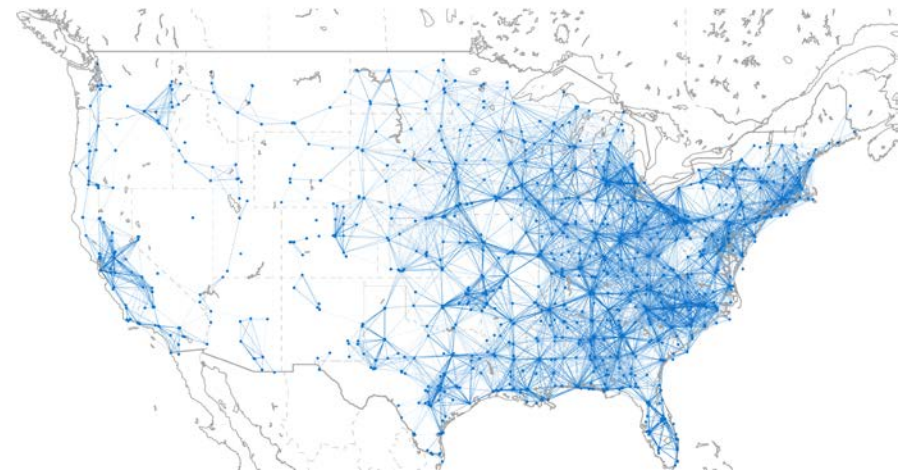
The FCC's "Incentive Auction"

- **Forward (ascending-price) auction** for telecom firms
 - prices in each region increase while demand exceeds supply
- **Reverse (descending-price) auction** for broadcasters
 - stations declare they're willing to stop broadcasting at a given, initially high, price
 - consider stations round robin:
 - check to see if the station could feasibly be "repacked" into the reduced band, given interference constraints
 - if not (or if we can't solve the problem), it's "frozen"
 - if so, offer it lower compensation, take-it-or-leave-it
- **When auctions terminate**, ensure revenue target is met
 - if not, grow the size of the reduced band (i.e., clear less spectrum); auctions continue

Feasibility Testing

Key computational problem: testing the feasibility of a given repacking, based on interference constraints

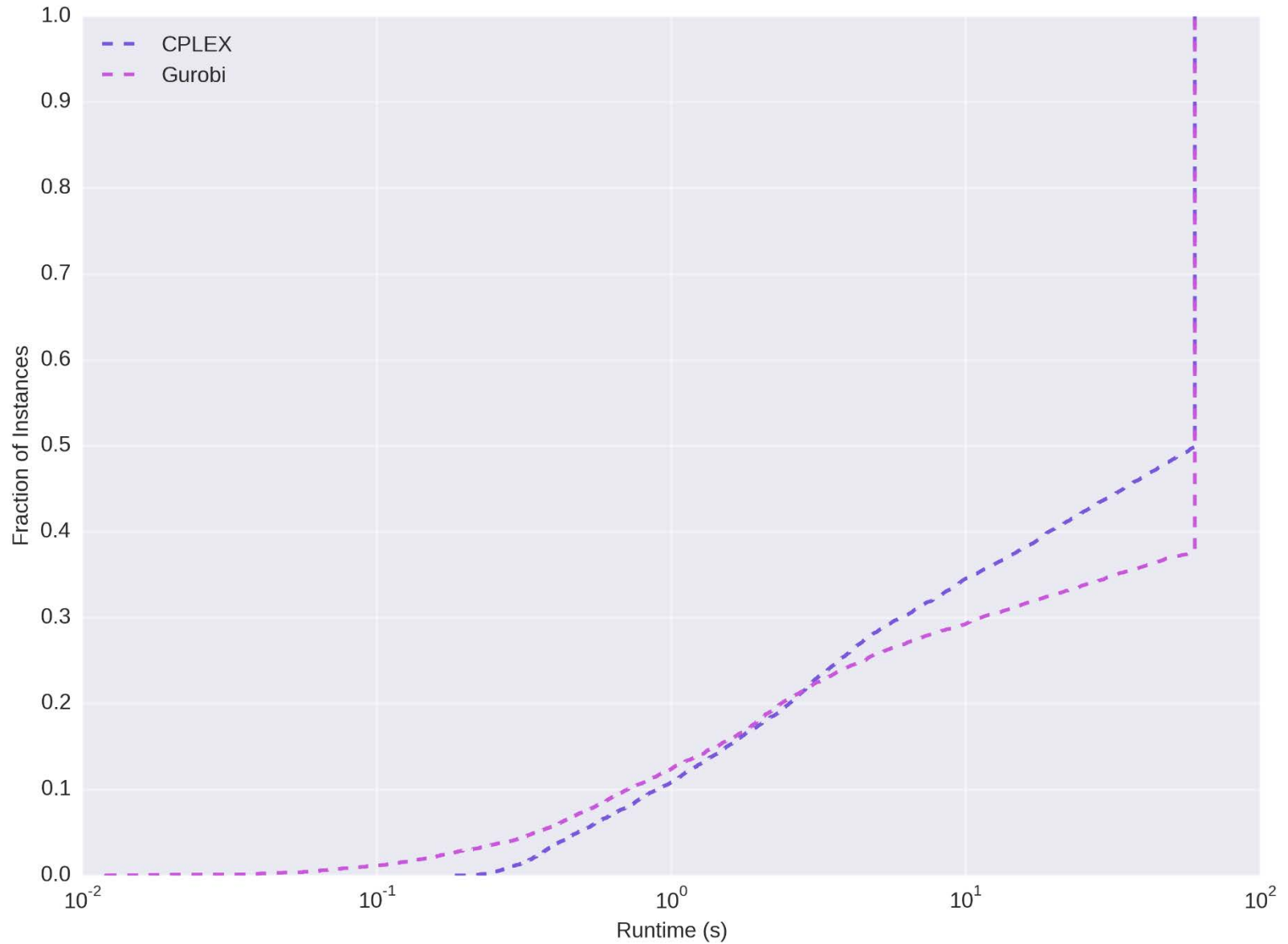
- Basis of “frozen test”: about 100K per auction; 20K are nontrivial
- A hard **graph-colouring** problem
 - 2991 stations (nodes)
 - 2.7 million interference constraints (channel-specific interference)
 - Initial skepticism about whether this problem could be solved exactly at a national scale
 - if not, say the problem is infeasible, and pay more than necessary
- We’re doing it, using tools from **empirical algorithmics**
 - automatic algorithm configuration
 - algorithm portfolios; Hydra
 - domain-specific heuristics
 - a powerful new caching scheme



Experimental Setup

- FCC's May 2014 **interference data**
 - clearing target: 19 UHF channels (14-32)
- **5 simulated auctions**
 - based on the “smoothed ladder” auction mechanism and a (proprietary) valuation model
 - UHF problems only; VHF was too easy
 - randomly partitioned into 3 sets: training (100,572 instances); validation (1,000 instances); test (10,000 instances)
- Our goal: solve as many problems as we can in within a **one-minute cutoff**

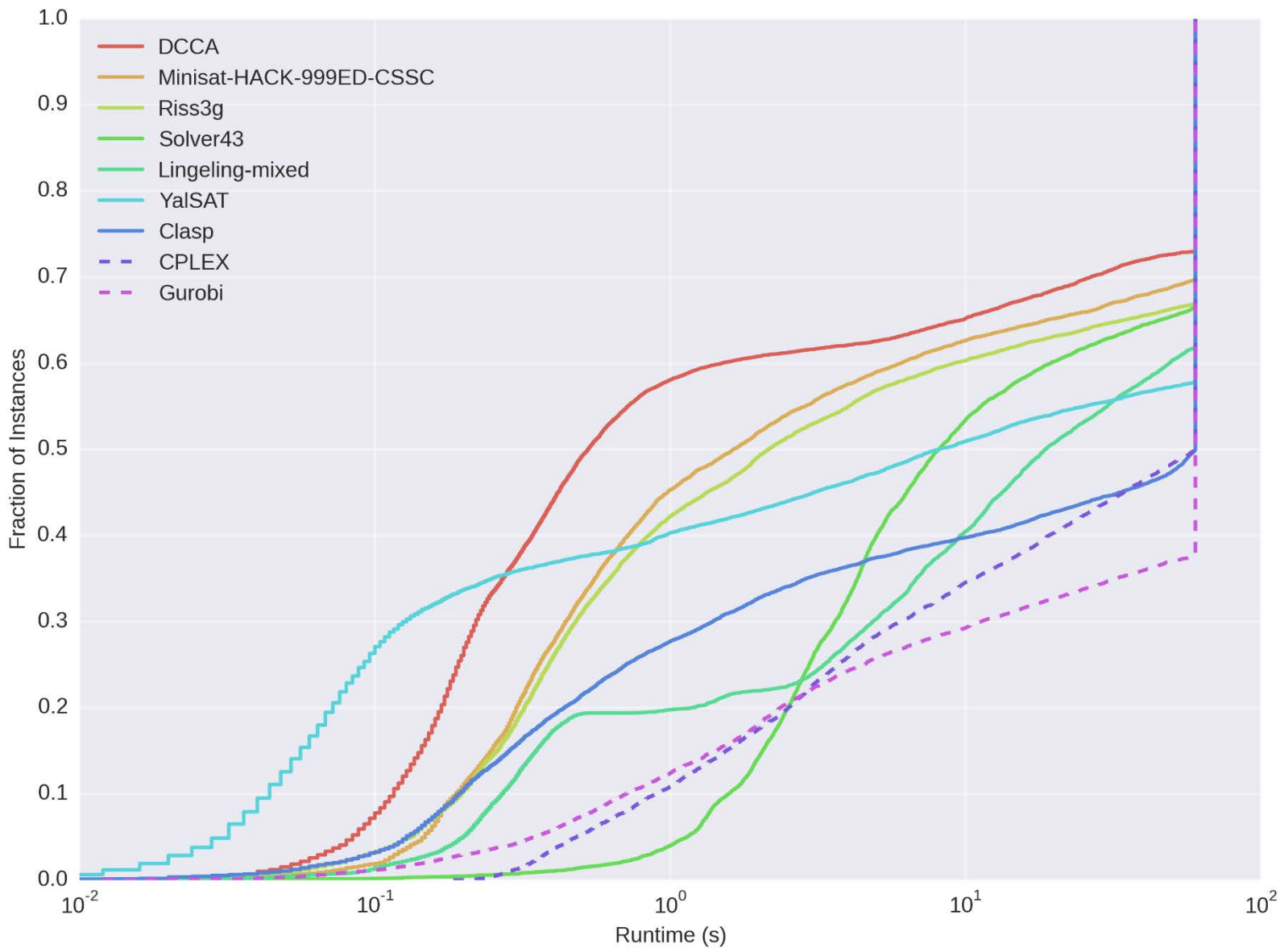
Feasibility Testing via MIP Encoding



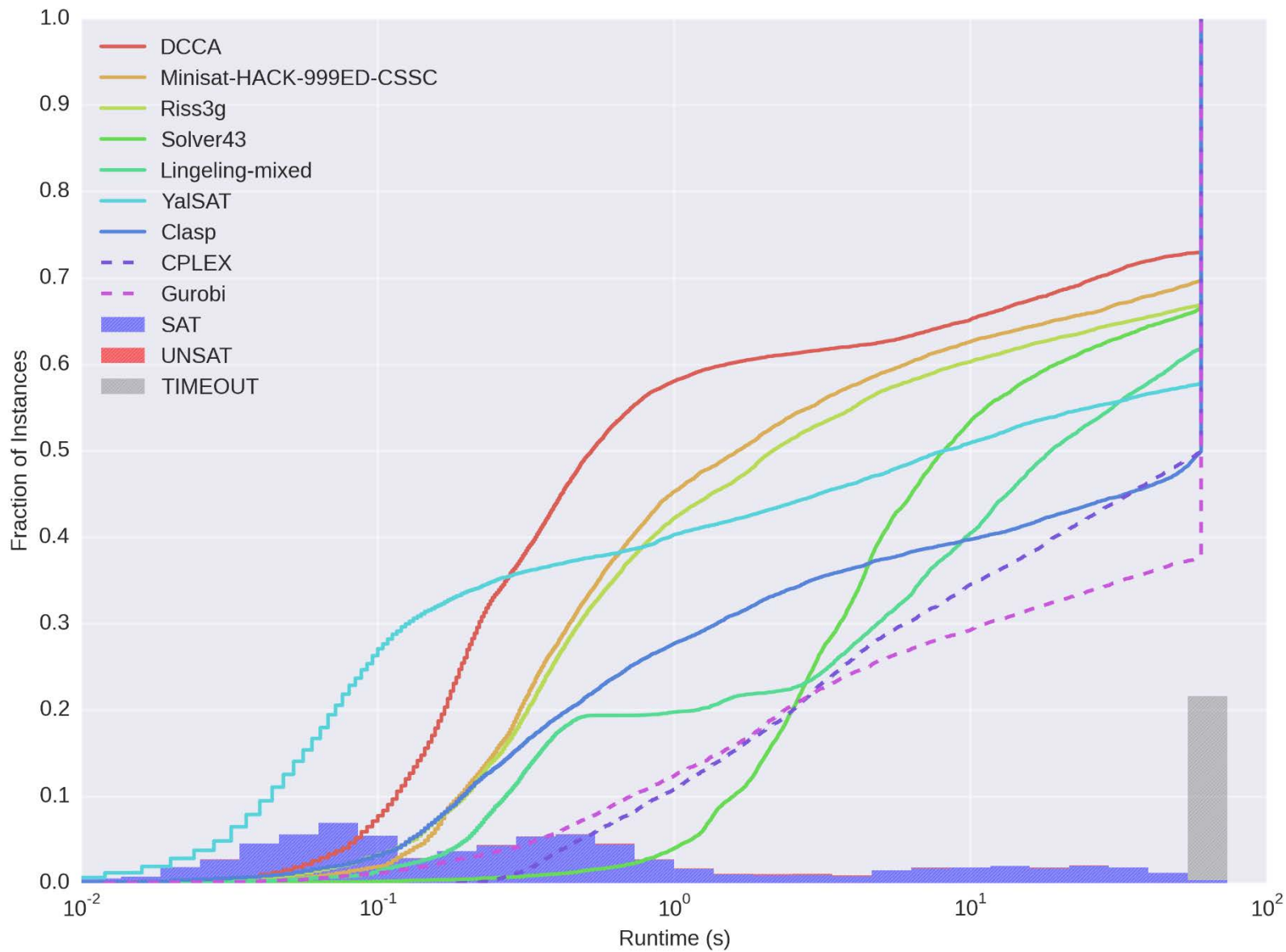
SAT Encoding

- $x_{s,c}$: 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 $(x_{s,c_1} \vee \dots \vee x_{s,c_n})$
- 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 $(\neg x_{s,c_1} \vee \neg x_{s,c_2})$
- 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

Feasibility Testing via SAT Encoding



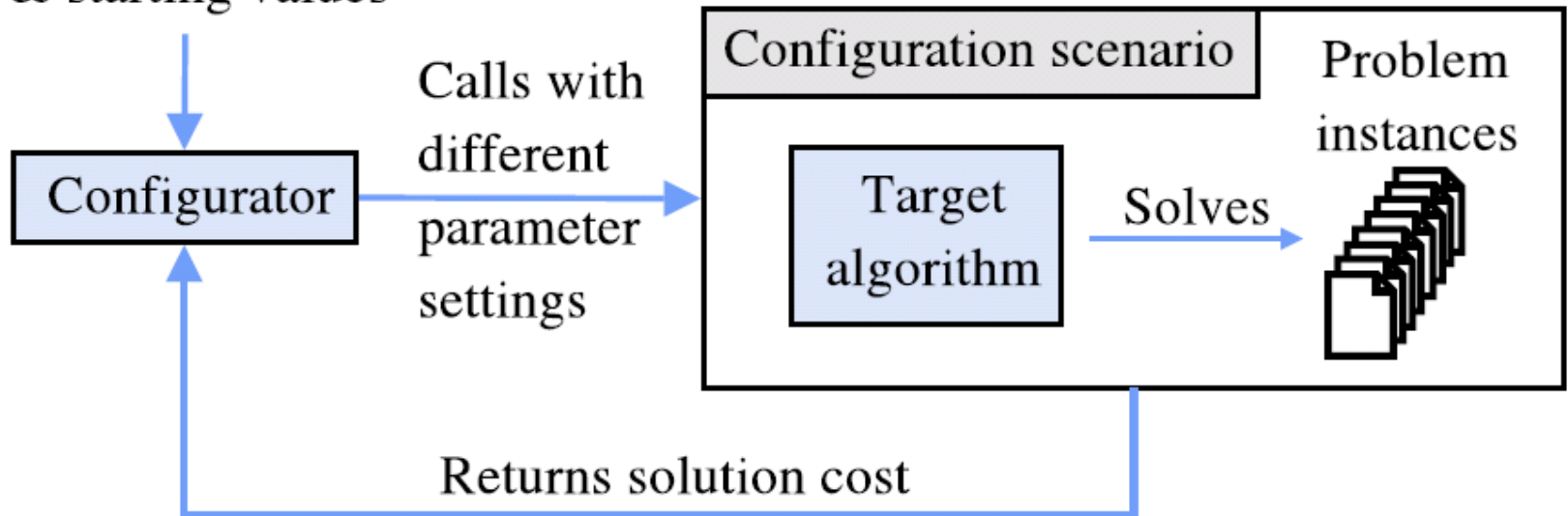
Feasibility Testing via SAT Encoding



Algorithm Configuration

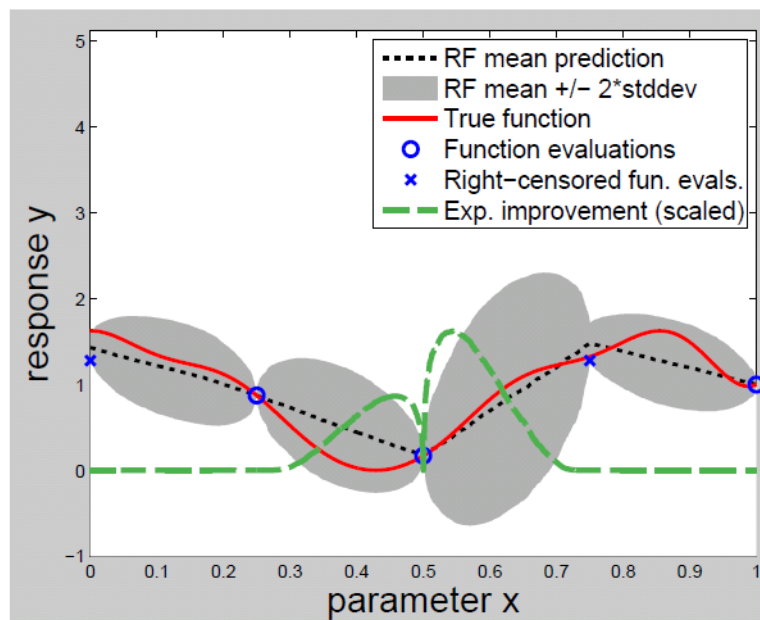
- High-performance solvers for NP-complete problems like SAT are typically **parameterized**
 - which branching heuristic, variable ordering, preprocessing strategy, clause learning technique, ...
- Address with **algorithm configuration**

Parameter domains
& starting values



Sequential Model-based Algorithm Configuration (SMAC)

[Hutter, Hoos & LB, 2011]



Initialize with a single run for the default configuration

repeat

Learn a random forest model $m : \Theta \times \Pi \rightarrow \mathbb{R}$ from data so far

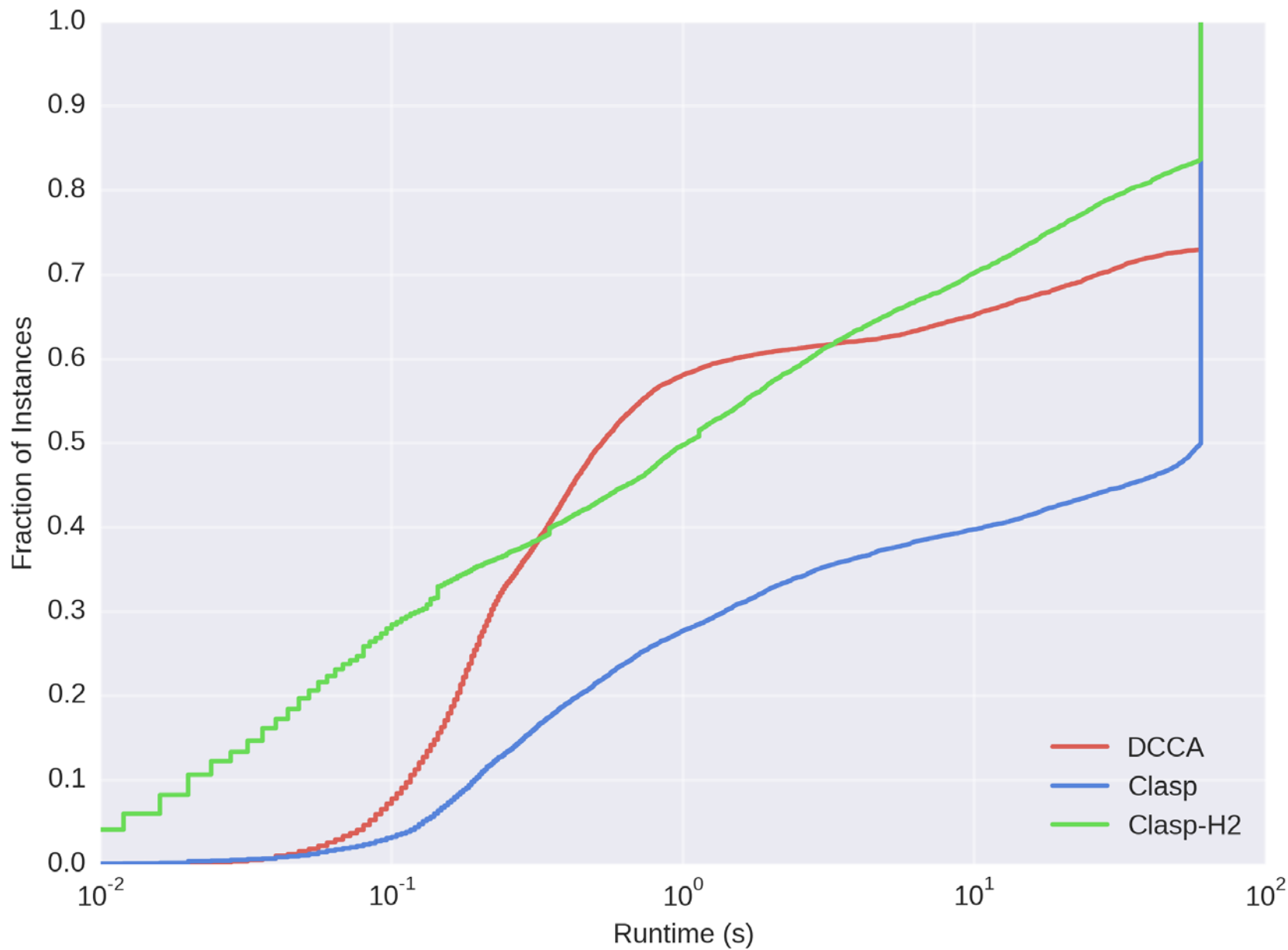
Marginalize out instance features: $f(\theta) = \mathbb{E}_{\pi}[m(\theta, \pi)]$

Find θ that maximizes expected improvement in $f(\theta)$ over incumbent

Compare θ to the incumbent, updating if it's better.

until *time budget exhausted*

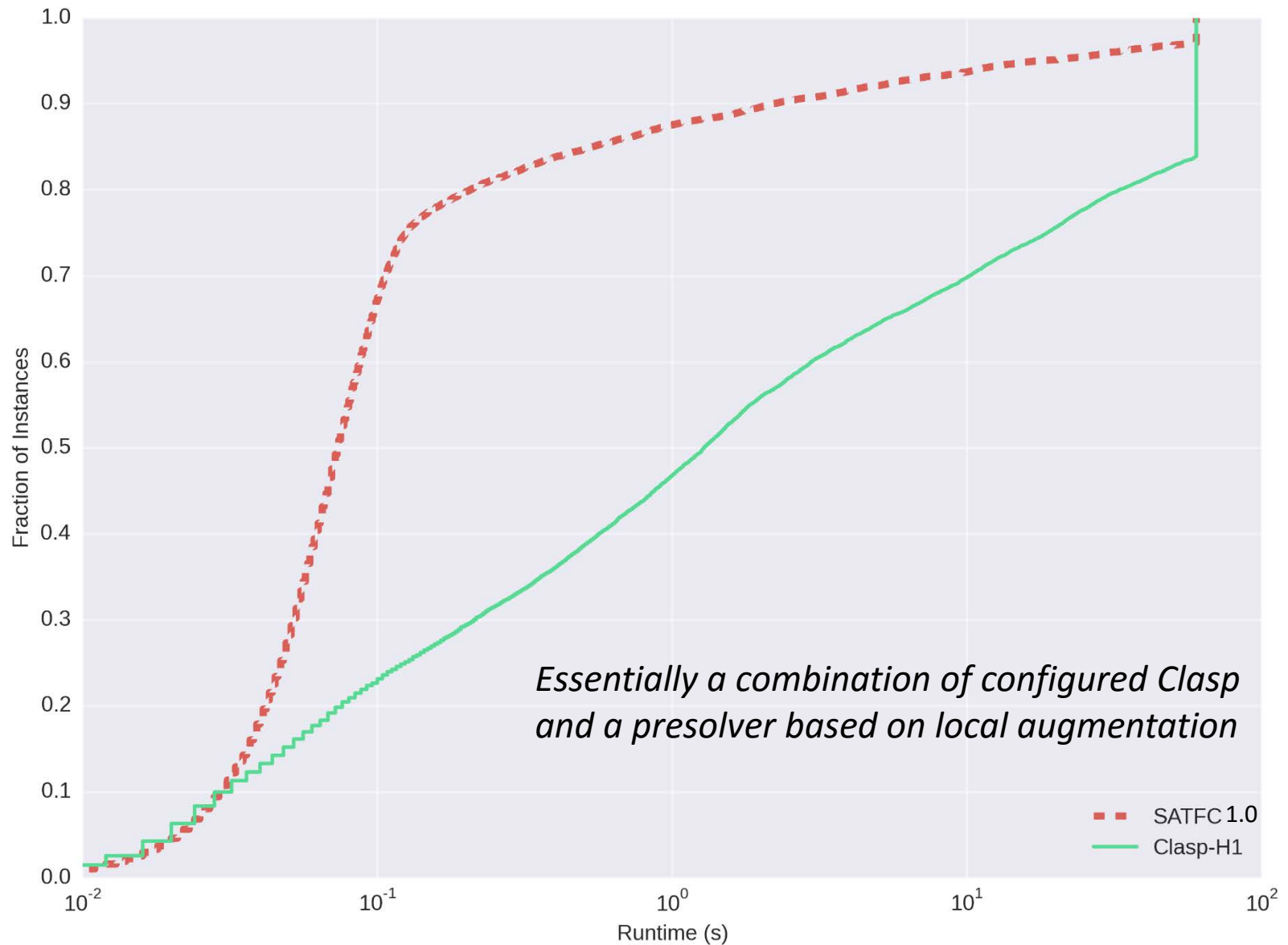
Best Configured Solver



Problem-Specific Speedups




- Incremental repacking
 - Local augmenting
 - **Can the previous assignment be augmented** by fixing all stations that don't neighbor the new station to their previous values?

SATFC 1.0 (officially released by the FCC in Nov 2014)



We Made the News! 😊

Source: TVTechnology.com

 **Tweet** **Like** 3 people like this. [Sign Up](#) to see what your friends like.  **PRINT**  **EMAIL** **SHARE**

DOUG LUNG / RF REPORT
11.07.2014 03:17 PM

Latest FCC Auction Software *Sucks Up Computer Power*

One terabyte of disk space, 4 GB of RAM and a modern multicore processor are recommended

A key part of the 600 MHz incentive auction will be determining whether sufficient spectrum remains in which to repack those stations that don't participate in the auction or have their bids accepted. This process involves creating "pairwise interference constraint" files that pre-calculate all the channels a station could be assigned to be consistent with preservation of coverage area and population served. During the auction, a "feasibility checker" can then be used to quickly determine whether a given channel assignment is feasible using the pairwise interference constraint files.

On Nov. 3, the Incentive Auction Task Force released a preliminary software that can be used to generate the pairwise constraint data and perform feasibility checking. The [notice](#) announcing the software cautions, "We emphasize that the software we are releasing today is the product of ongoing staff work implementing decisions adopted by the Commission and has not been finalized for use in the incentive auction. The full Commission will make all final decisions regarding the repacking process and other matters relating to the incentive auction at a later date."

[...]

The README on the [Feasibility Checker GitHub site](#) notes the SATFC (SAT-based Feasibility Checker) combines a formulation of feasibility checking based on propositional satisfiability with a heuristic pre-solver and a SAT solver tuned for the types of instances observed in auction simulations. SATFC was created by Auctionomics, notably Alexandre Fréchette, Guillaume Saulnier-Comte, Nick Arnosti, and Kevin Leyton-Brown. For more information on how it works see Kevin Leyton-Brown's presentation [Investigating the Viability of Exact Feasibility Testing](#). The paper includes a comparison of different solver methods, including PicoSAT (see [T-Mobile: 84 MHz of UHF Spectrum Can Be Reclaimed With Limited Impact on TV](#)).

The Feasibility Checker is primarily intended to run on Unix-like platforms and requires Java 7 or Java 8 to run. A modified version of the SAT solver clasp v2.2.3 compiled for JNA library usage is required, which in turn requires having gcc v4.8.1 or higher as well as the standard Unix C libraries. Instructions are provided on how to compile clasp. The Feasibility Checker requires two sets of data to run. One is the domain file that lists every station to be studied and the channels on which it can broadcast. The other is the pairwise interference constraint files data created by the Constraint Generator.

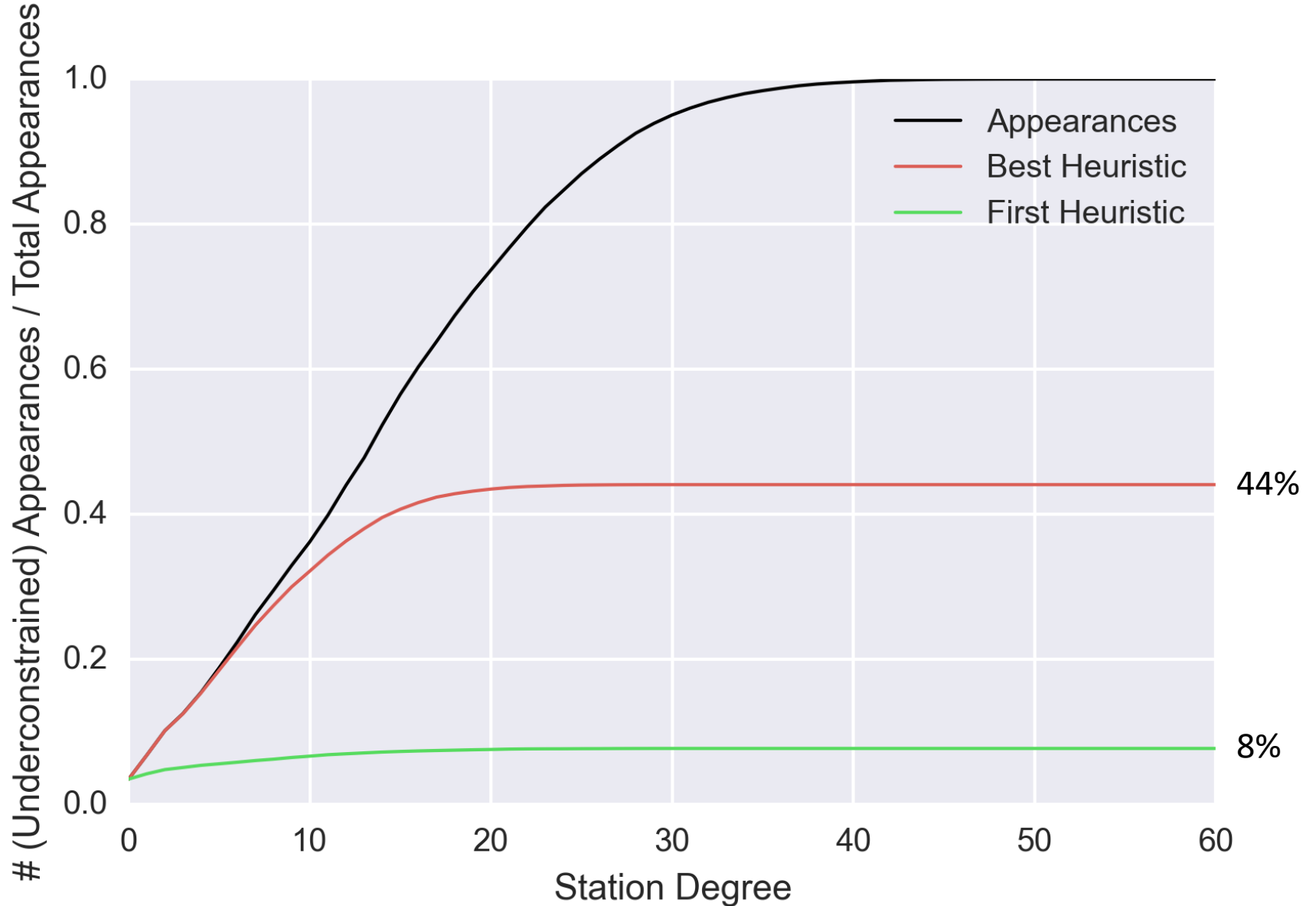
I doubt many individuals will have the time or resources to generate their own constraint files and perform some feasibility checks to see how the system will function. I would expect groups like the National Association of Broadcasters to put the software to the test. It will be interesting to see what they find!

But, they still want it to be faster!

Problem-Specific Speedups

- Incremental repacking
 - Local augmenting
 - **Can the previous assignment be augmented** by fixing all stations that don't neighbor the new station to their previous values?
 - Starting **local search solvers** with the previous assignment
- Problem simplification
 - **Graph decomposition**
 - Separately solve each component of the (induced) interference graph
 - Showing that any component is UNSAT suffices for the whole problem
 - **Unconstrained station removal**
 - Drop stations that, for every assignment of other stations, can always take a feasible value
 - Doing this first can cause even more graph decomposition

Unconstrained Stations

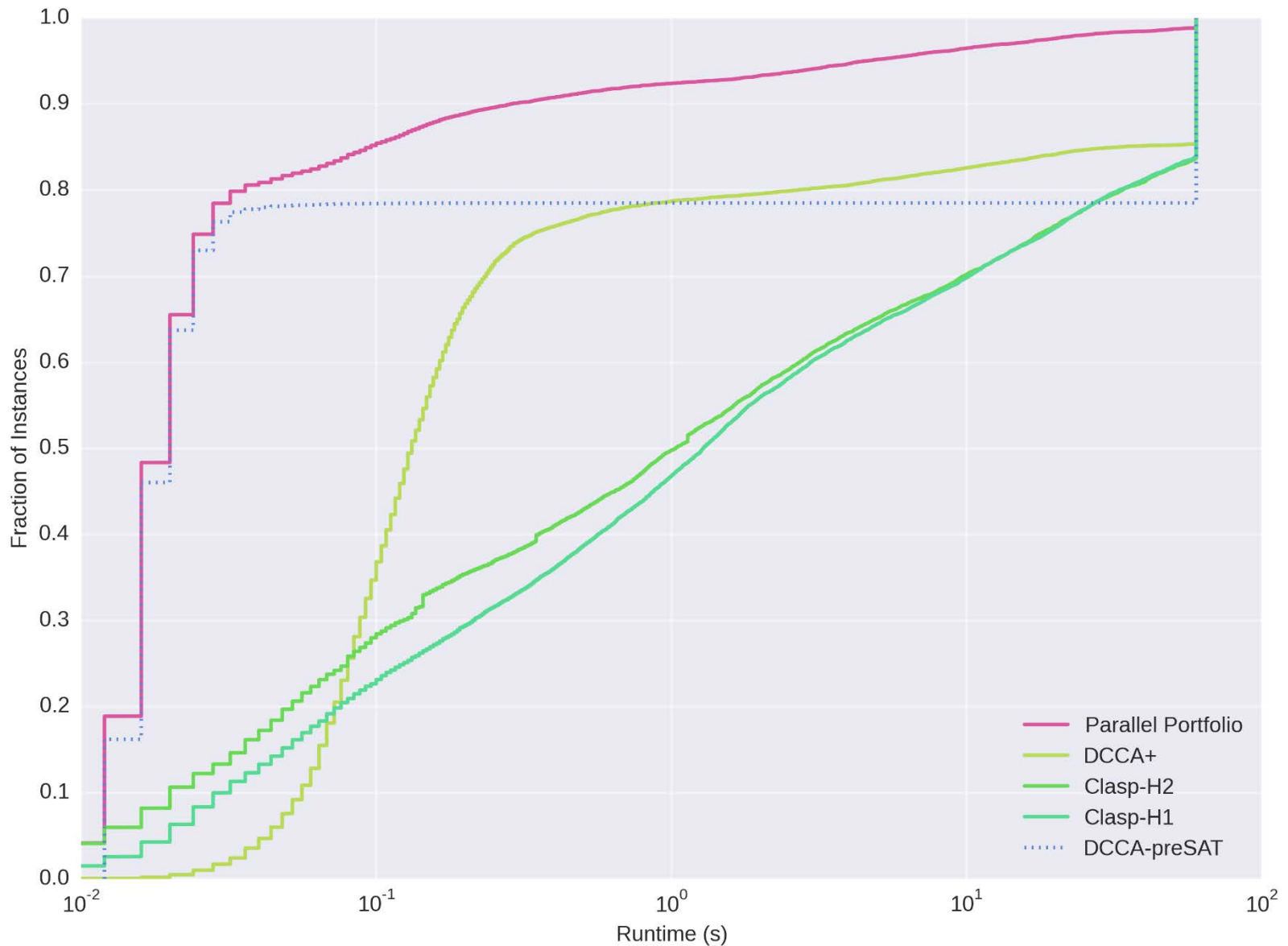


Algorithm Portfolios

- Often different solvers perform well on different problem instances
- Idea: build an **algorithm portfolio**, consisting of different algorithms that can work together to solve a problem
- **SATzilla**: state-of-the-art portfolio developed by my group (2003-present)
 - machine learning to **choose algorithm** on a per-instance basis
- **Iteratively configure** constituent solvers (“Hydra”)
 - Keep asking “which solver will offer the greatest marginal contribution to the existing portfolio?”



Performance of the Algorithm Portfolio



Caching

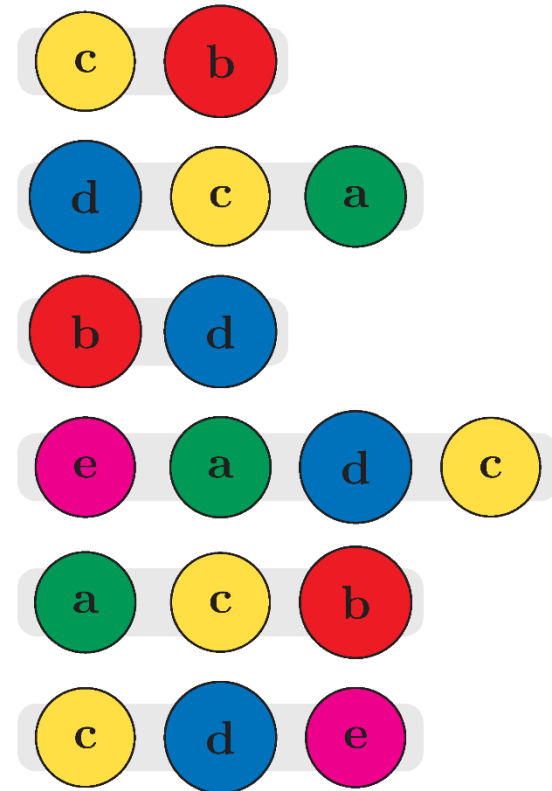
- We'd be willing to leverage enormous amounts of **offline computation** to make a faster solver
- Opportunity: we know the **constraint graph** in advance
- Obstacle: 2^n possible repacking problems
- Reason for optimism: **not all occur in practice**
 - The order in which stations exit the auction and hence have to be repacked is induced by valuations + auction mechanism
 - Valuations depend on the population served by a station, and hence are nonuniform
- So, would it work to **cache previous solutions**?
We tried and... No. **Almost no cache hits**
 - cache built on a training set of 200k problems
 - evaluated on test set of 10k problems

Supersets and Subsets

- Observation: if station set S is repackable, **so is every station set $S' \subseteq S$**
 - Useful because there are exponentially many such sets S'
 - Likewise, if S is not repackable, neither is any $S'' \supseteq S$
- Idea: when we encounter a new station set S , look for any **satisfiable superset** or any **unsatisfiable subset**
- Problem: we can't **query this cache** with a hash function
 - An exponential number of keys could match a given query
- Solution: build a **traditional cache** that we can hash into, and a **secondary cache** for subset/superset queries

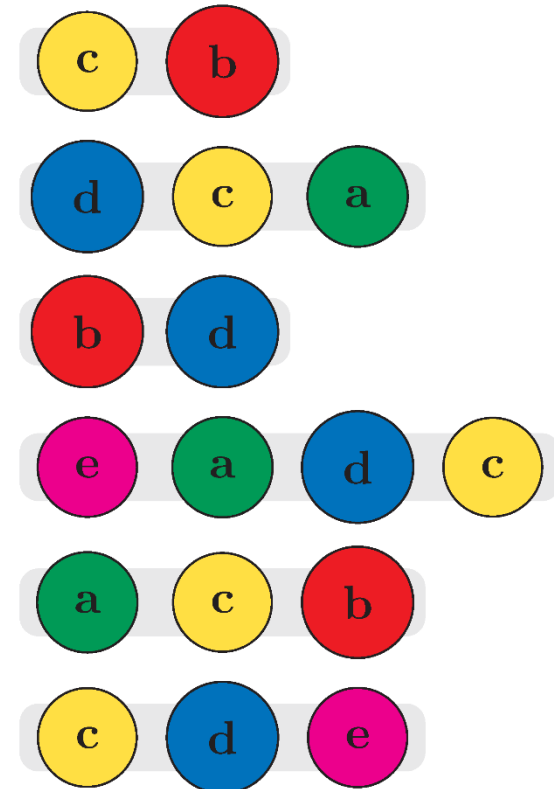
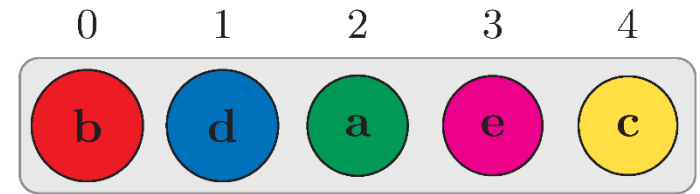
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache



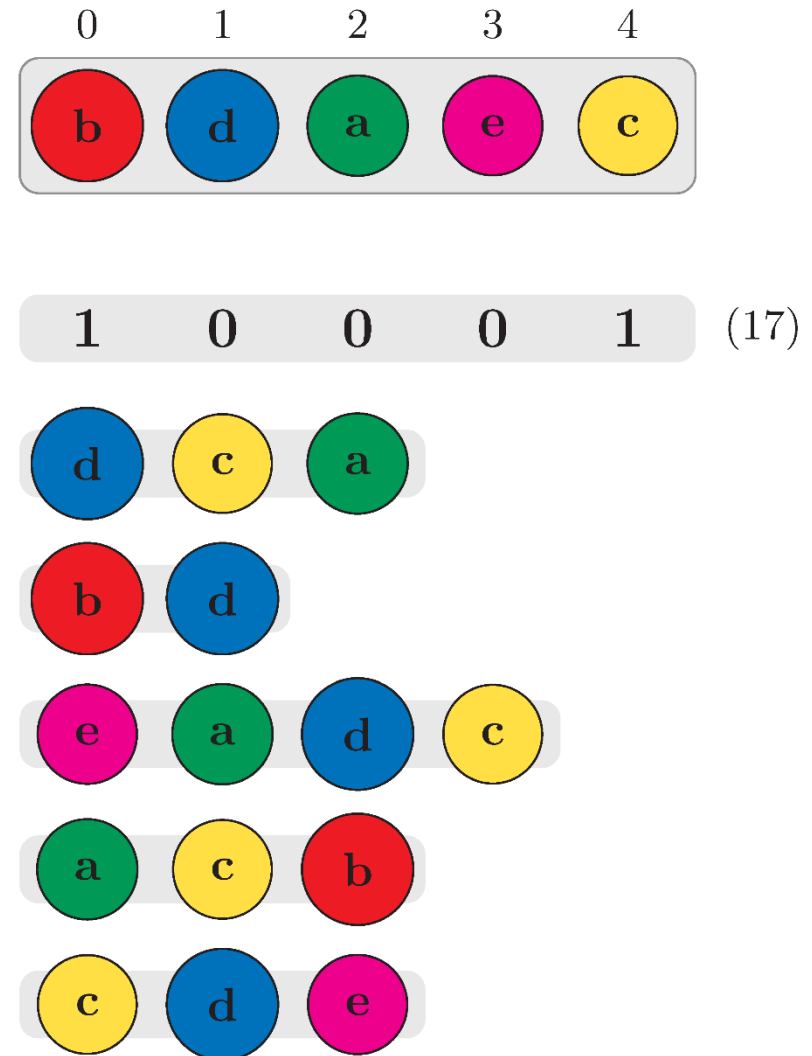
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o



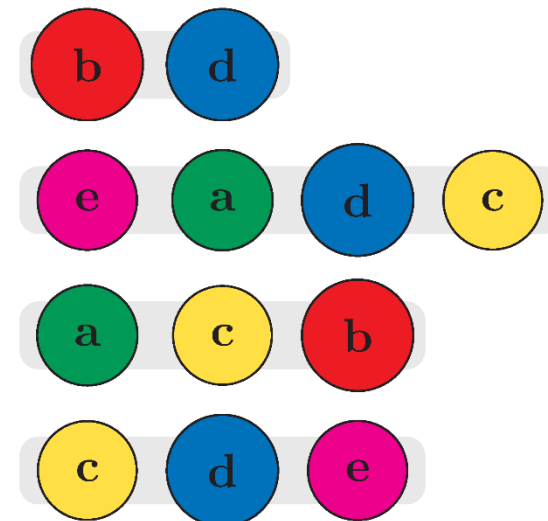
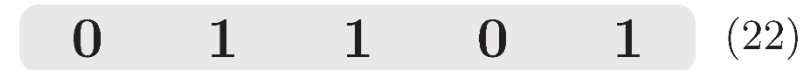
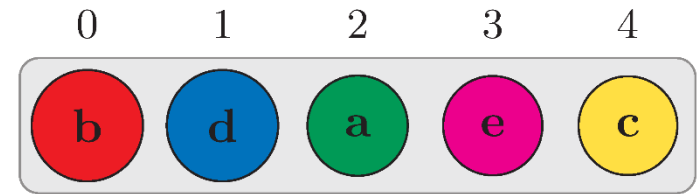
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o
 - Represent every station set as a **bit string** ordered by o



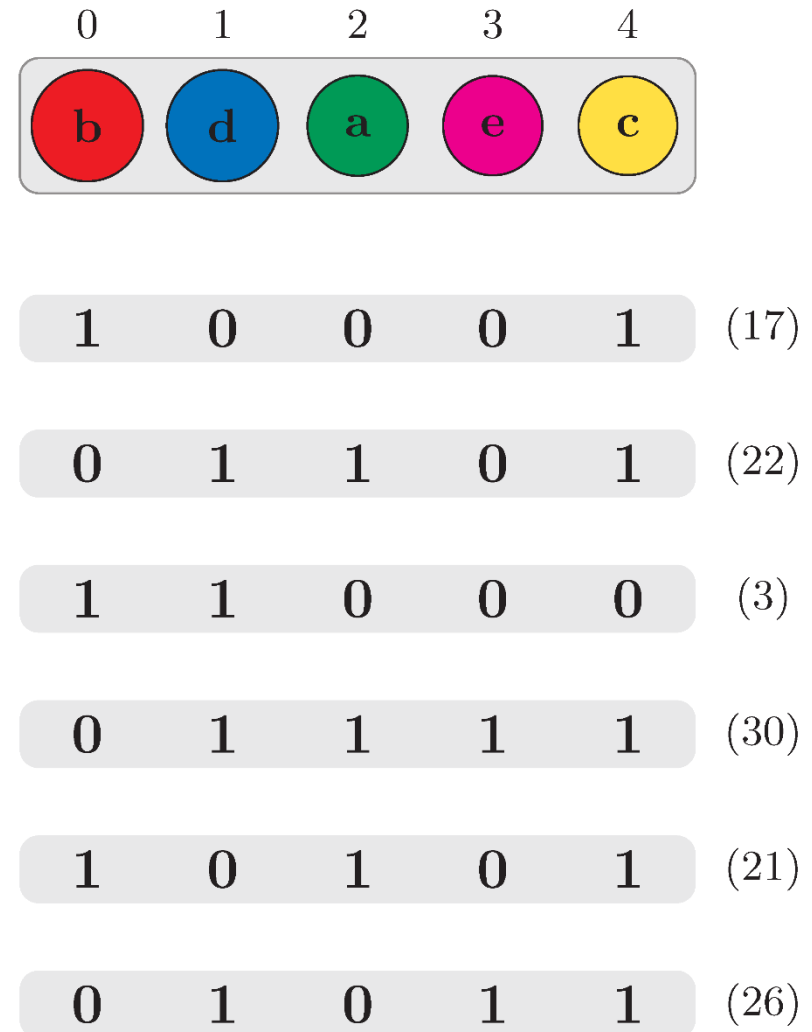
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o
 - Represent every station set as a **bit string** ordered by o



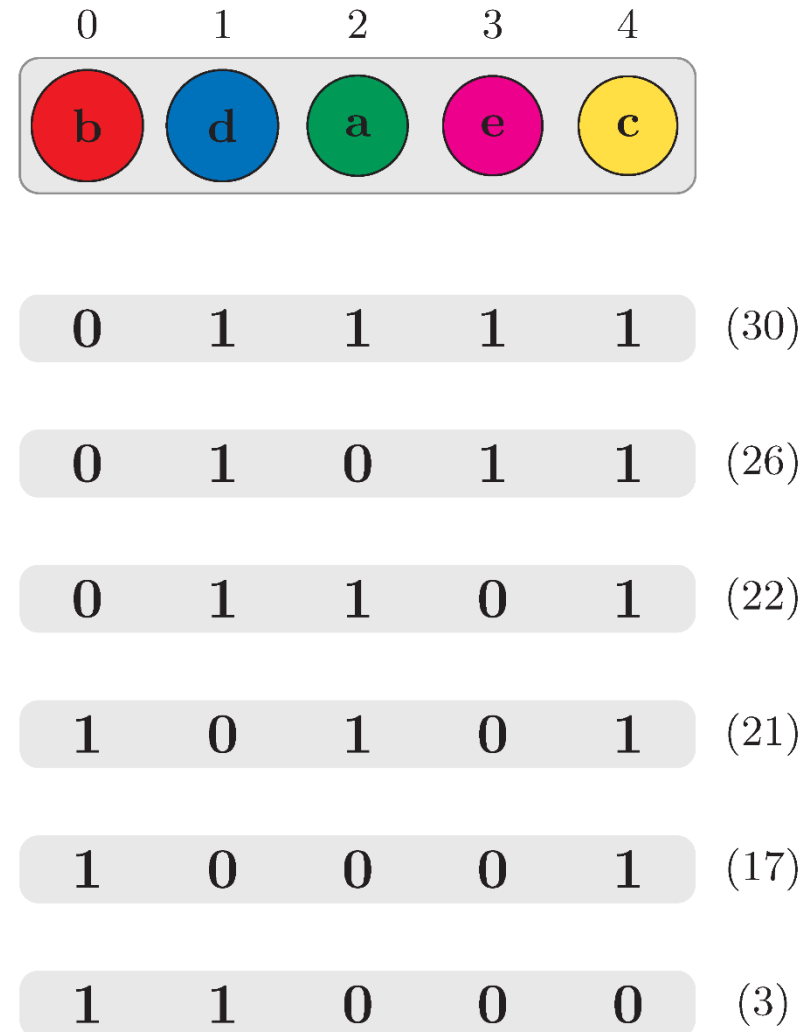
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o
 - Represent every station set as a **bit string** ordered by o
 - Sort station sets in **descending order**, interpreted as integers



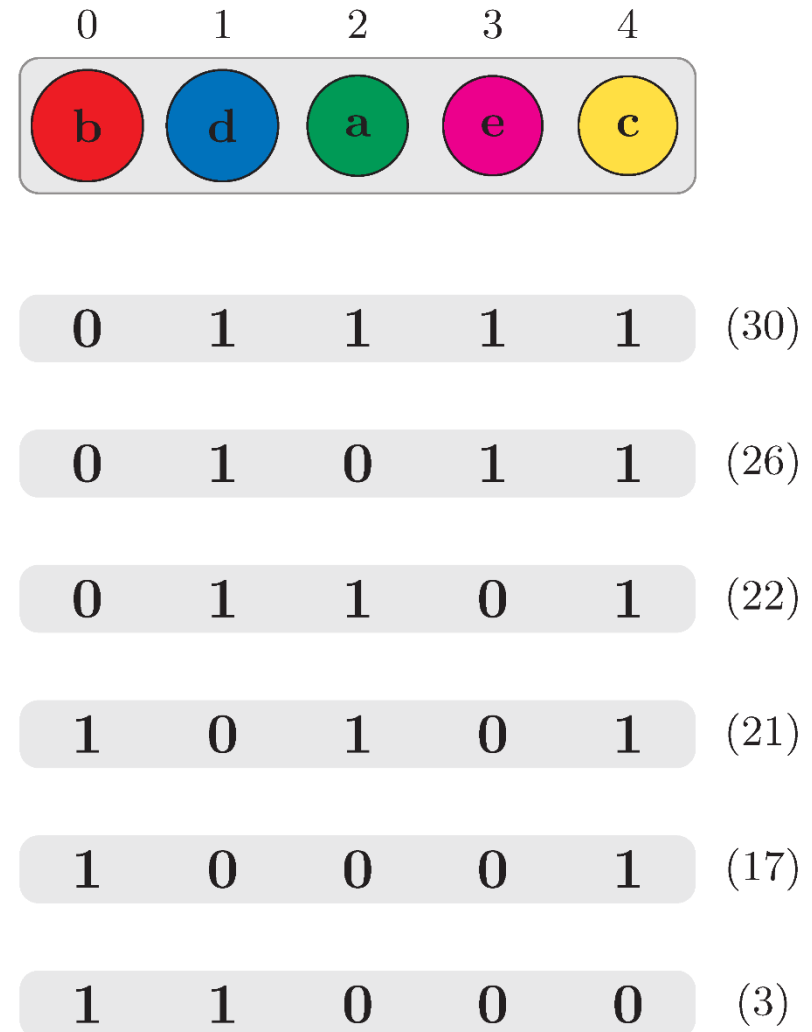
Secondary Cache (satisfiable instances)

- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o
 - Represent every station set as a **bit string** ordered by o
 - Sort station sets in **descending order**, interpreted as integers



Secondary Cache (satisfiable instances)

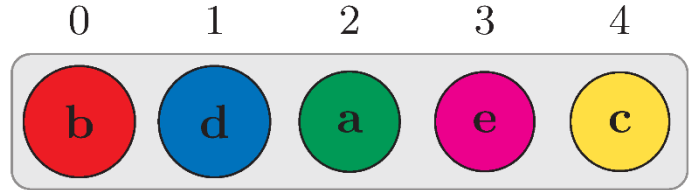
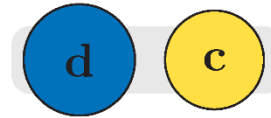
- Start with **all station sets** from the primary cache
- **Secondary cache** is defined by some ordering o
 - Represent every station set as a **bit string** ordered by o
 - Sort station sets in **descending order**, interpreted as integers
 - Very compact: 200k entries, each 2k stations/bits, takes 50 MB
 - So: we can afford multiple secondary caches, each with a different (random) ordering



Superset Queries

Does the cache contain any
superset of the query S ?

Get a superset of:



0 1 1 1 1 (30)

0 1 0 1 1 (26)

0 1 1 0 1 (22)

1 0 1 0 1 (21)

1 0 0 0 1 (17)

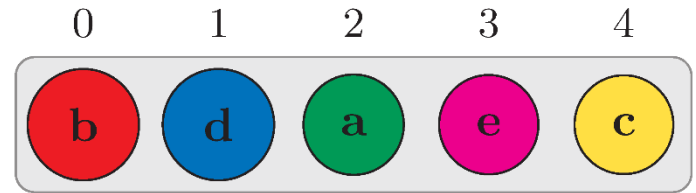
1 1 0 0 0 (3)

Superset Queries

Does the cache contain any
superset of the query S ?

Get a superset of:

0 1 0 0 1 (18)



0 1 1 1 1 (30)

0 1 0 1 1 (26)

0 1 1 0 1 (22)

1 0 1 0 1 (21)

1 0 0 0 1 (17)

1 1 0 0 0 (3)

Superset Queries

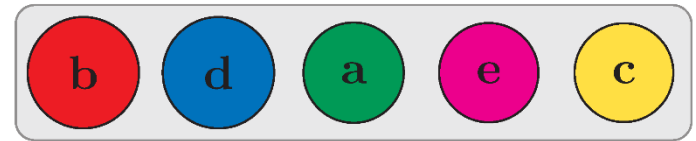
Does the cache contain any **superset of the query S** ?

- Perform **binary search** to find the index corresponding to S
 - If it's there: **direct hit**
- If not, look for a superset
 - This will be an entry “larger” than S , and thus **ordered earlier**

Get a superset of:

0 1 0 0 1 (18)

0 1 2 3 4



0 1 1 1 1 (30)

0 1 0 1 1 (26)

0 1 1 0 1 (22)

1 0 1 0 1 (21)



1 0 0 0 1 (17)

1 1 0 0 0 (3)

Superset Queries

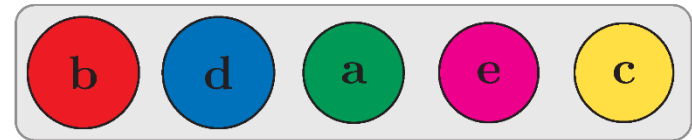
Does the cache contain any **superset of the query S** ?

- Perform **binary search** to find the index corresponding to S
 - If it's there: **direct hit**
- If not, look for a superset
 - This will be an entry “larger” than S , and thus **ordered earlier**
 - Work linearly through the list
 - Efficient test for a superset: “Is the cached bit string **bitwise logically implied** by the query?”

Get a superset of:

0 1 0 0 1 (18)

0 1 2 3 4



0 1 1 1 1 (30)

0 1 0 1 1 (26)

0 1 1 0 1 (22)

↑ 1 0 1 0 1 (21)

1 0 0 0 1 (17)

1 1 0 0 0 (3)

Superset Queries

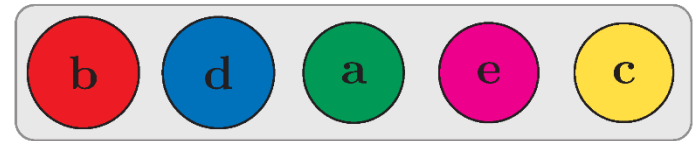
Does the cache contain any **superset of the query S** ?

- Perform **binary search** to find the index corresponding to S
 - If it's there: **direct hit**
- If not, look for a superset
 - This will be an entry “larger” than S , and thus **ordered earlier**
 - Work linearly through the list
 - Efficient test for a superset: “Is the cached bit string **bitwise logically implied** by the query?”
- Speed things up with multiple secondary caches:
 - look up S in all of them
 - Use the secondary cache with the **smallest number of entries** $> S$
- 200k cache entries → average query time **30 ms**

Get a superset of:

0 1 0 0 1 (18)

0 1 2 3 4



0 1 1 1 1 (30)

0 1 0 1 1 (26)

0 1 1 0 1 (22)

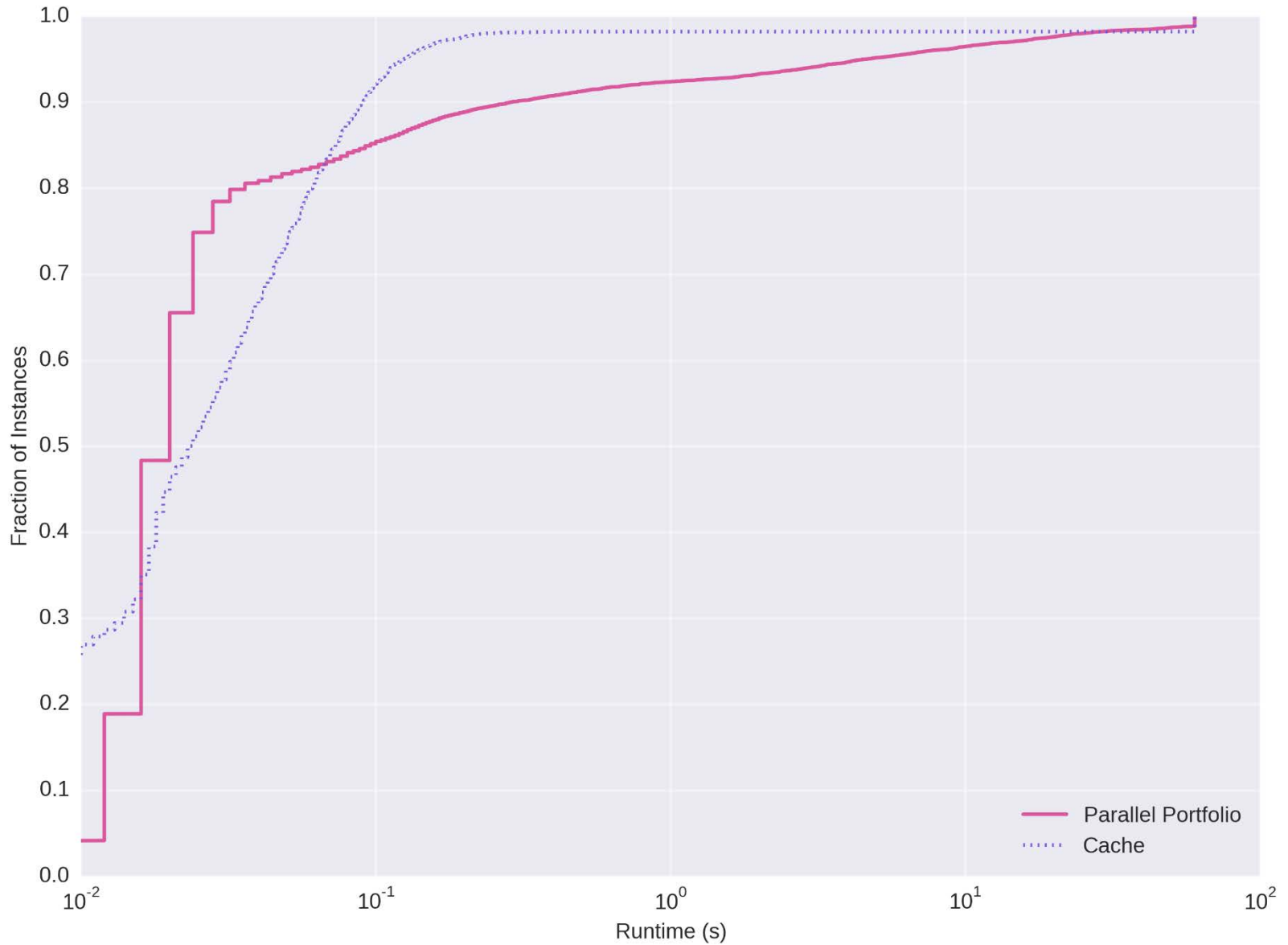
1 0 1 0 1 (21)



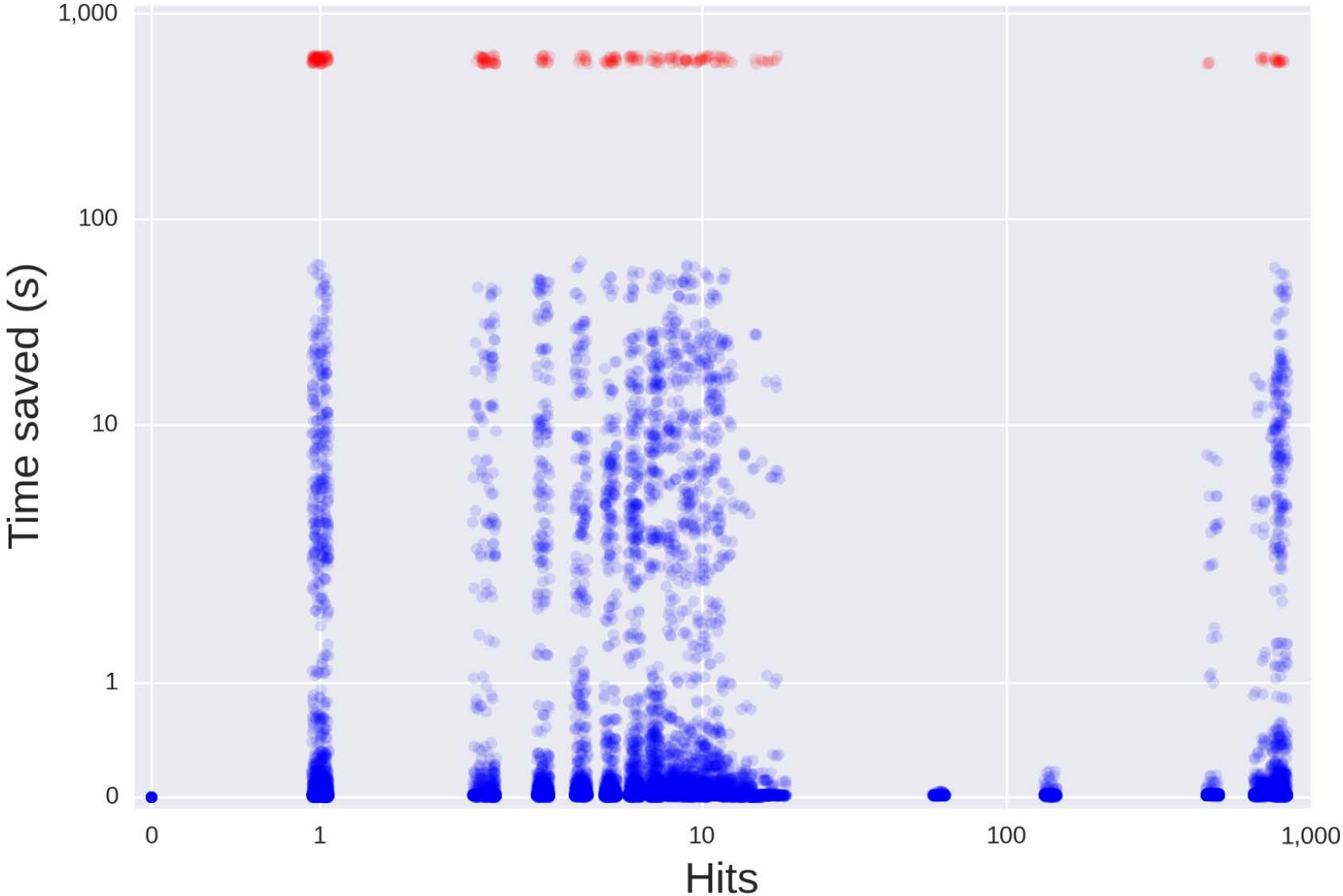
1 0 0 0 1 (17)

1 1 0 0 0 (3)

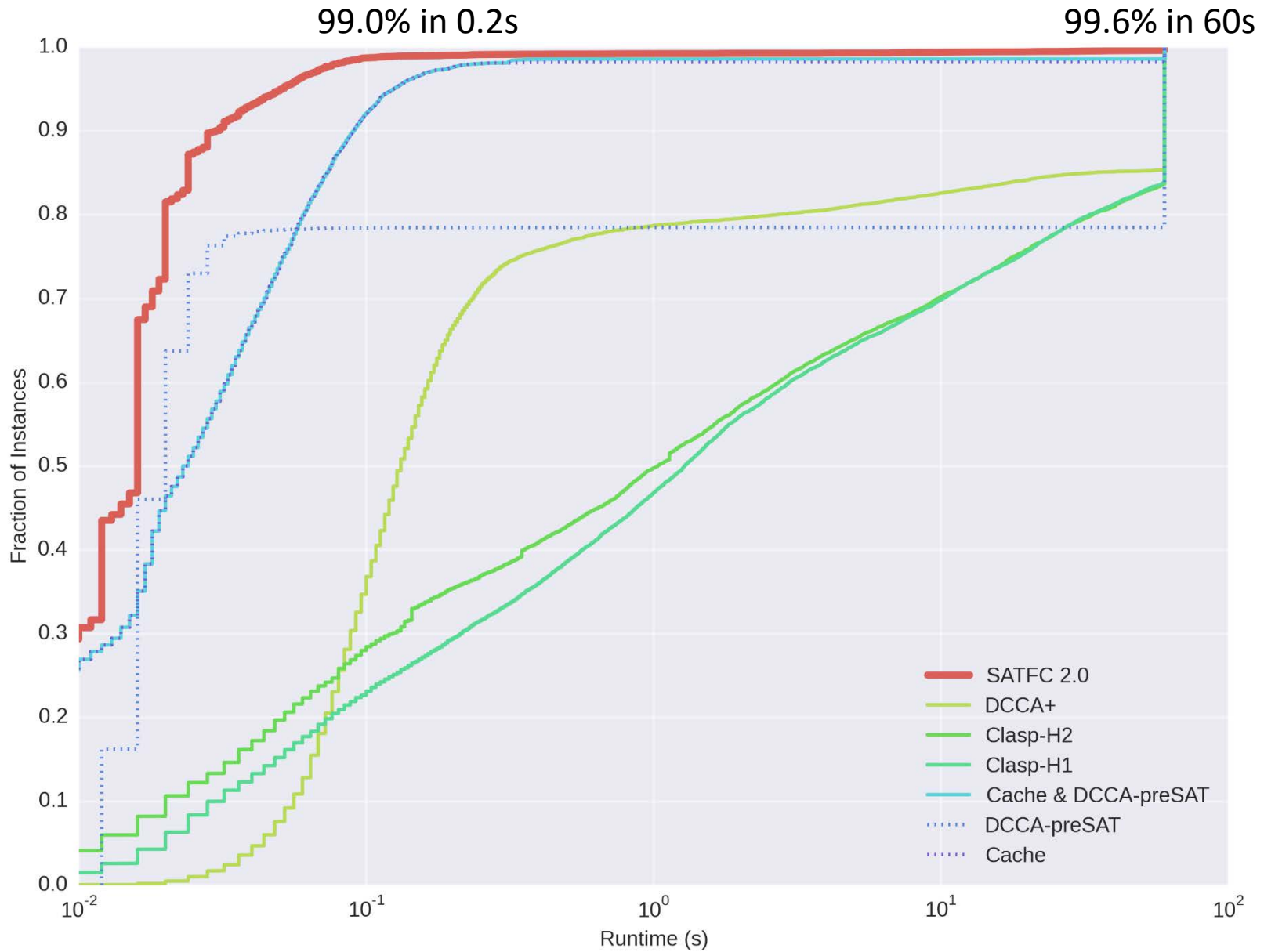
Cache Performance



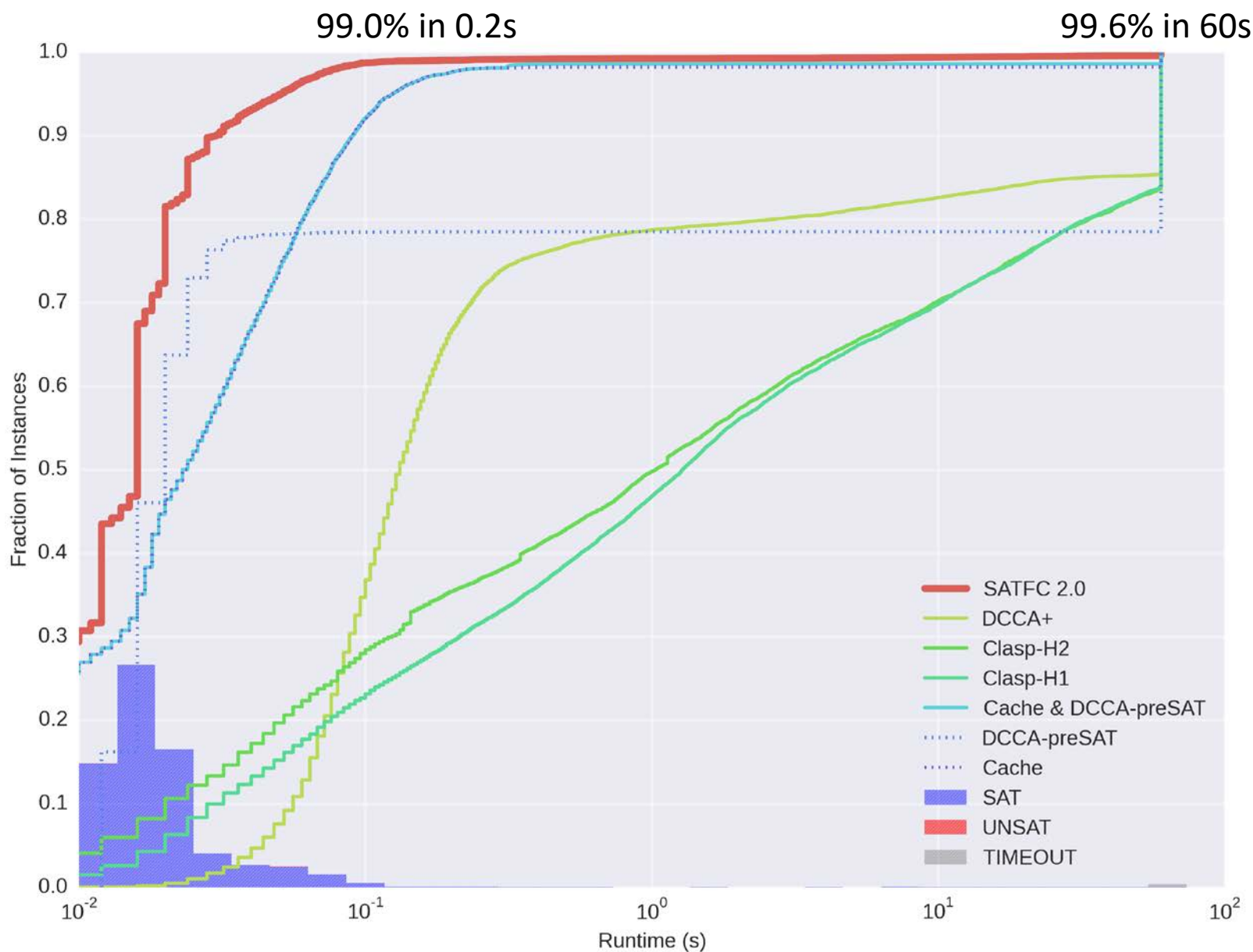
Containment Cache Analysis



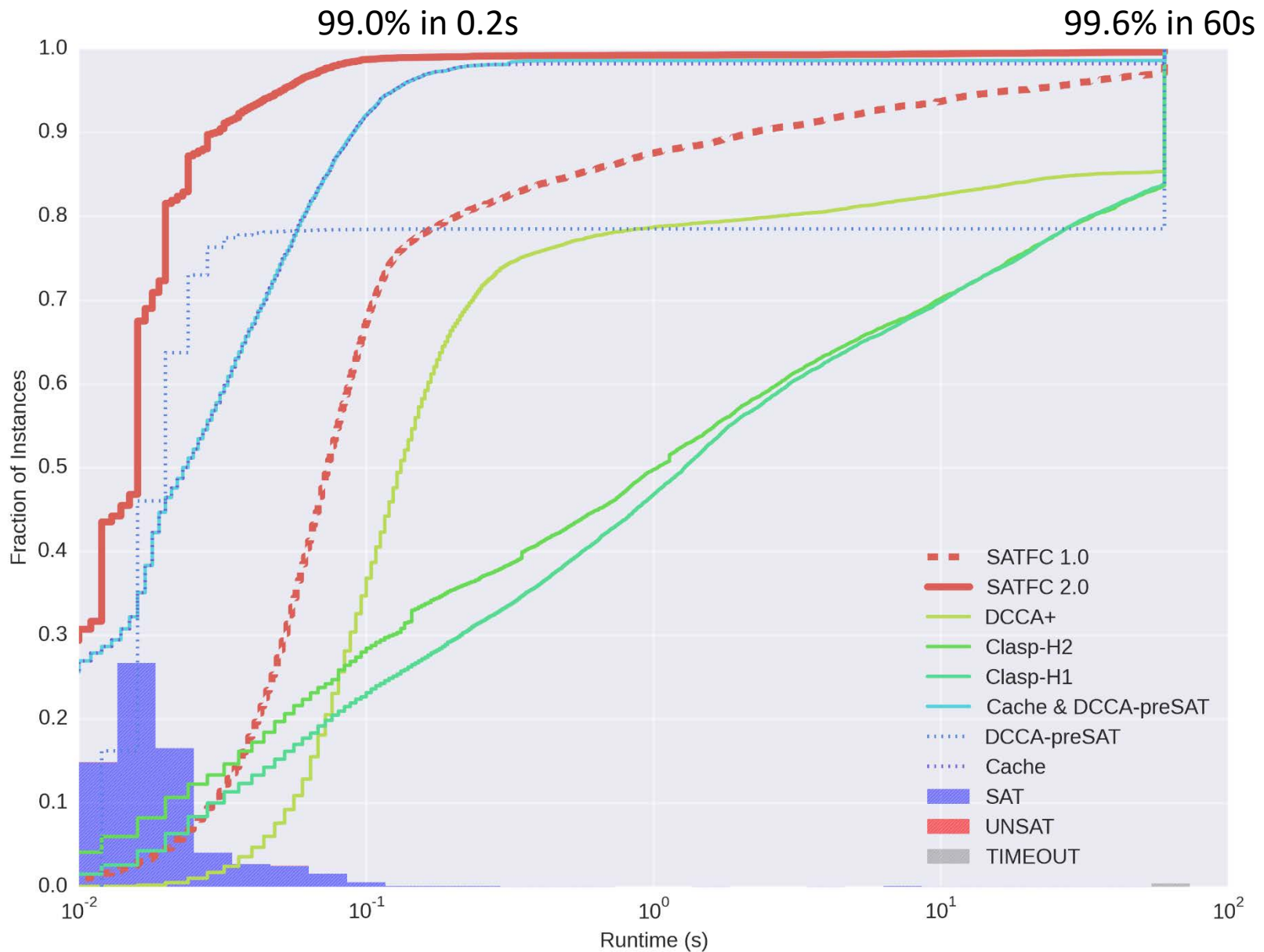
Putting It All Together



Putting It All Together



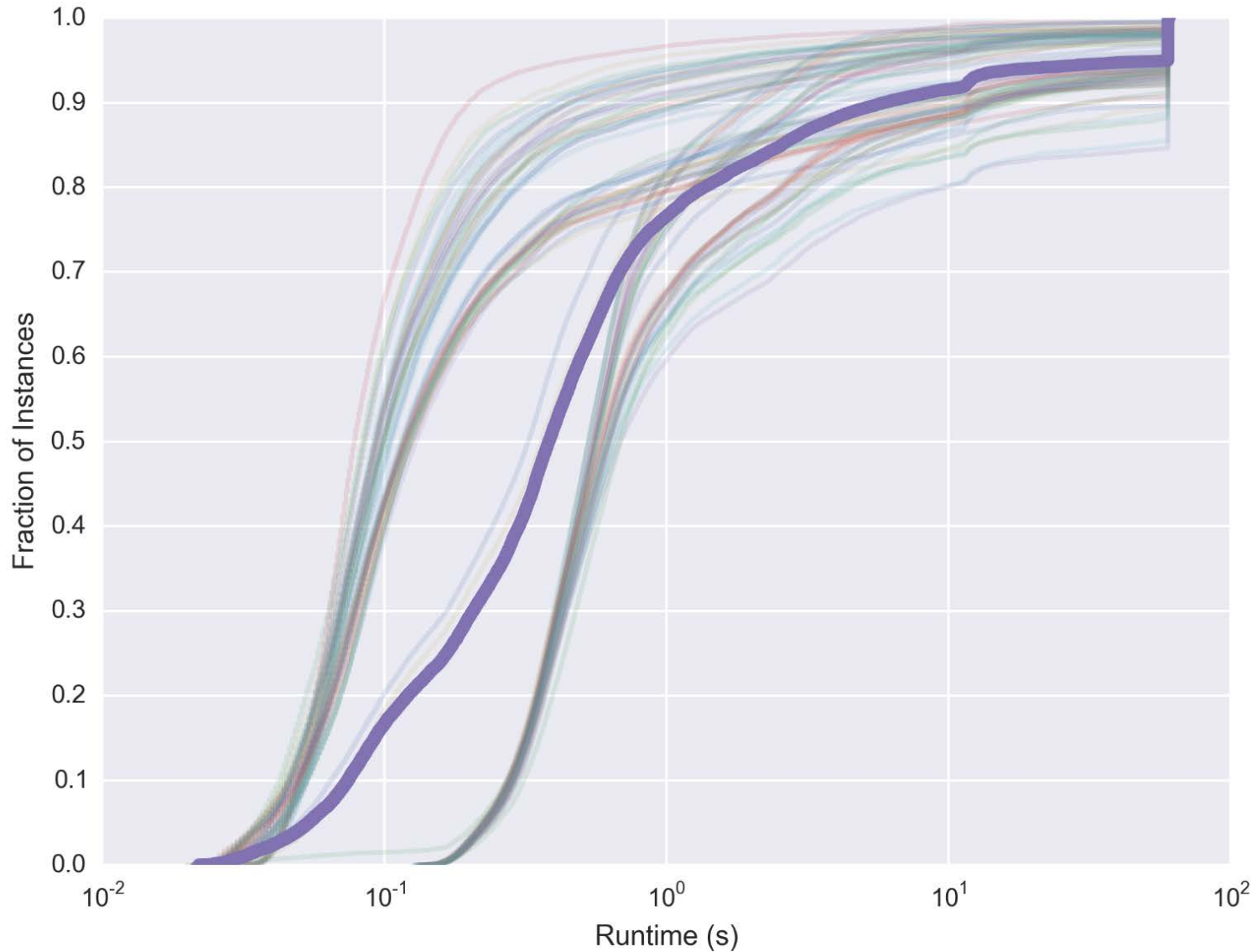
Putting It All Together



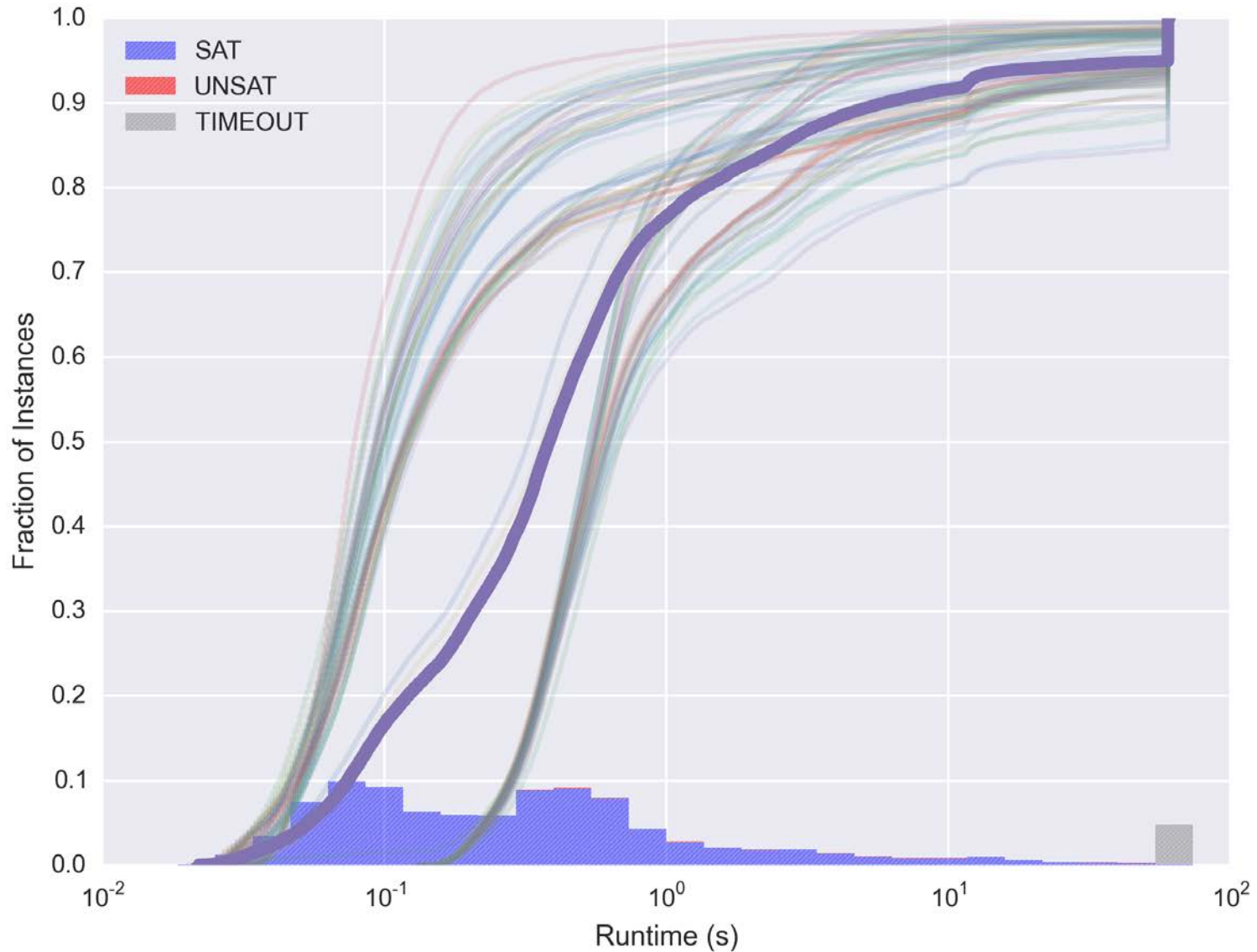
Performance on Further Simulations

- 80 **auctions**
 - 5,924 – 43,721 instances per auction
- 1,109,707 **instances** total
 - all not solvable by directly augmenting the previous solution
 - about 20% of the problems encountered in full simulations
- data generated Oct 22 – Nov 6 using the FCC’s “smoothed ladder” simulator and varying **simulation assumptions**:
 - stations’ valuations
 - which stations opt to participate
 - how much spectrum is being cleared
 - 84 MHz: pack into channels 14 – 36
 - 126 MHz: pack into channels 14 – 29
 - the timeout given to SATFC (1 min; 5 min)

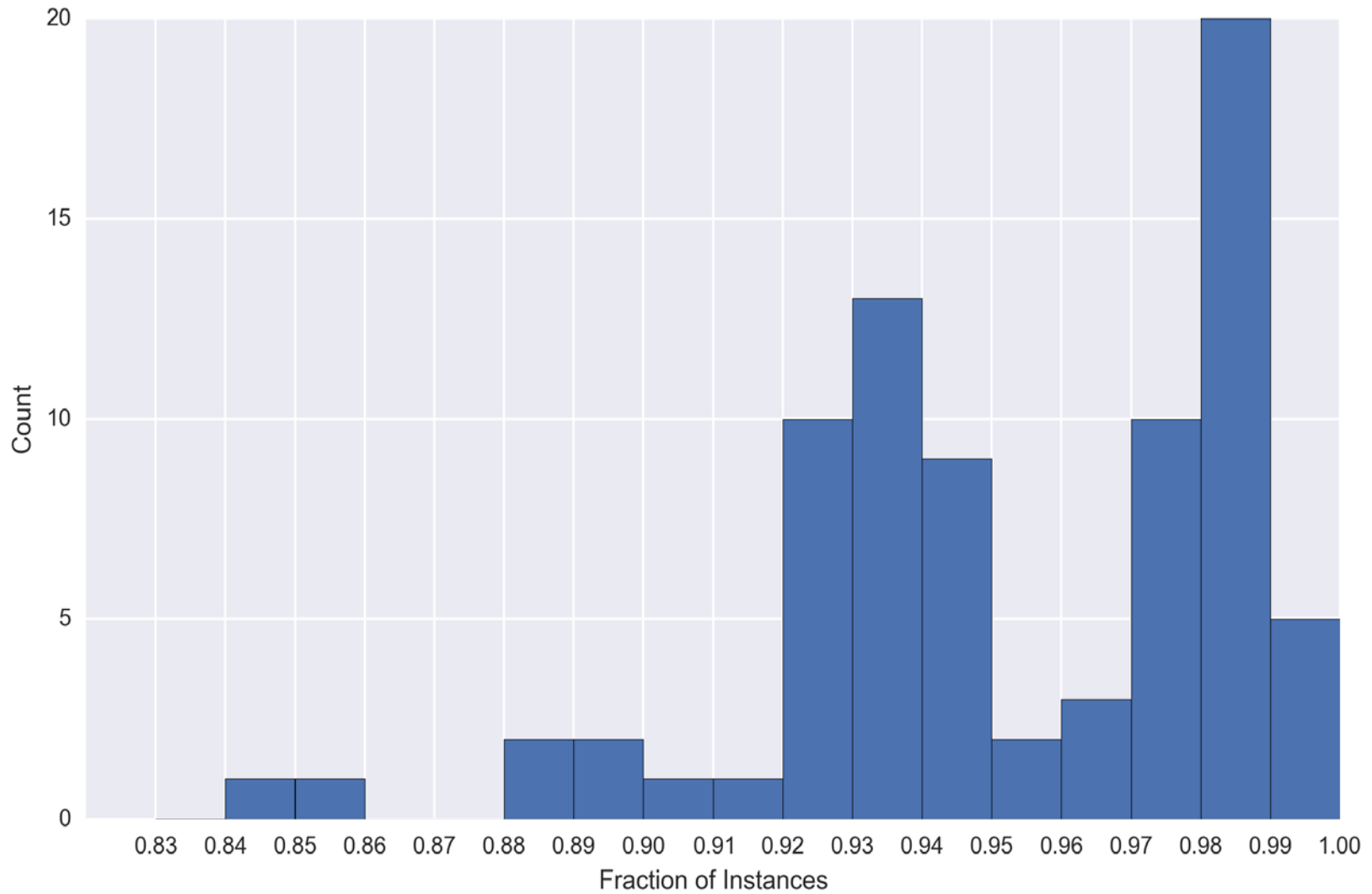
SATFC Performance on New Data (no cache)



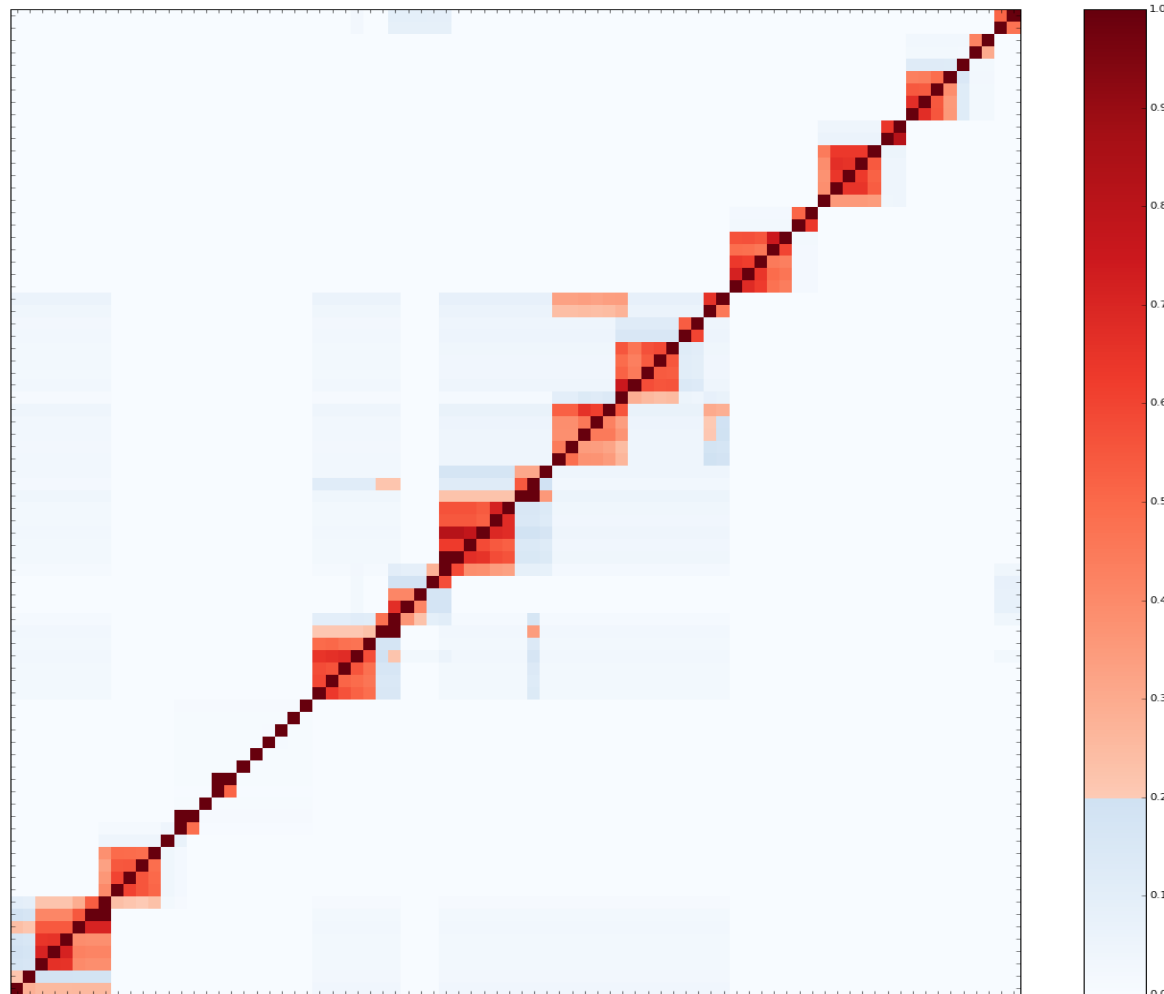
SATFC Performance on New Data (no cache)



Solved Percentages (no cache)

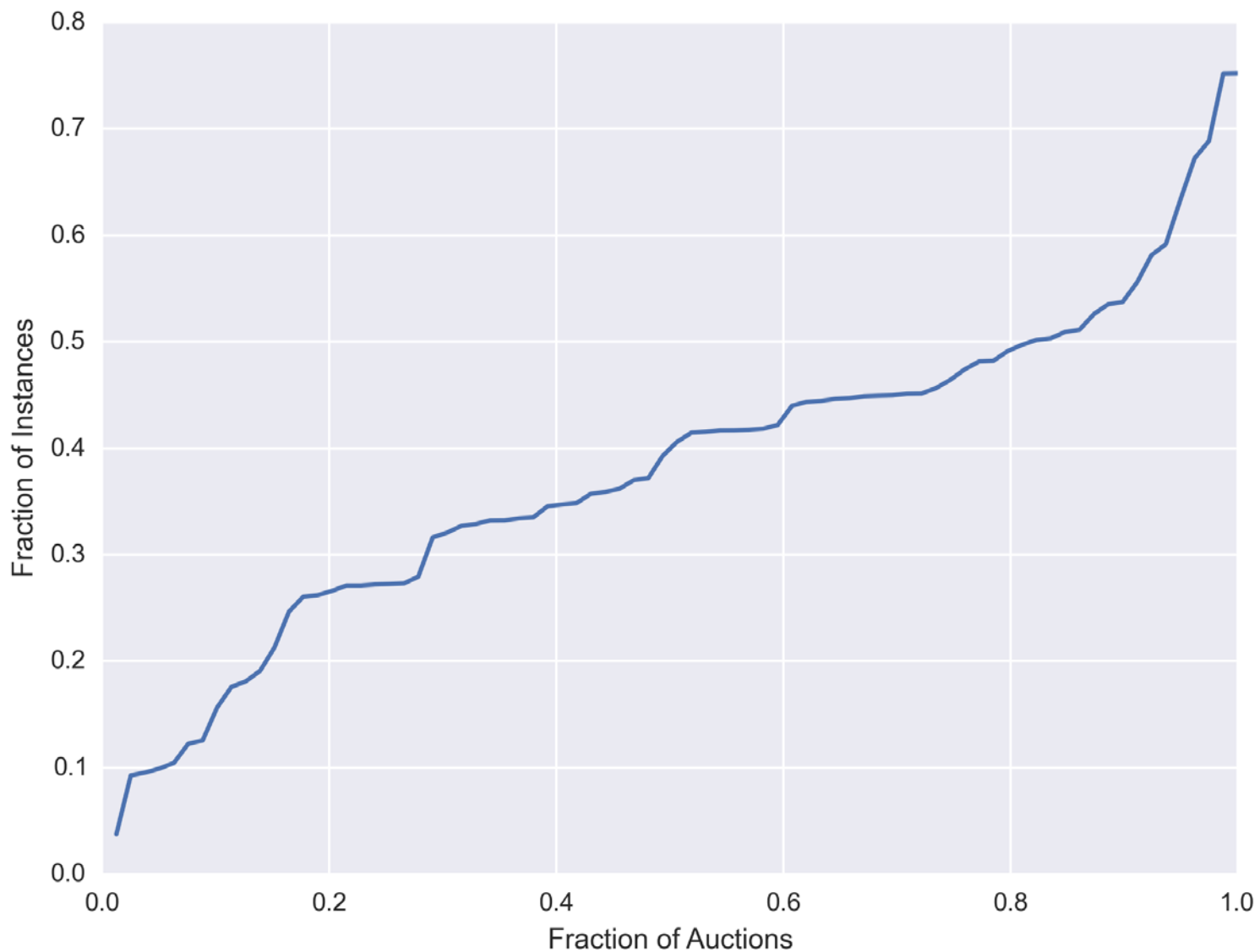


Similarity between auction simulations



For each auction (row), train a cache based on every auction that does not share at least 20% of the same problems; evaluate

Cache Performance (leave-one-out analysis)



Conclusions

- Spectrum reallocation is a **socially important problem** that poses interesting new challenges for auction theory
- The FCC has proposed the use of **descending auctions** to buy back spectrum from TV broadcasters
 - **advantage:** “obviously” strategyproof
 - **disadvantage:** millions of NP-complete problems must be solved in real time; auction revenue suffers when they can’t be
- We showed how this repacking problem can be **solved at national scale**, via:
 - algorithm configuration; algorithm portfolios
 - problem-specific speedups; problem simplifications
 - containment caching
 - likely of independent interest: any setting where many problems must be solved, derived from subsets of a given constraint set