Economics and Computer Science of a Radio Spectrum Reallocation

Kevin Leyton-Brown

Computer Science Department
University of British Columbia

&

Auctionomics, Inc.

[Auction]omics

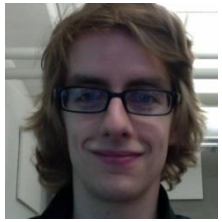
FCC's "Incentive Auction"

- Over 13 months in 2016-17 the FCC held an **"incentive auction"** to repurpose radio spectrum from broadcast television to wireless internet

- In total, the auction yielded **$19.8 billion**
  - **over $10** billion was paid to 175 broadcasters for voluntarily relinquishing their licenses across 14 UHF channels (84 MHz)
  - Stations that continued broadcasting were **assigned potentially new channels** to fit as densely as possible into the channels that remained
  - The government netted **over $7 billion** (used to pay down the national debt) after covering costs

Thanks to all those who helped make this work possible!

Key collaborators on market design: **Paul Milgrom**  , **Ilya Segal** 

Student leads on feasibility checking: **Neil Newman**  , **Alexandre Fréchette** 

Colleagues and students (then) at UBC:

- Chris Cameron
- **Holger Hoos**
- **Frank Hutter**
- Ashiqur Khudabukhsh
- Steve Ramage
- James Wright
- Lin Xu

Students who made code contributions:

- Nick Arnosti
- Emily Chen
- Ricky Chen
- Paul Cernek
- Guillaume Saulnier Comte
- Alim Virani

FCC & Auctionomics:

- **Melissa Dunford**
- Gary Epstein
- Ulrich Gall
- Karla Hoffman
- **Sasha Javid**
- Evan Kwerel
- Jon Levin
- Rory Molinari
- Brett Tarnutzer
- Venkat Veeramneni
- Karen Wrege

Funding from: Auctionomics; Compute Canada; NSERC Discovery; NSERC E.W.R. Steacie

Unusual Freedom in the Design Process

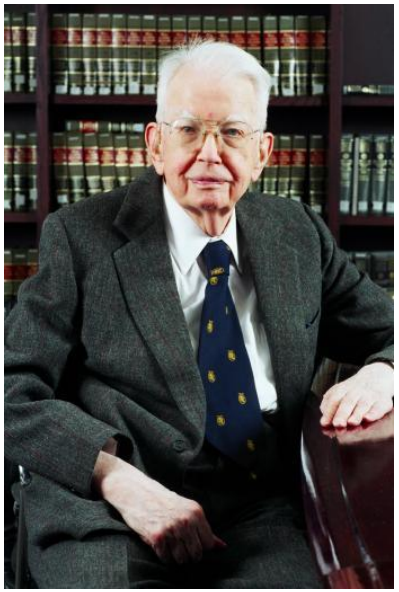
[L-B, Milgrom, Segal, PNAS 2017]

Went beyond just the choice of mechanism to include:

- Participants' **property rights**
- Definition of **goods to be traded**
- **Quantity** of goods to trade
- **Outcomes** the market should seek to achieve
  - efficiency
  - revenue
  - increased competition in the consumer market
  - bidding simplicity for unsophisticated participants

Computational tractability was a first-order concern

Property Rights



- Law was unclear about **broadcasters' property rights**
  - but confiscation would have triggered a long legal process
- Famous argument from Coase: for efficient allocation, need only **clear property rights** and no "frictions"
- Unfortunately, our setting gives rise to a critical friction: **holdout power**
  - wireless companies want to clear many channels' worth of spectrum in large, contiguous geographic areas
  - one channel could threaten to block the whole transaction in exchange for a big payout
  - any efficient market (e.g., VCG) enforces such high payments to each channel; not budget balanced

Defining Property Rights

- This problem is reduced by a redefinition of property rights: stations have a right to keep broadcasting if they don't sell, but **not necessarily on their original channel**
  - Thus, we don't have to buy out a specific set of stations, but rather a sufficient number of them
  - In other words, **stations are made substitutes** for each other, fostering competition

H.R.3630 - Middle Class Tax Relief and Job Creation Act of 2012

112th Congress (2011-2012)

LAW Hide Overview

Sponsor: Rep. Camp, Dave [R-MI-4] (Introduced 12/09/2011)

Committees: House - Ways and Means; Energy and Commerce; Financial Services; Foreign Affairs; Transportation and Infrastructure; Agriculture; Oversight and Government Reform; House Administration; Budget; Natural Resources; Rules; Intelligence (Permanent)

Committee Reports: H. Rept. 112-399 (Conference Report)

Latest Action: 02/22/2012 Became Public Law No: 112-96. (TXT | PDF) (All Actions)

Roll Call Votes: There have been 10 roll call votes

Tracker:

Introduced | Passed House | Passed Senate | Resolving Differences | To President | **Became Law**

More on This Bill

Constitutional Authority Statement

CBO Cost Estimates [3]

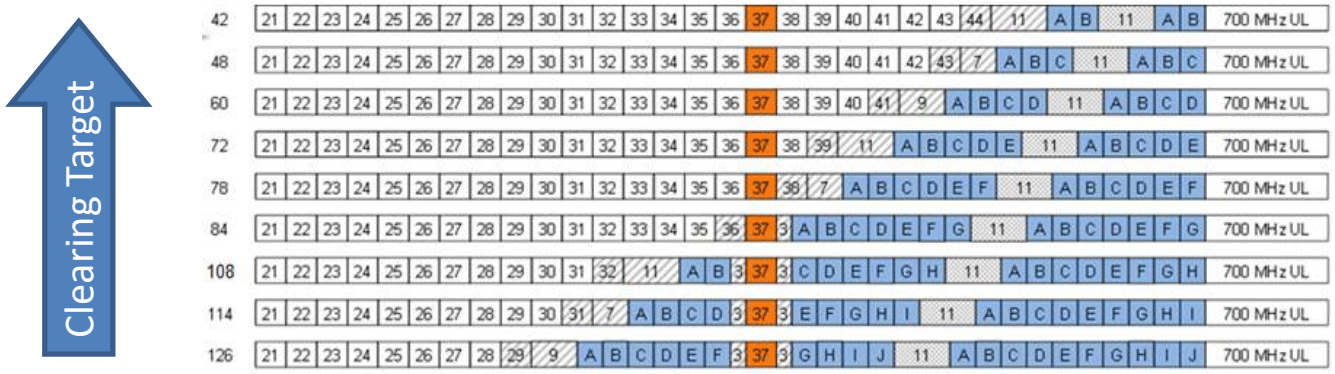
Subject — Policy Area:

Economics and Public Finance

View subjects »

How Much Spectrum to Clear?

The FCC decided to **standardize the amount of spectrum cleared** across the country. How much should this be?



- Standard economics solution (with homogeneous goods): trade the quantity of good for which there's a market clearing price with **supply meeting demand**
- In our setting, **no homogeneous good**, no single price
  - every station's broadcast license covers a different population
  - every wireless license is distinct
  - these two kinds of licenses are different from each other

Externalities

- Economic theory: best to define property rights to ensure that others don't care who wins a good
  - In the incentive auction: assigning a given station to a given channel should **not cause more than minimal interference** (0.5% of population) for any other channel

- But: verifying on the fly **not computationally feasible**
  - quantifying the number of customers affected by interference under a given assignment of channels to stations takes **days of computer time**
  - with 2990 stations needing to be assigned into 29 channels, $29^{2990} \cong 10^{4300}$ possible assignments
    - compare to 10^{80} atoms in the universe!

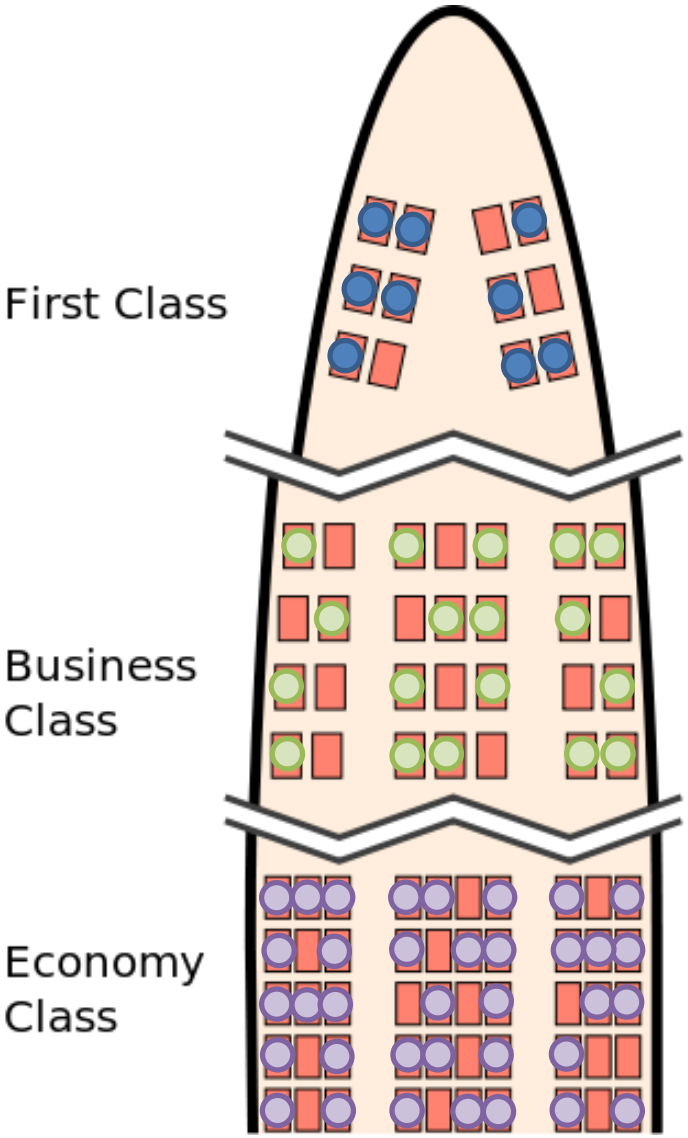
Redefining Harmful Interference

- A station *j* suffers minimal interference if no other **single station** interferes with > 0.5% of *j*'s preauction audience
  - such pairwise constraints can be precomputed
- Even so, the problem of **determining whether there exists *any* channel assignment** for a set of stations is NP-complete (graph coloring)
  - thus, worst-case running time must scale exponentially with number of stations (unless P = NP)
  - typically possible to do better in practice, but it's not easy
- We cannot expect a decentralized process to solve an NP-complete problem tractably
  - would imply an efficient distributed algorithm
  - so, **there's a role for a central authority** like the FCC and for careful market design

A Heuristic Clock Auction Alternative

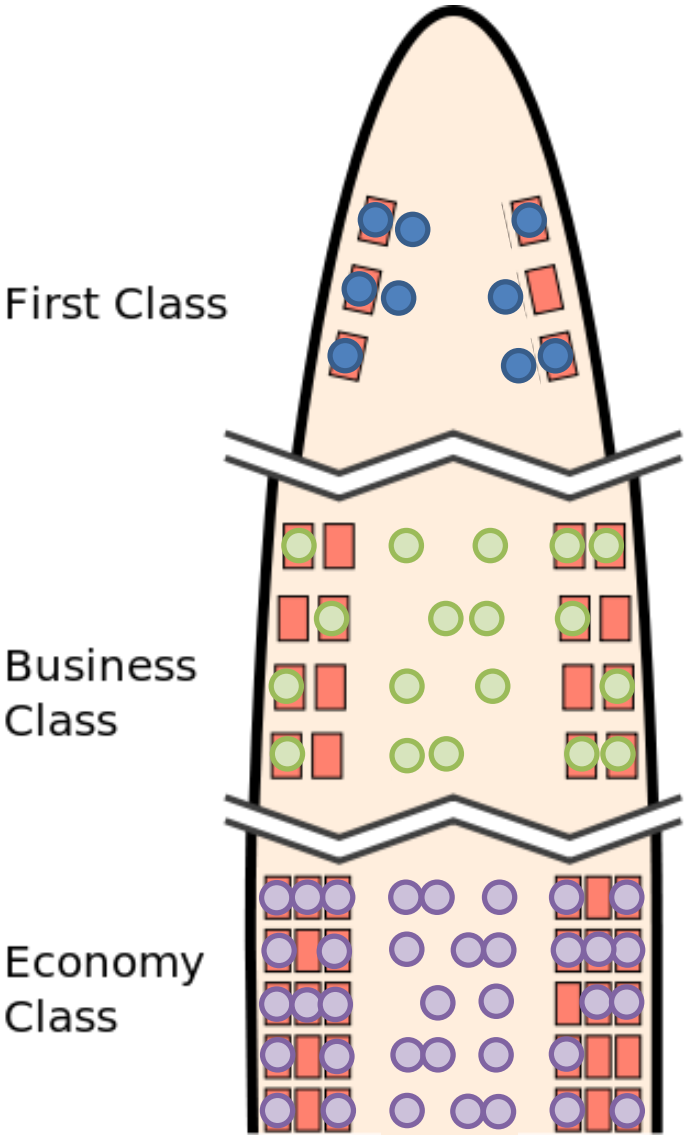
- **Forward (ascending-price) auction** for telecom firms
  - prices in each region increase while demand exceeds supply
- **Reverse (descending-price) auction** for broadcasters
  - prices offered for stations decreases while supply exceeds demand
- **When auctions terminate,** ensure revenue target is met
  - if not, grow the size of the reduced band (i.e., clear less spectrum); auctions continue

How Does the Reverse Auction Work?



- Let's consider the example of **airline overbooking**, where passengers either fly in their assigned cabin or are compensated to give up their seat
- Thus, the **feasibility constraint** is (# passengers in cabin) $\leq$ (# seats)
- We'll use a **descending clock** auction to set compensations
- Let's start with a plane big enough to hold everyone…

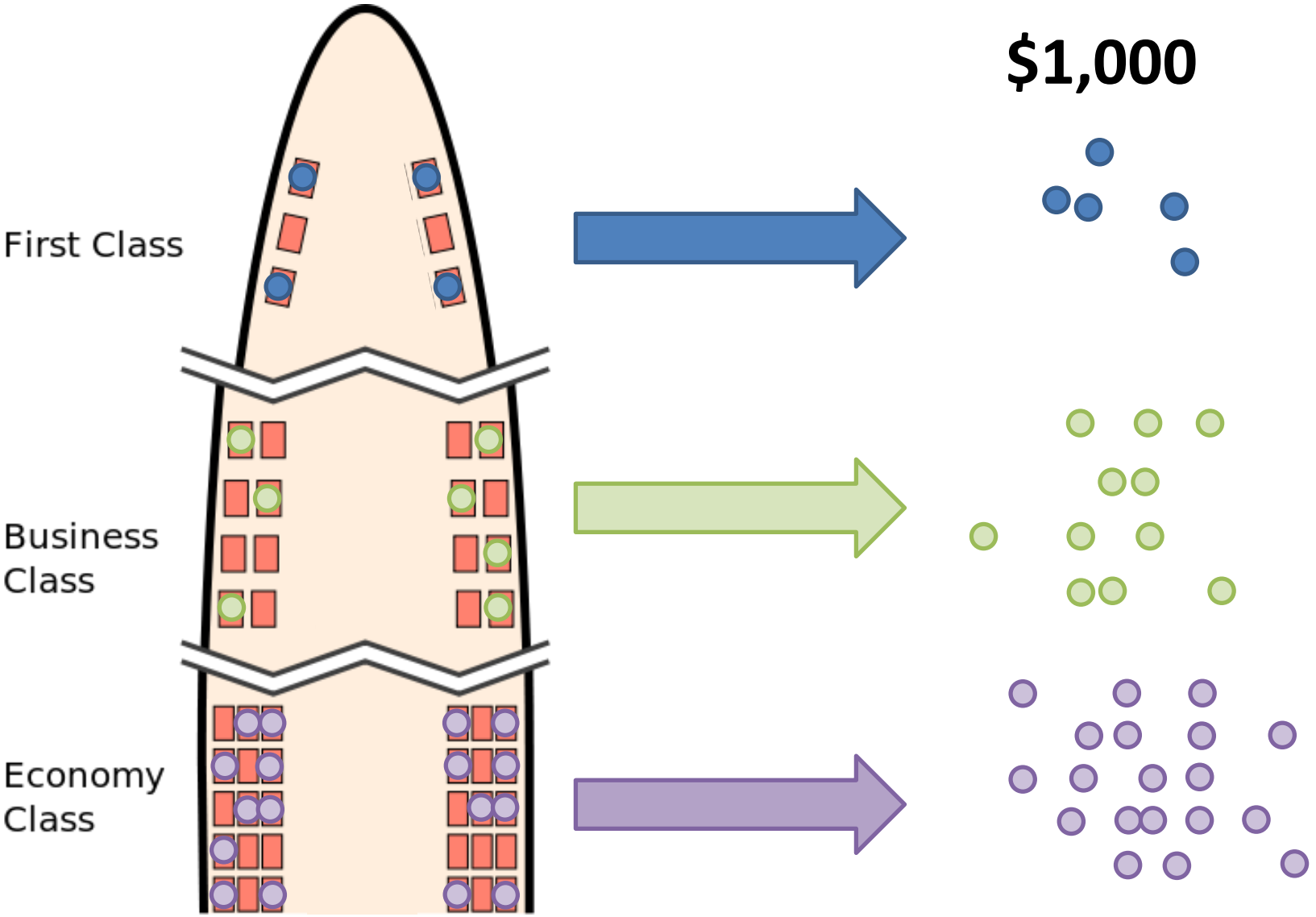
Reverse Auction: Descending Clock



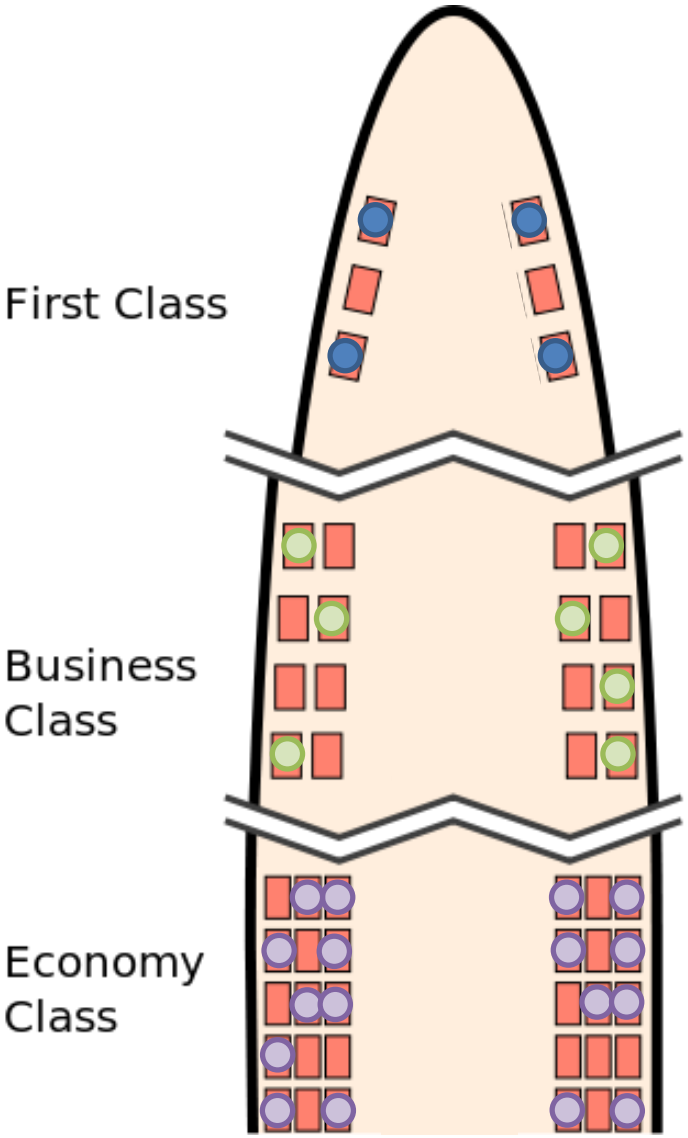
The airline substitutes a smaller plane and offers compensation

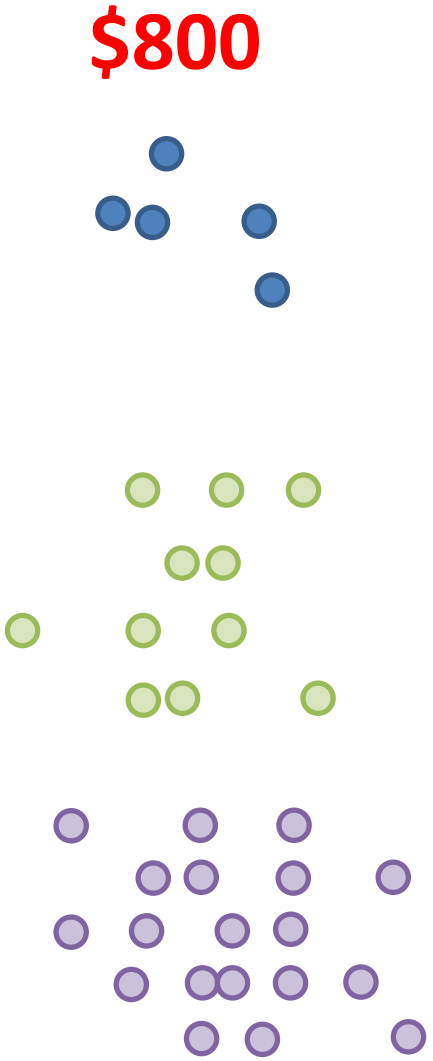
$1,000

Reverse Auction: Descending Clock

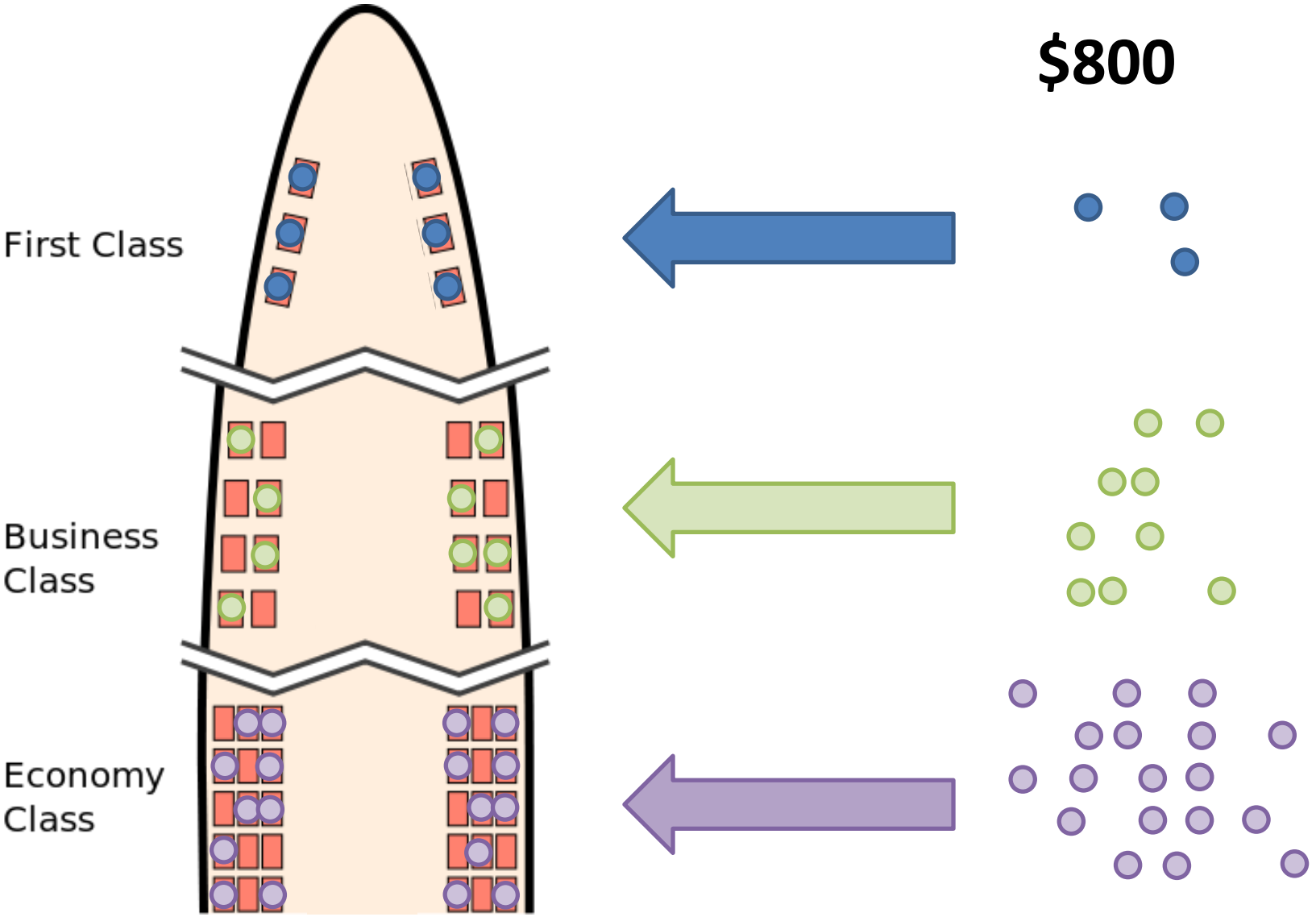


Reverse Auction: Descending Clock

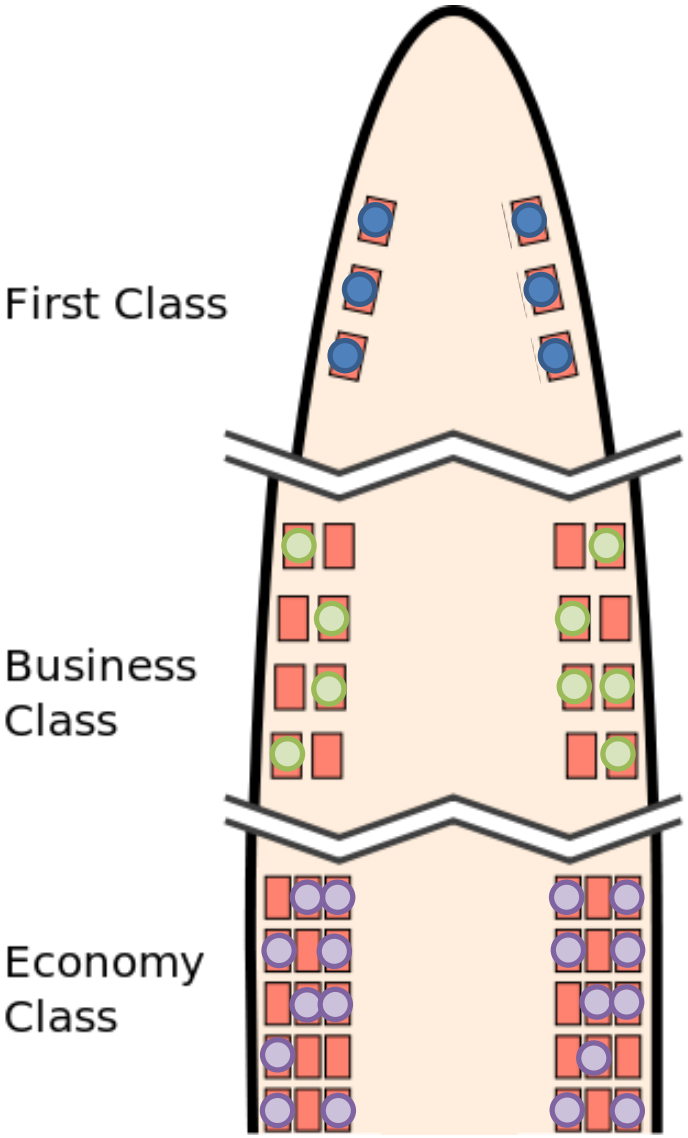


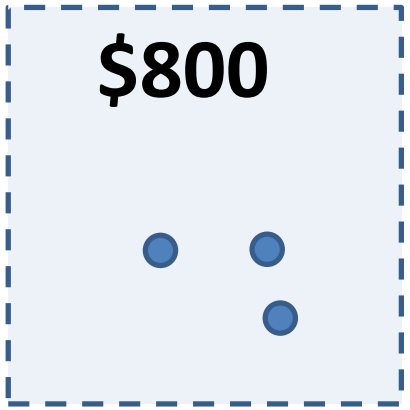


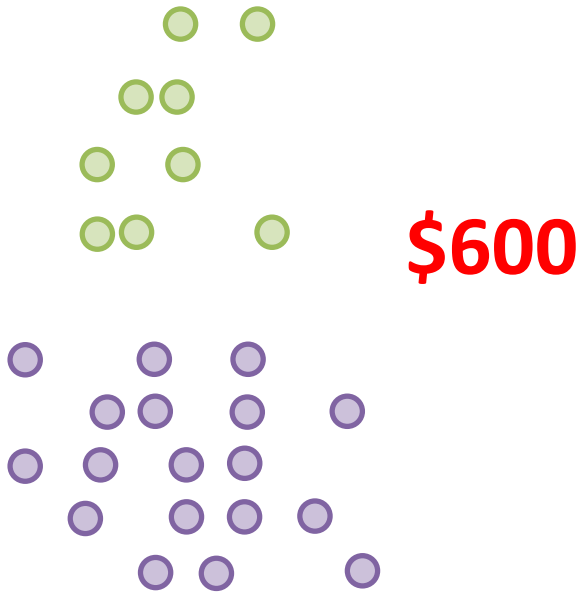
Reverse Auction: Descending Clock



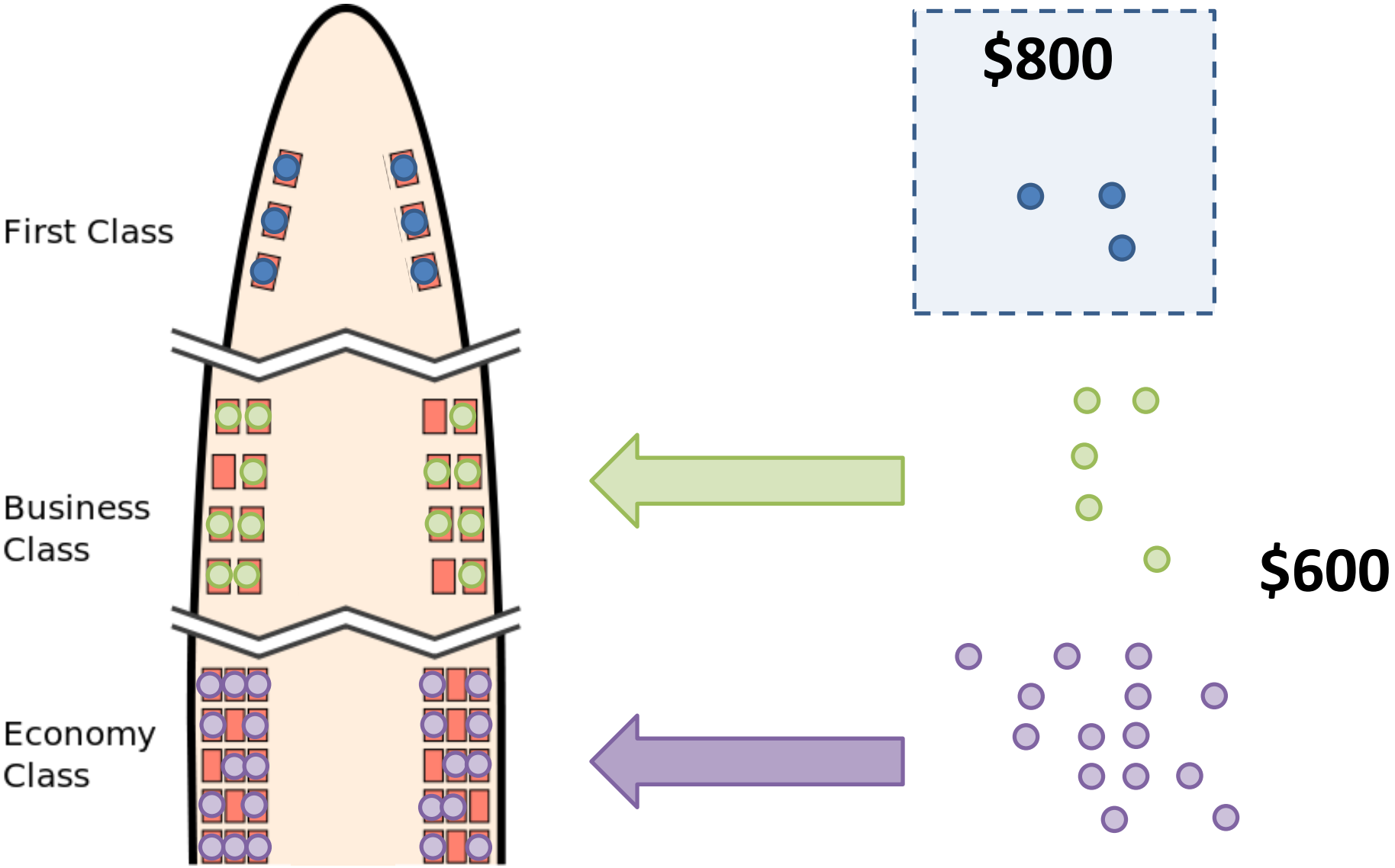
Reverse Auction: Descending Clock



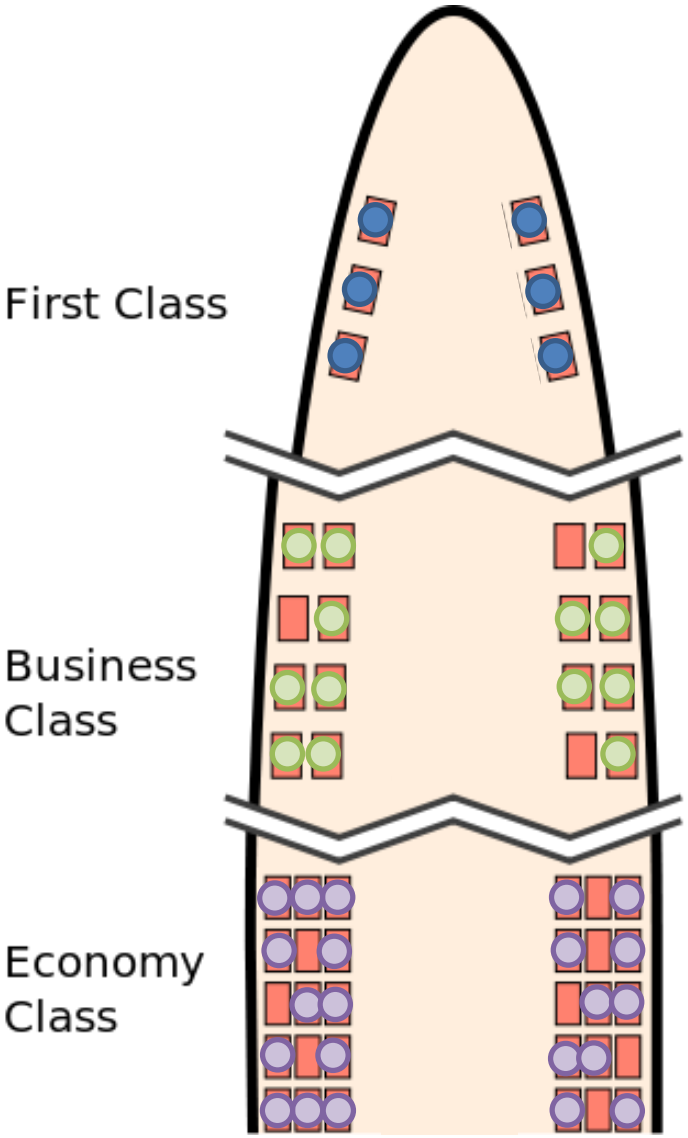


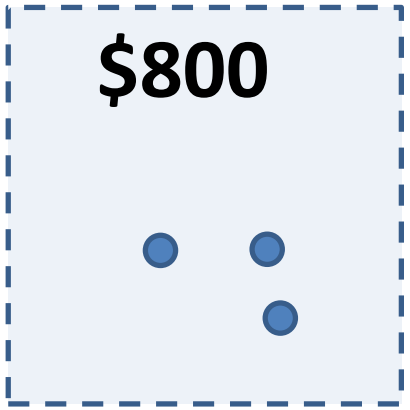


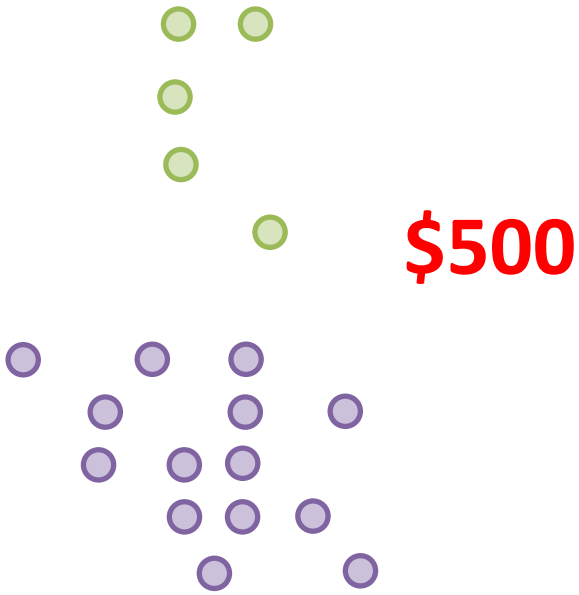
Reverse Auction: Descending Clock



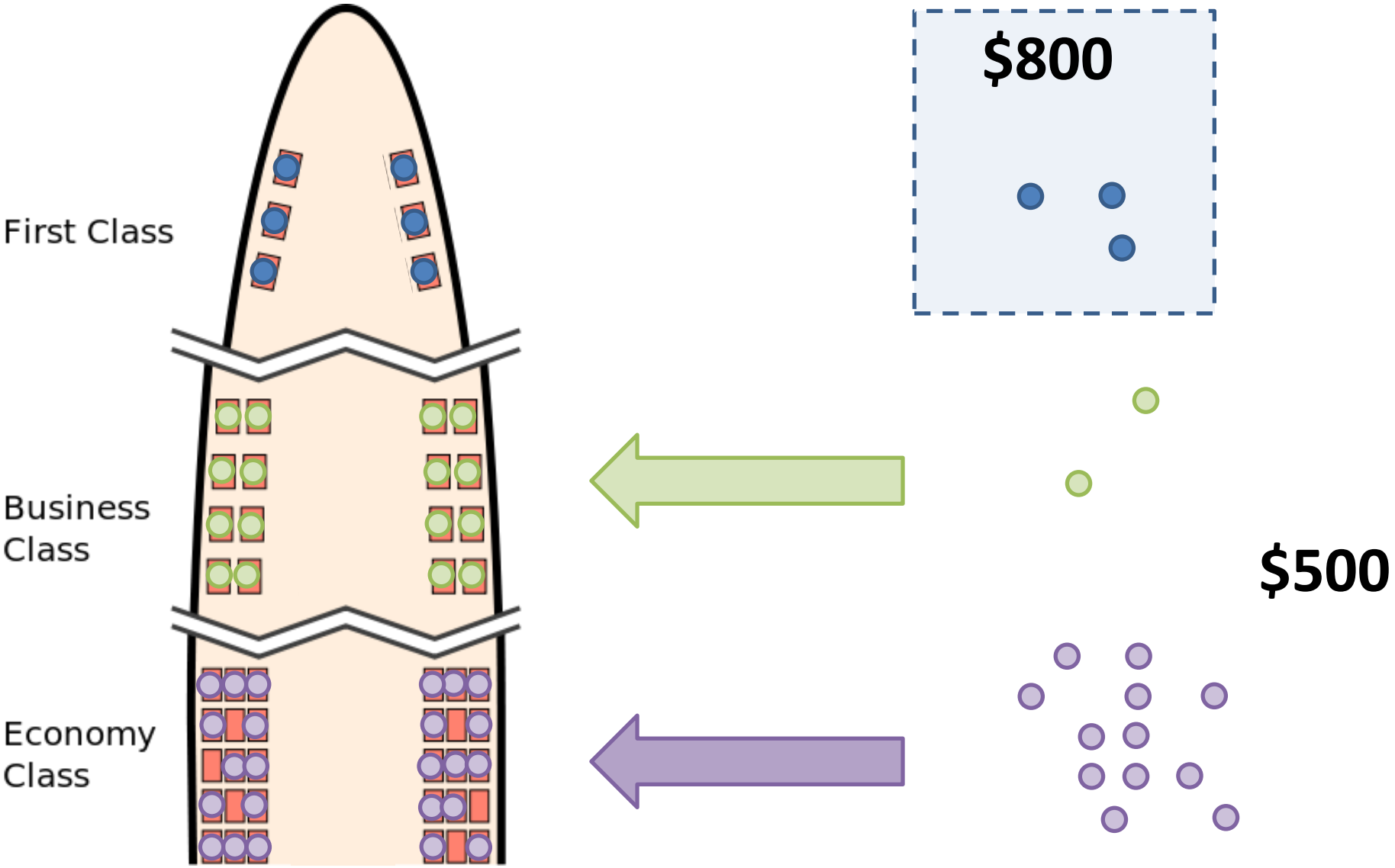
Reverse Auction: Descending Clock



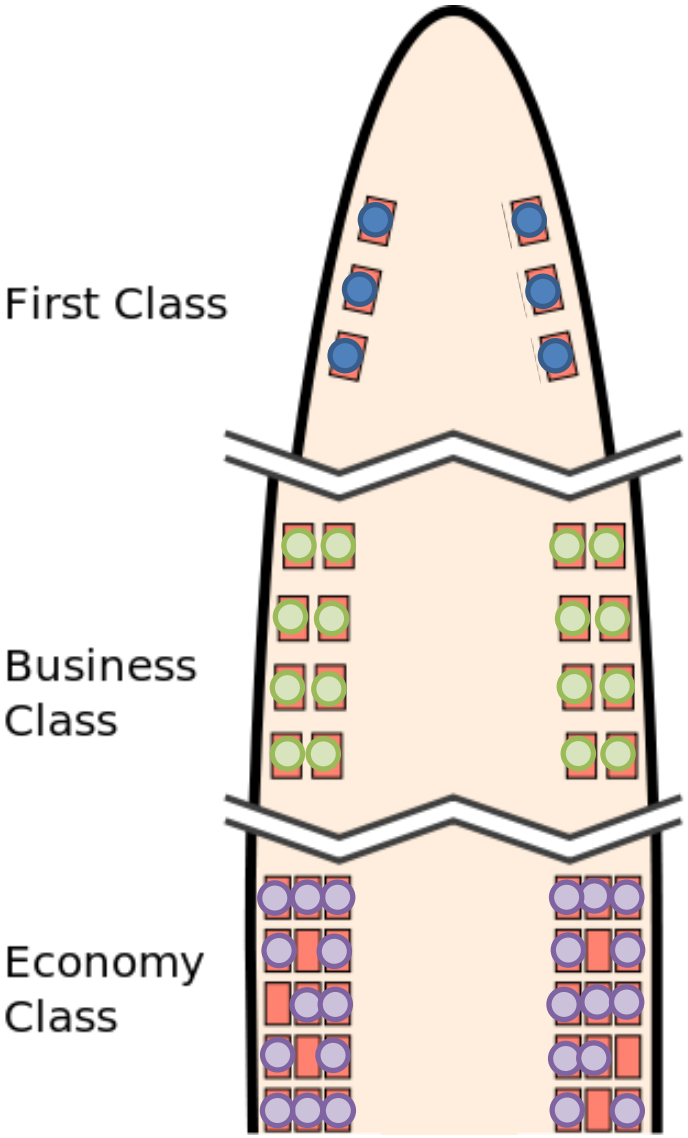


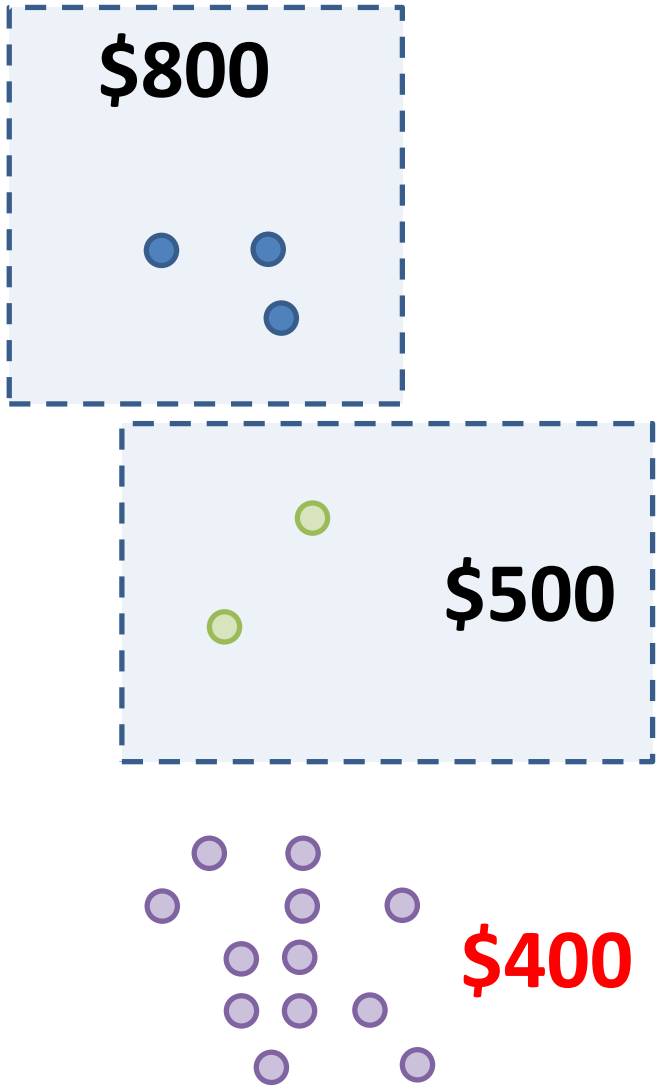


Reverse Auction: Descending Clock

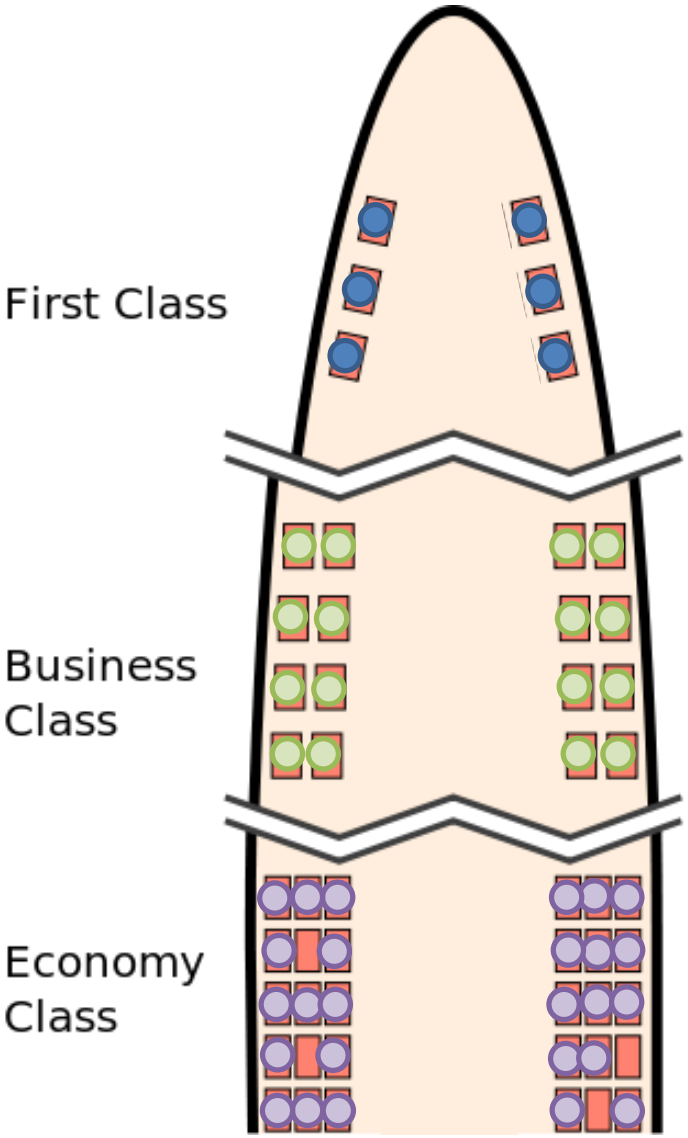


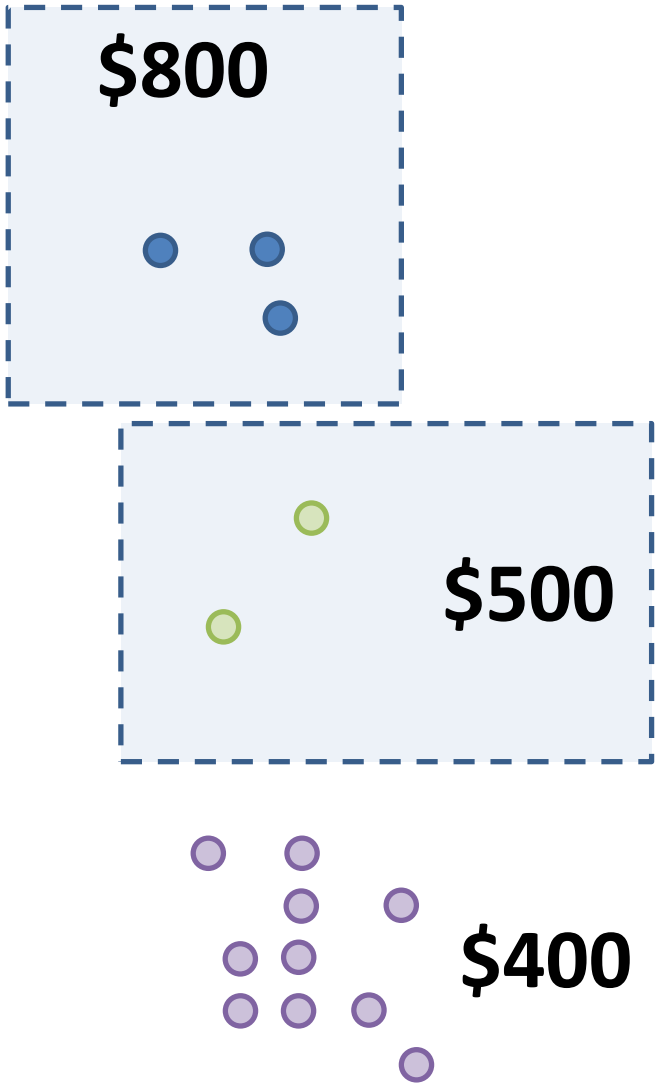
Reverse Auction: Descending Clock



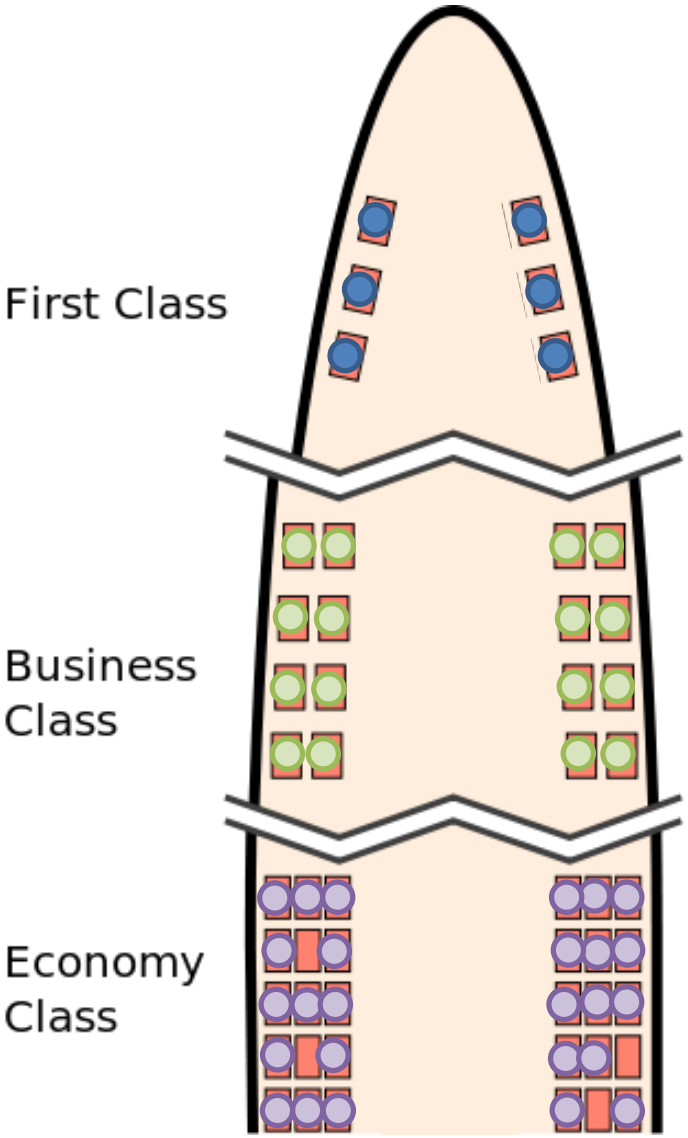


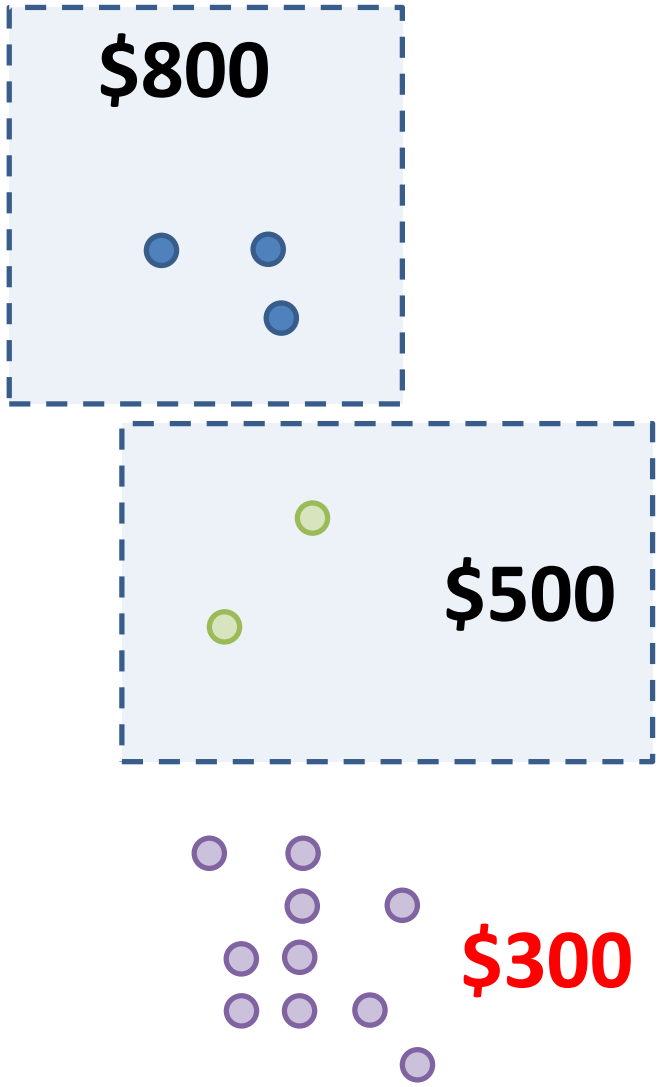
Reverse Auction: Descending Clock



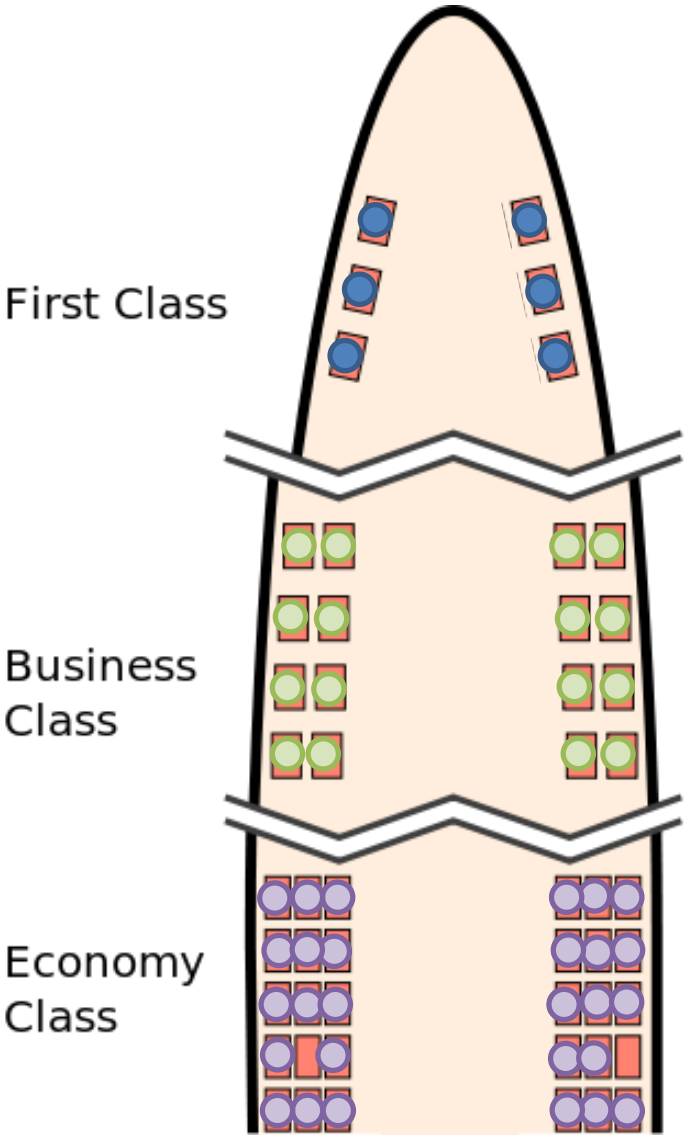


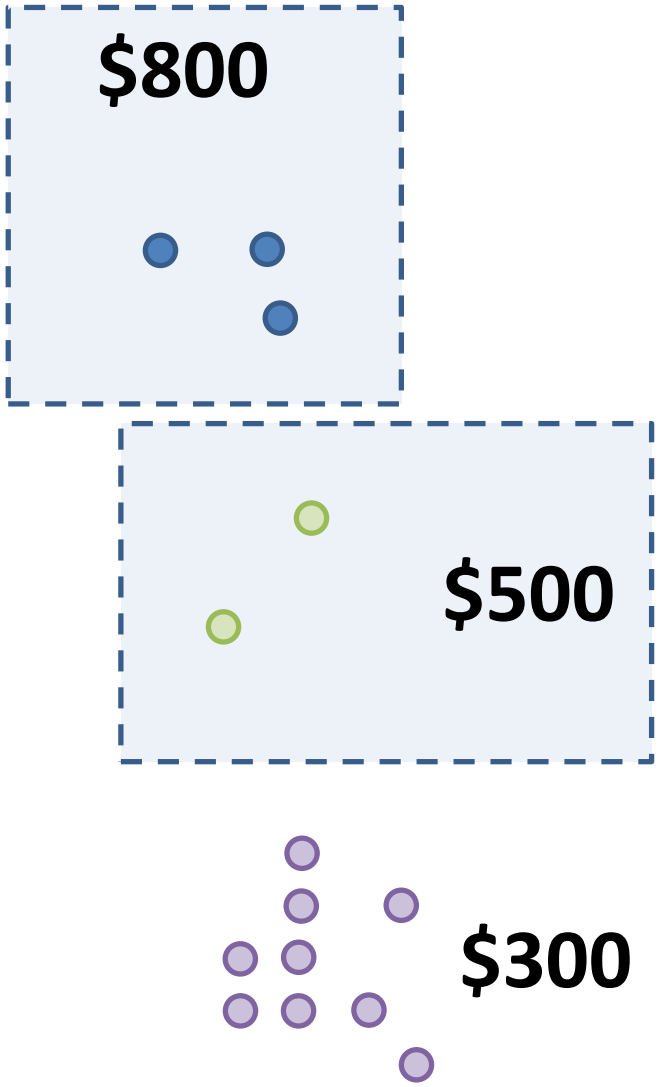
Reverse Auction: Descending Clock



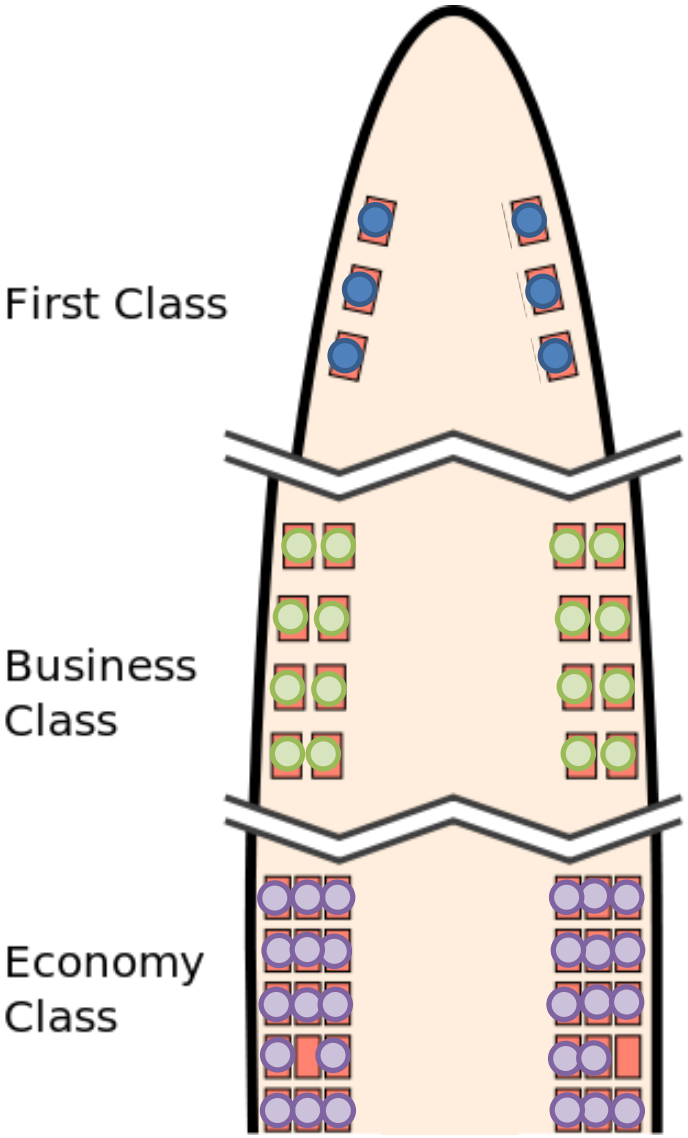


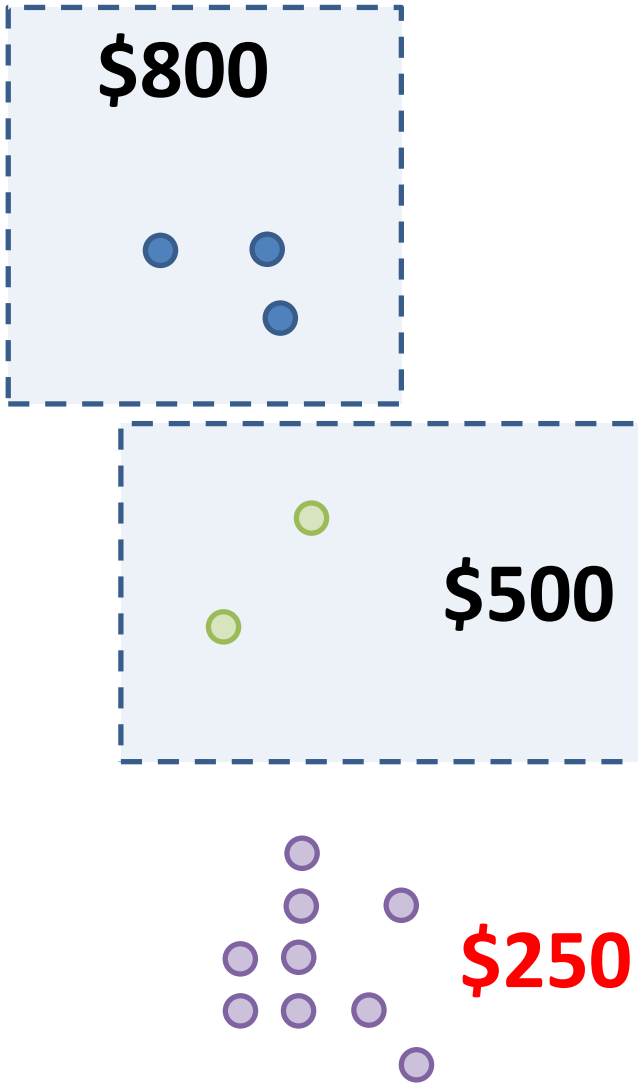
Reverse Auction: Descending Clock



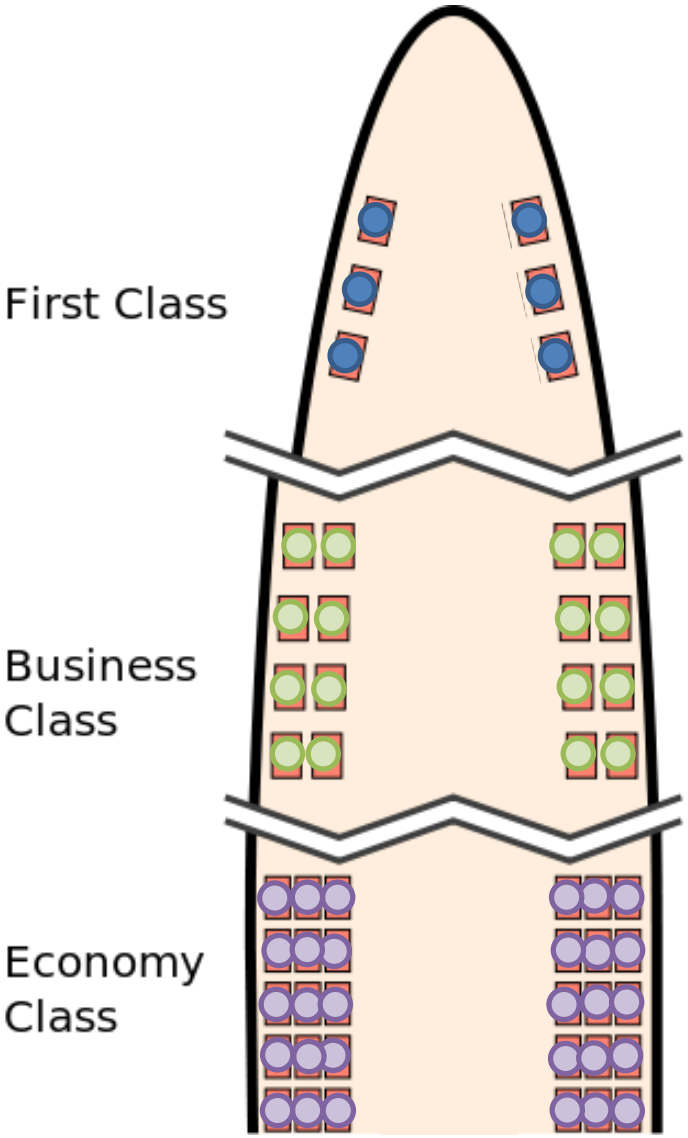


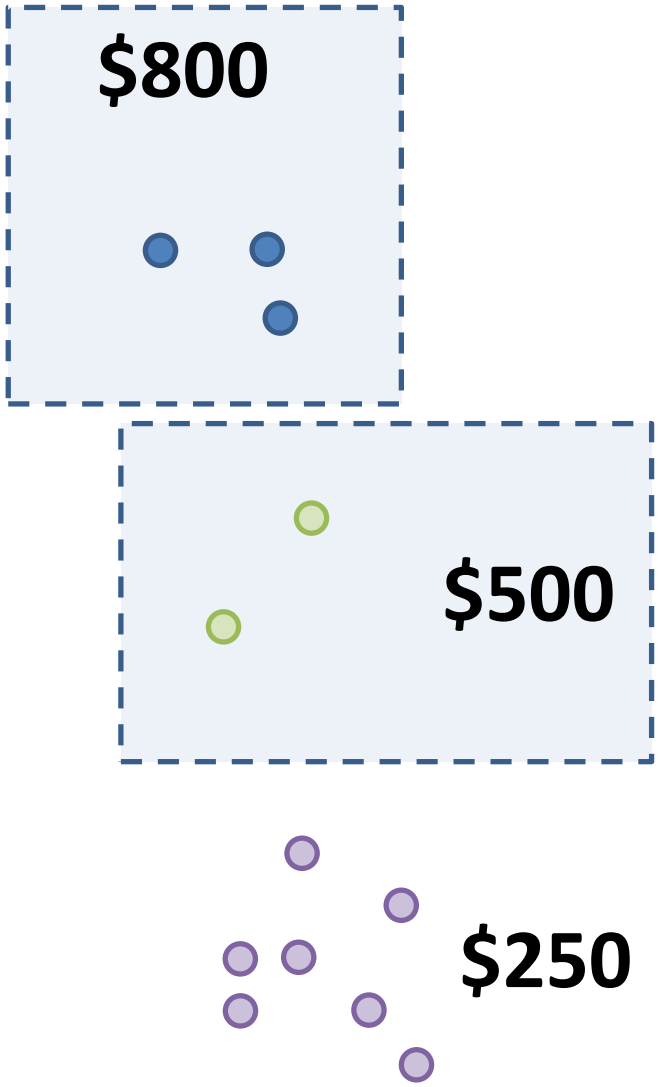
Reverse Auction: Descending Clock



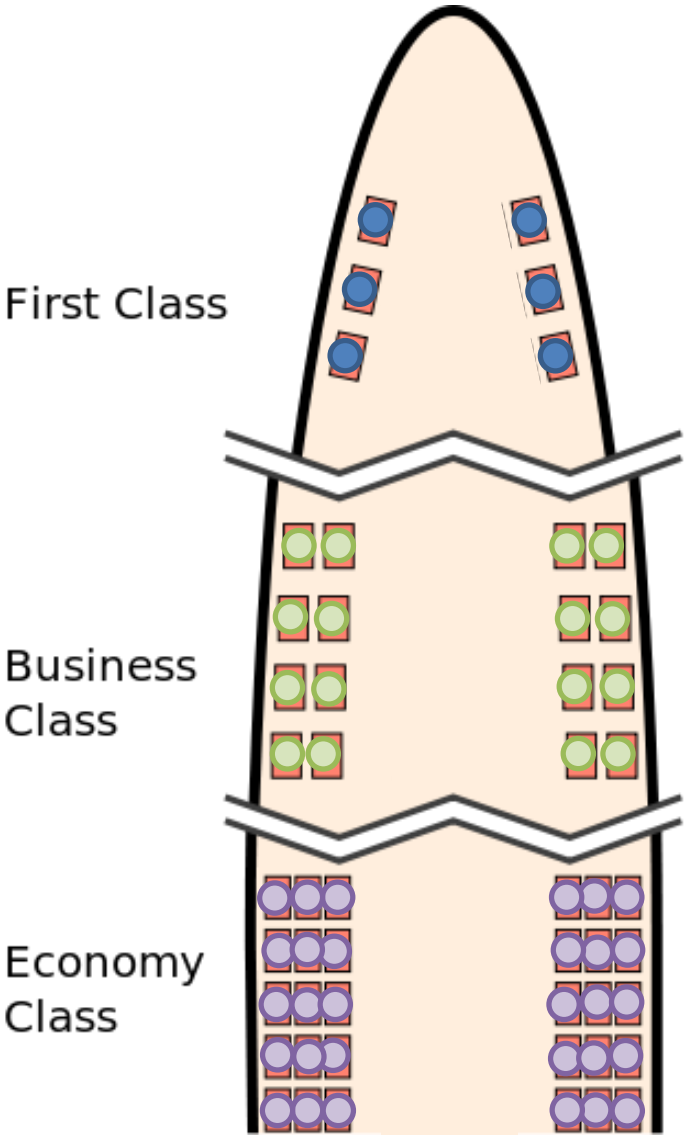


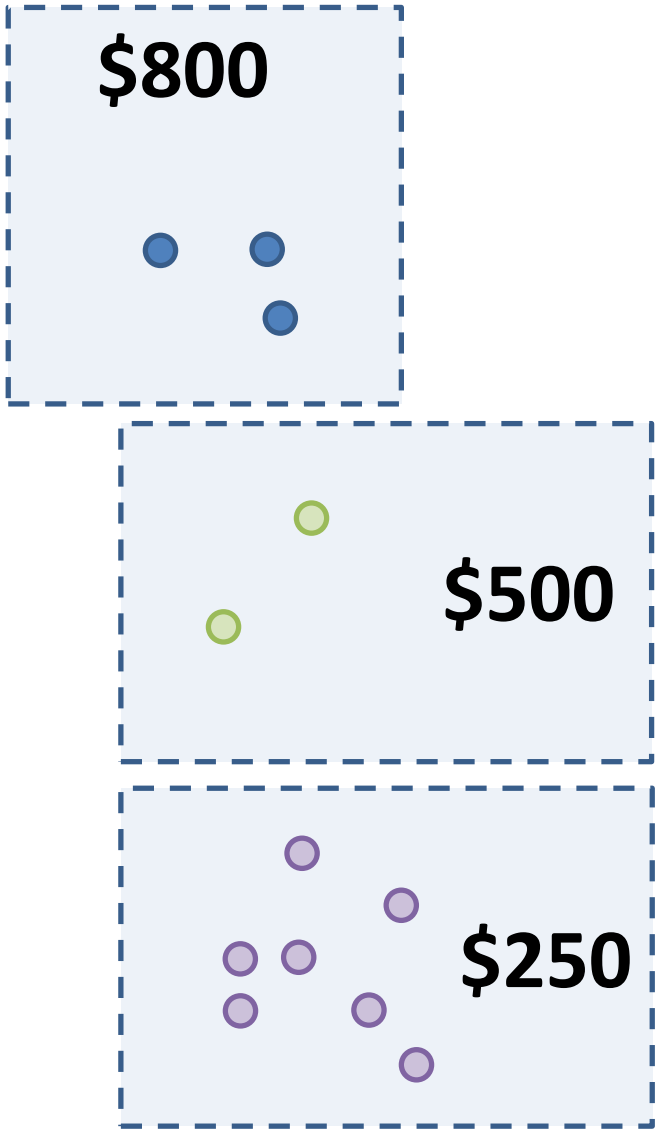
Reverse Auction: Descending Clock



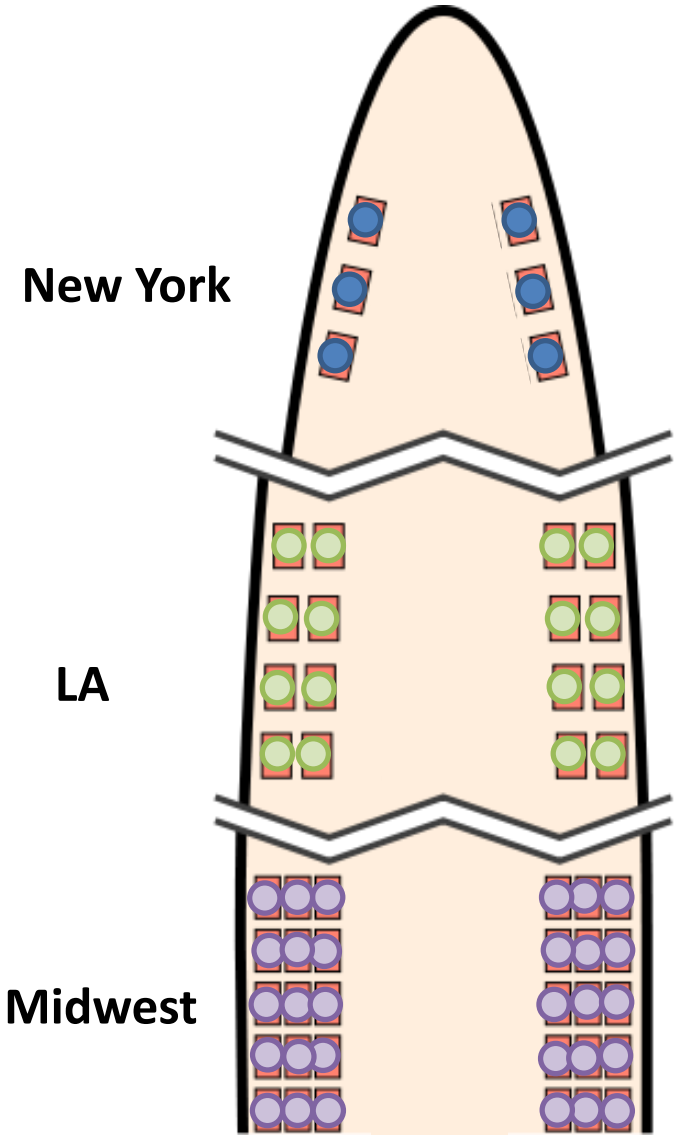


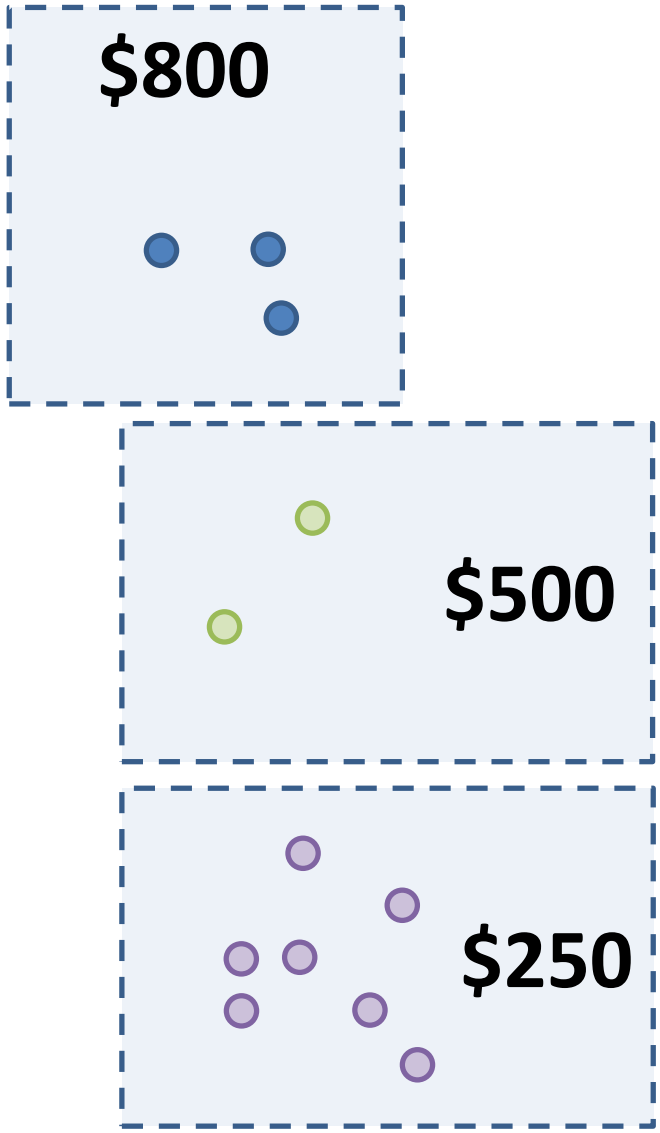
Reverse Auction: Descending Clock



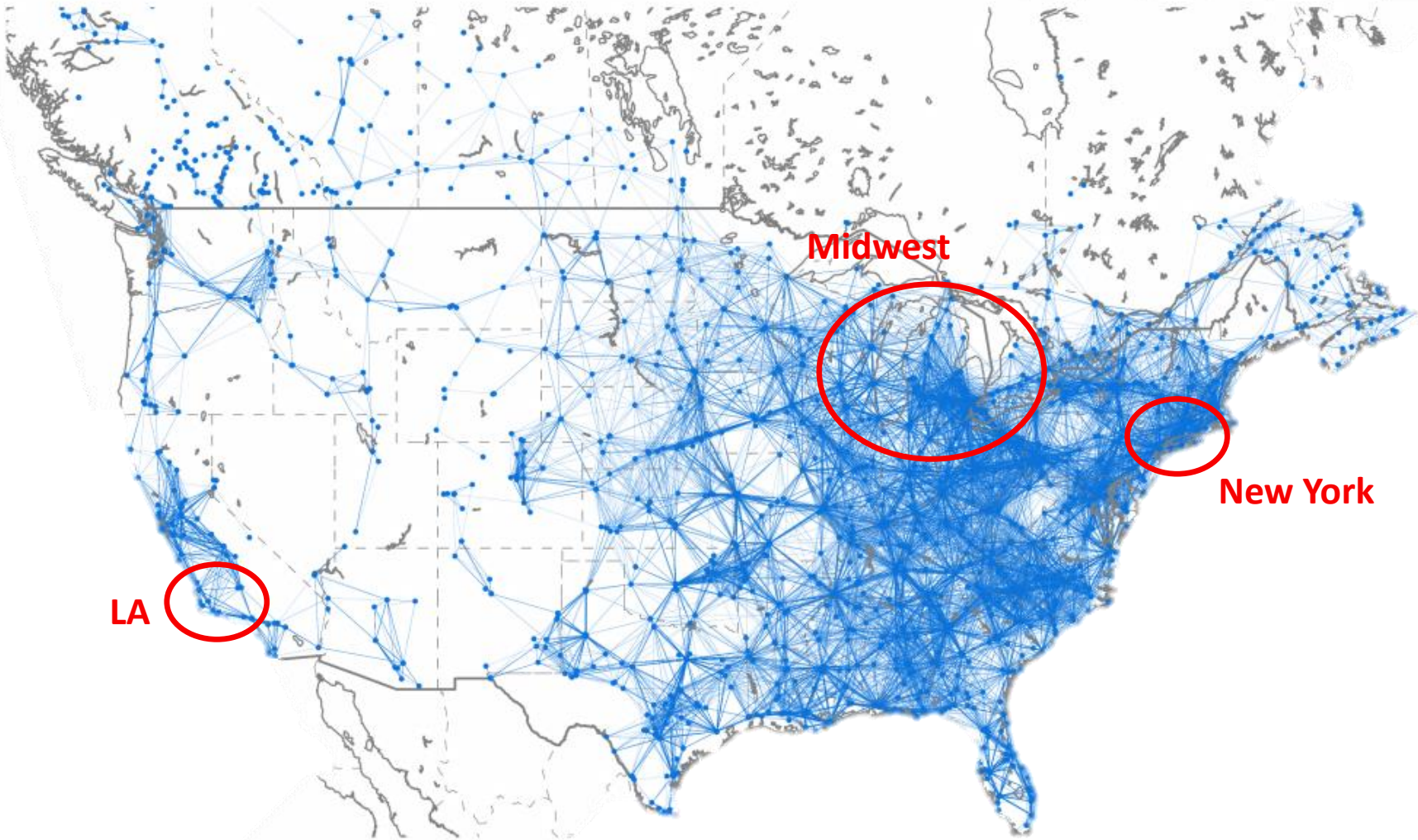


Reverse Auction: Descending Clock





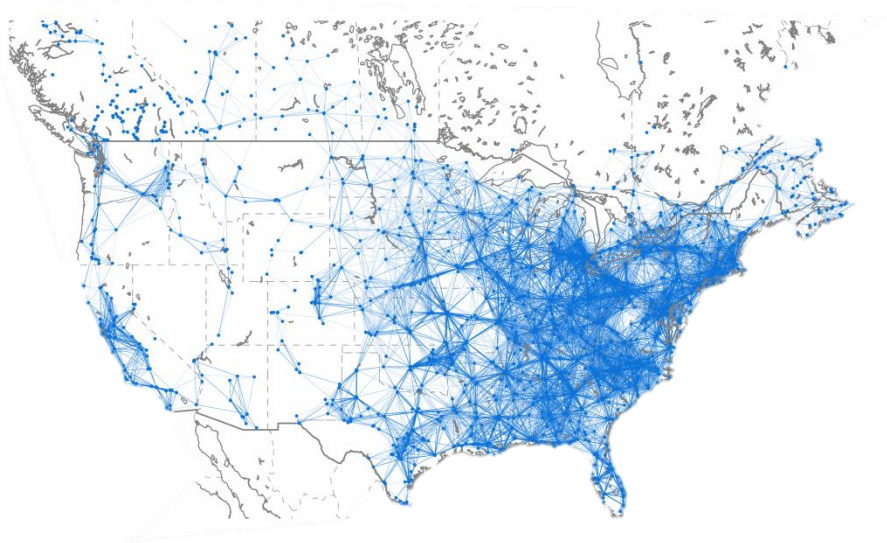
Real Constraints are Highly Complex



- The **feasibility constraints** are not uniform
  - nearby stations can **freeze at different times**

Feasibility Testing

- Basis of “frozen test”: ~100K per auction; ~20K nontrivial
- A hard **graph-colouring** problem
  - 2990 stations (nodes)
  - 2.7 million interference constraints (channel-specific interference)
  - Initial skepticism about whether this problem could be solved exactly at a national scale
  - We did it via “deep optimization” [Newman, Frechette, L-B, CACM 2017]

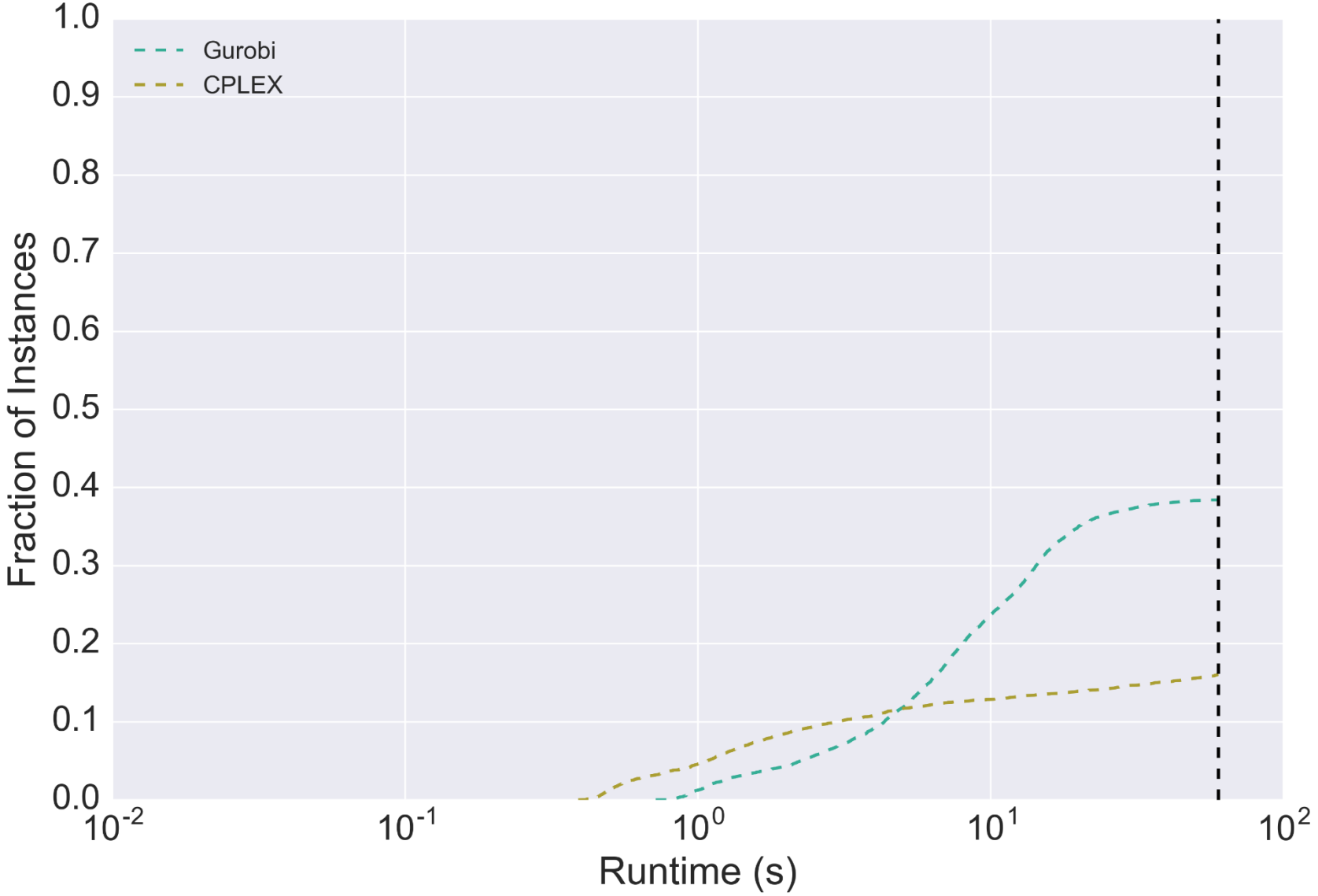


- What if we **can’t solve an instance**?
  - Needed a minimum of two price decrements per 8h business day
    - each feasibility check was allowed a **maximum of one minute**
  - Treat **unsolved problems as infeasible**
    - raises costs slightly, but doesn’t hurt incentives
    - contrast with VCG, which can’t gracefully degrade

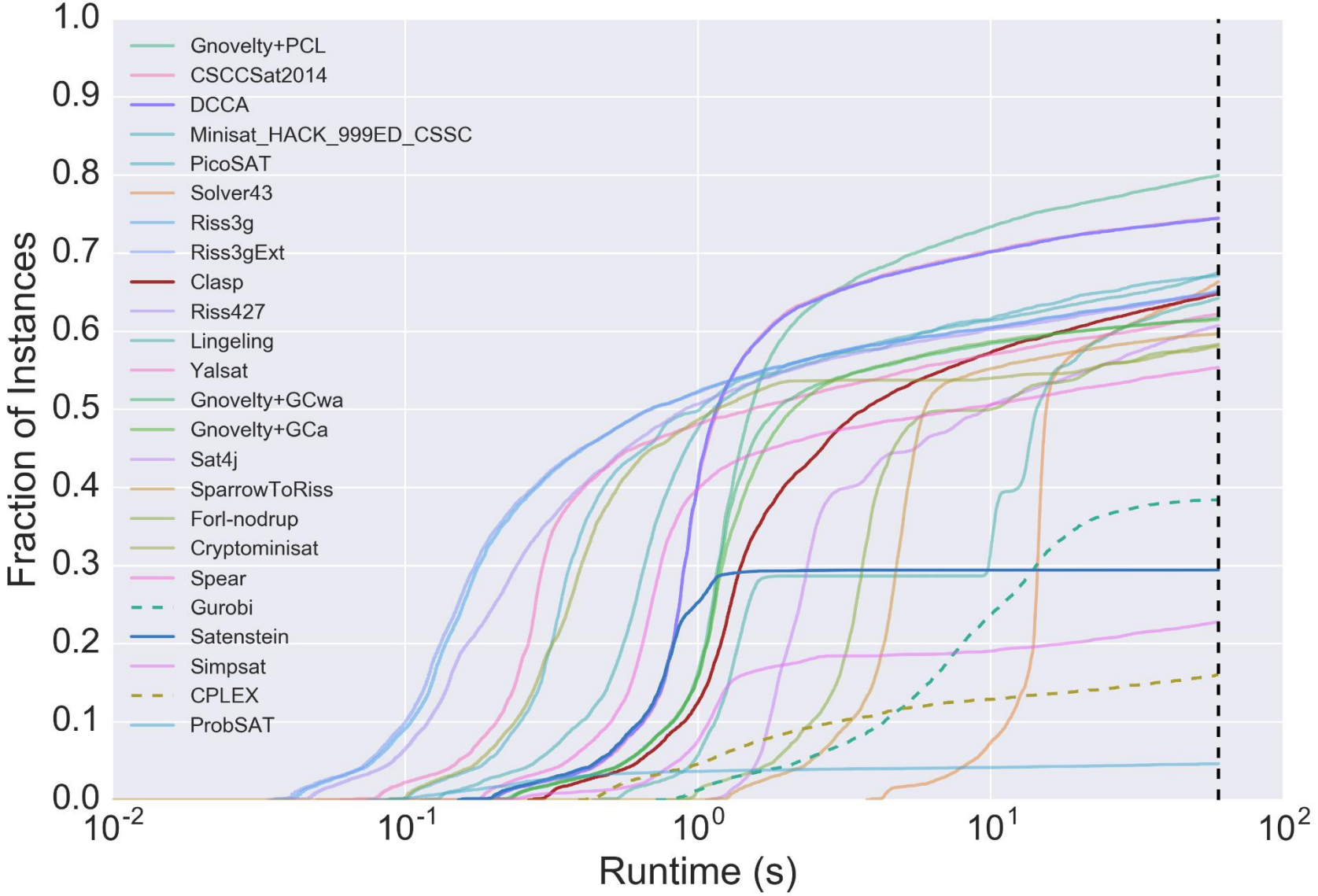
Building (& Evaluating) a Feasibility Tester

- Our original analysis used proprietary data from the FCC
- Evaluation here is based on new data gathered from a full **reverse auction simulator** (UHF; VHF) we wrote ourselves
- Simulation **assumptions**:
  - 84 MHz clearing target
  - valuations generated by sampling from a model due to Doraszelski, Seim, Sinkinson and Wang [2016]
  - stations participated when their private value for continuing to broadcast was smaller than their opening offer for going off-air
  - 1 min timeout given to SATFC
- 20 simulated **auctions** $\Rightarrow$ 60,057 **instances**
  - 2,711 – 3,285 instances per auction
    - all not solvable by directly augmenting the previous solution
    - about 3% of the problems encountered in full simulations
- Our goal: solve problems within a **one-minute cutoff**

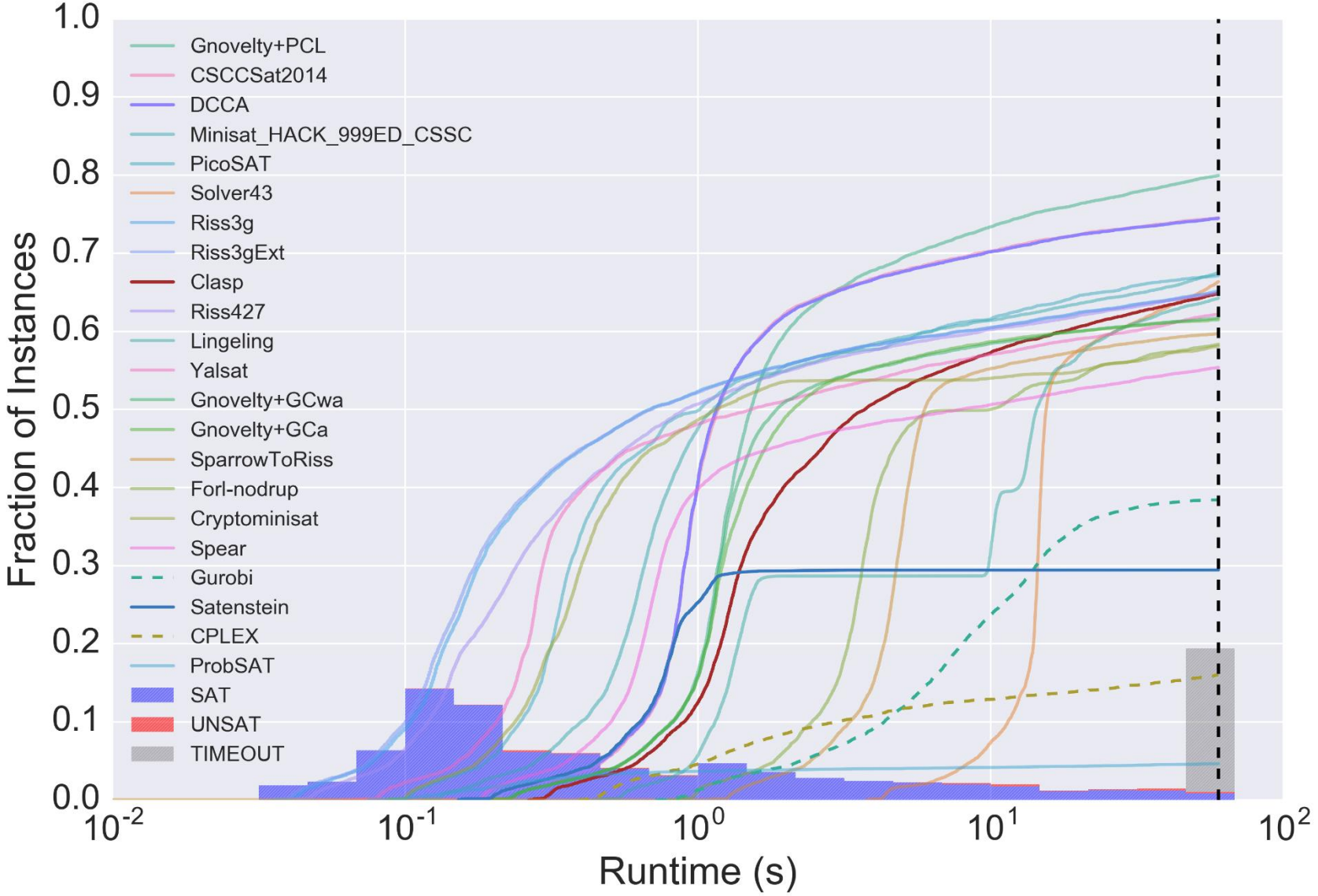
Feasibility Testing via MIP Encoding



Feasibility Testing via SAT Encoding

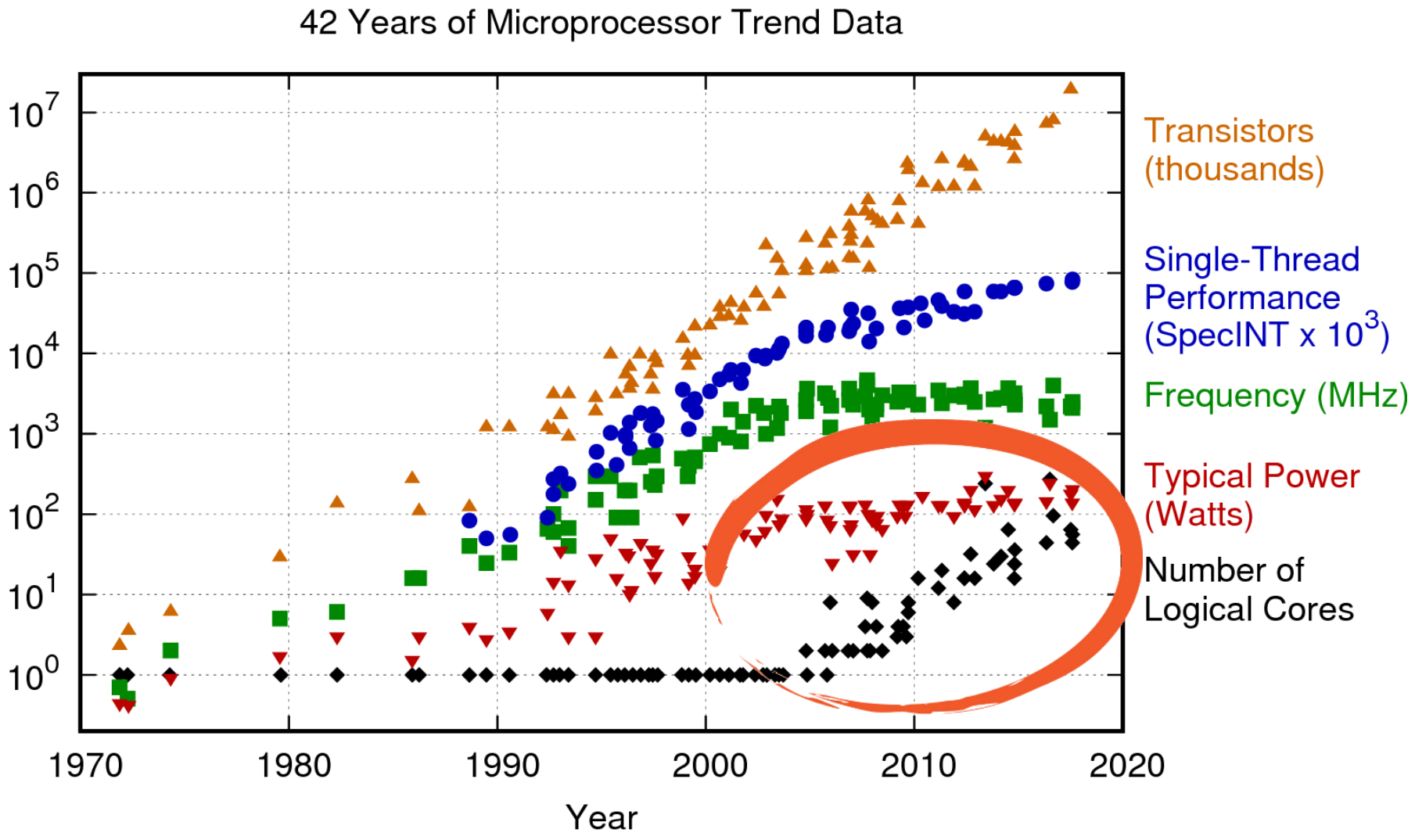


Feasibility Testing via SAT Encoding



Continued, huge increases in compute power

Approaches that might have seemed crazy even in 2005 ***make a lot more sense now…***



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp Taken from https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Deep Optimization

Machine learning

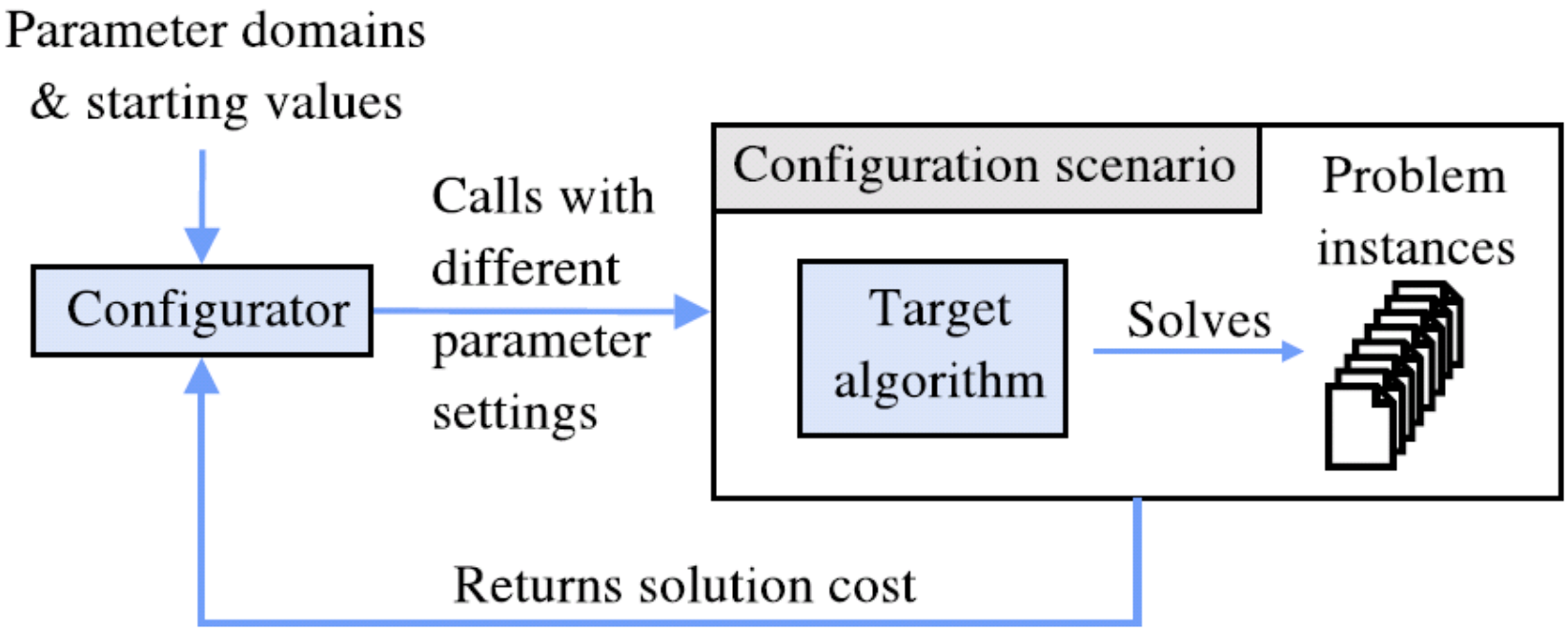
- **Classical approach**
  - Features based on expert insight
  - Model family selected by hand
  - Manual tuning of hyperparameters
- **Deep learning**
  - Very highly parameterized models, using expert knowledge to identify appropriate invariances and model biases (e.g., convolutional structure)
    - "deep": many layers of nodes, each depending on the last
  - Use lots of data (plus "dropout" regularization) to avoid overfitting
  - Computationally intensive search replaces human design

Discrete Optimization

- **Classical approach**
  - Expert designs a heuristic algorithm
  - Iteratively conducts small experiments to improve the design
- **Deep optimization**
  - Very highly parameterized algorithms express a combinatorial space of heuristic design choices that make sense to an expert
    - "deep": many layers of parameters, each depending on the last
  - Use lots of data to characterize the distribution of interest
  - Computationally intensive search replaces human design

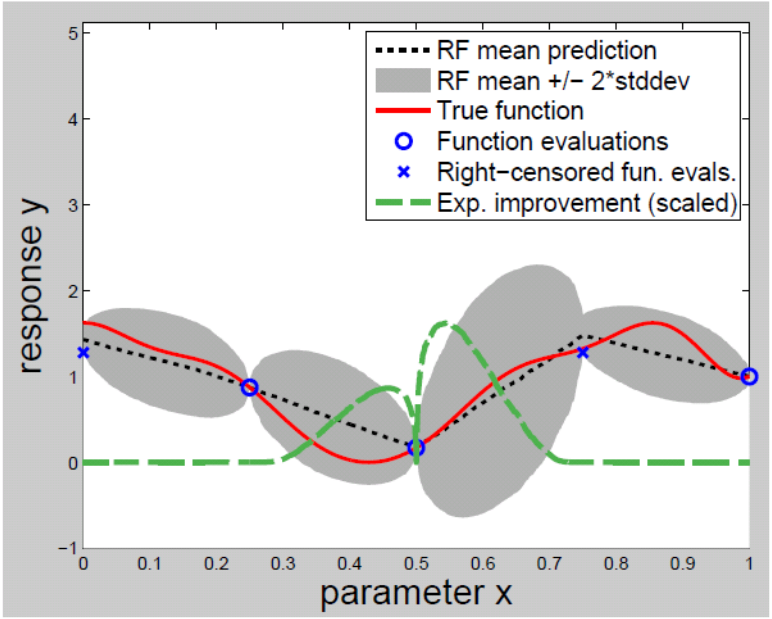
Algorithm Configuration

- Deep optimization: use automated methods to choose algorithm designs from a **highly parameterized space**
  - which branching heuristic, variable ordering, preprocessing strategy, clause learning technique, …
- Such automated methods are called **algorithm configurators**



Sequential Model-based Algorithm Configuration (SMAC)

[Hutter, Hoos & L-B; 2011]



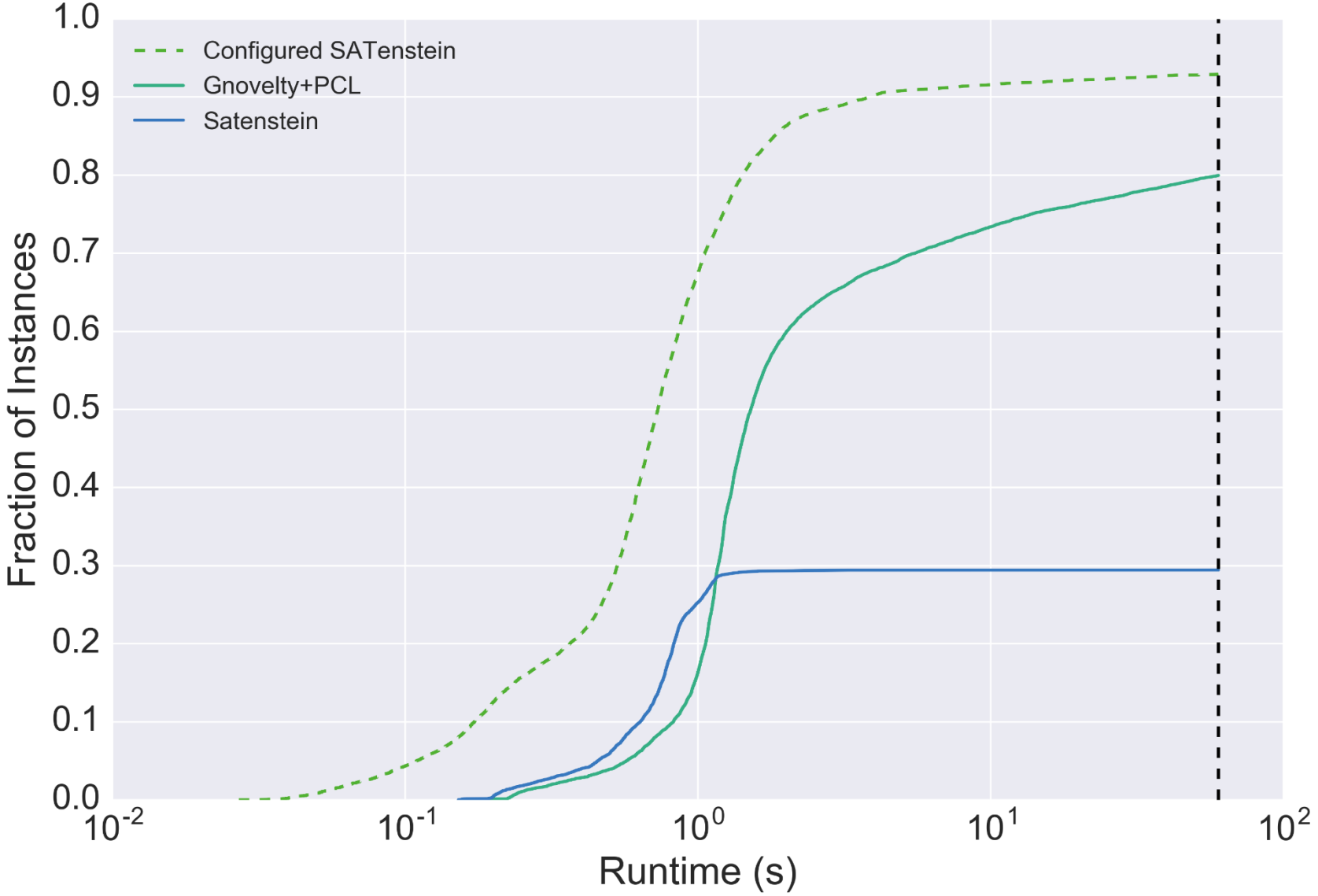
Initialize with a single run for the default configuration

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.

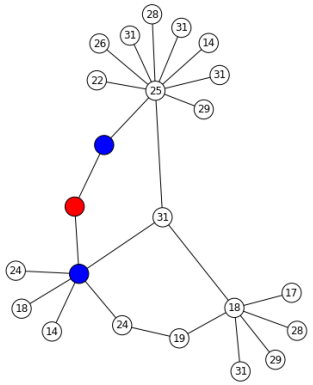
until *time budget exhausted*

Best Configured Solver

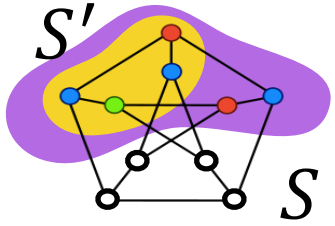


Problem-Specific Speedups

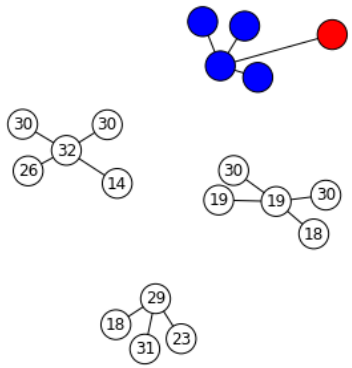
- All problems ask whether it's feasible to add one station to **an existing set known to be feasible**
  - local search: initialize at the known solution
  - incomplete approach: fix channels for non-neighboring stations, solve the remaining problem optimally



- **Containment caching:** search for supersets of the given station set that have been proven feasible in past runs



- **Decompose** the induced constraint graph



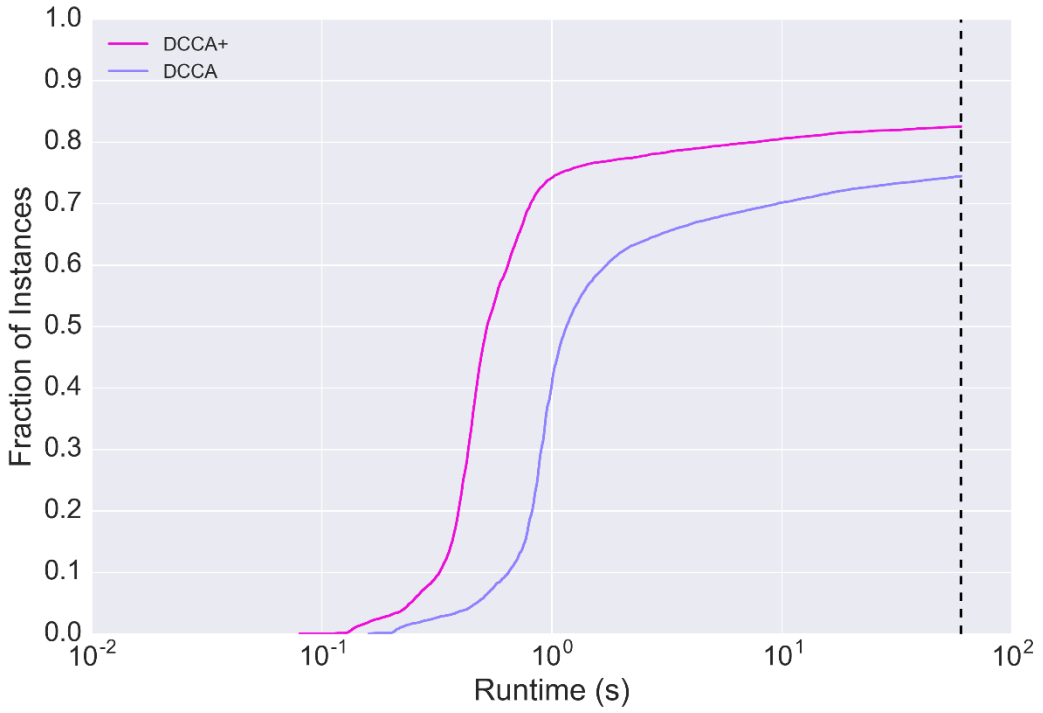
- Identify & remove **underconstrained stations**

(skip the details)

Reusing Previous Solutions (I)

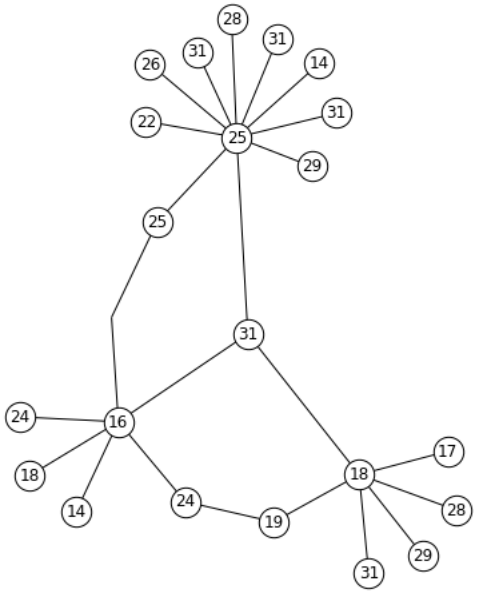
Now I'll discuss some problem-specific heuristics that we can expose as additional parameters…

- Problems arise sequentially by **adding a single station** to a SAT problem
  - we always have **a solution** to the previous problem
- Local search solvers can be **initialized** with the previous solution

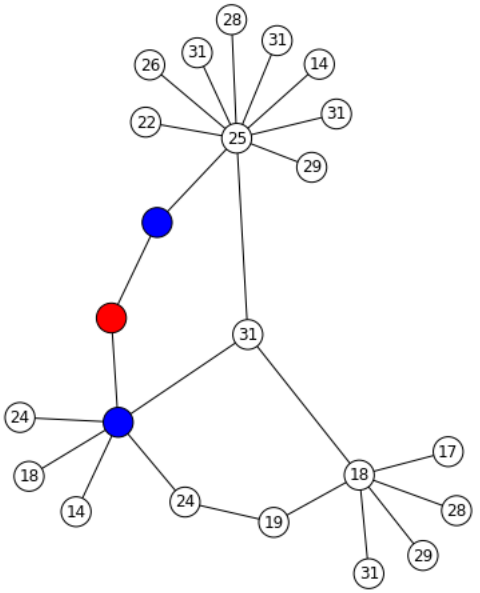


Reusing Previous Solutions (II)

- When **adding a new station** we can try to reuse the previous solution
  - Fix all non-neighboring stations to their channels from previous solution
  - Just a heuristic – cannot prove UNSAT



- We can slowly **expand** the problem (e.g. neighbors of neighbors)

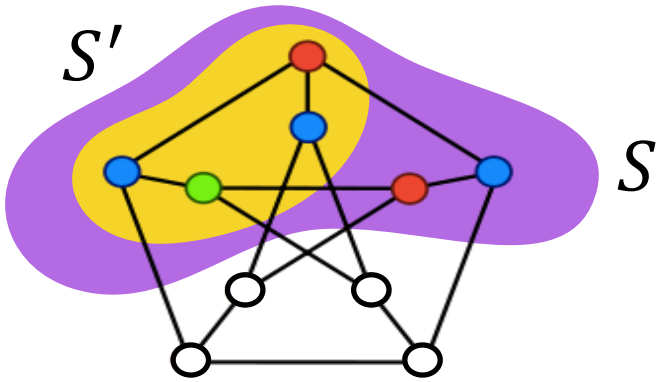


Caching

- 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**

Supersets and Subsets

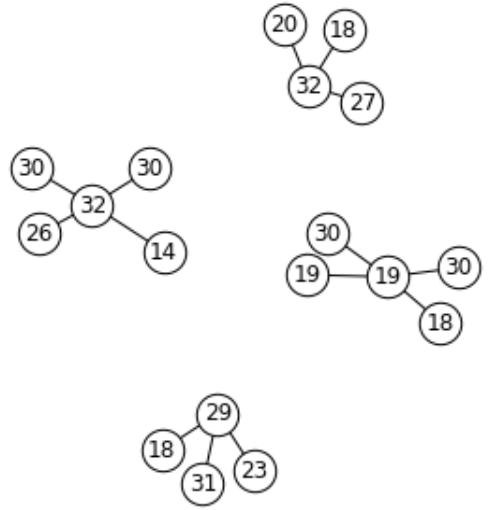
- Observation: if station set S is repackable, **so is every station set $S' \subseteq S$**
  - Useful because there are exponentially many such sets S'
  - Likewise, if S is not repackable, neither is any $S'' \supseteq S$



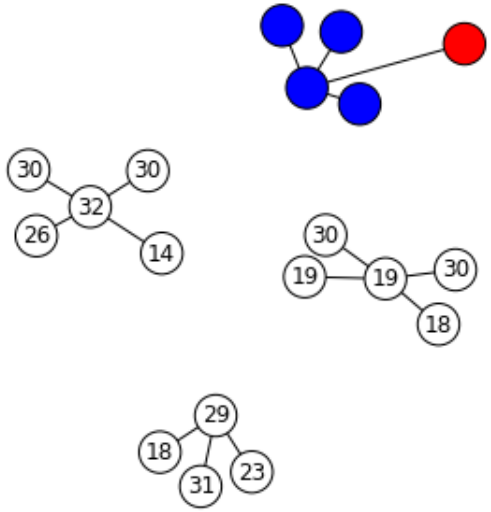
- Idea: when we encounter a new station set S, look for any **satisfiable superset** or any **unsatisfiable subset**
- Problem: we can't **query this cache** with a hash function
  - An exponential number of keys could match a given query
- Solution: a **fast, novel caching scheme** that permits subset/superset queries

Problem Decomposition

- Disconnected components of a given problem's induced constraint graph are **independent problems**
  - Smaller problems also more likely to generate **cache hits**



- Previous solution solves all but **one component** containing the added station



Removing Underconstrained Stations

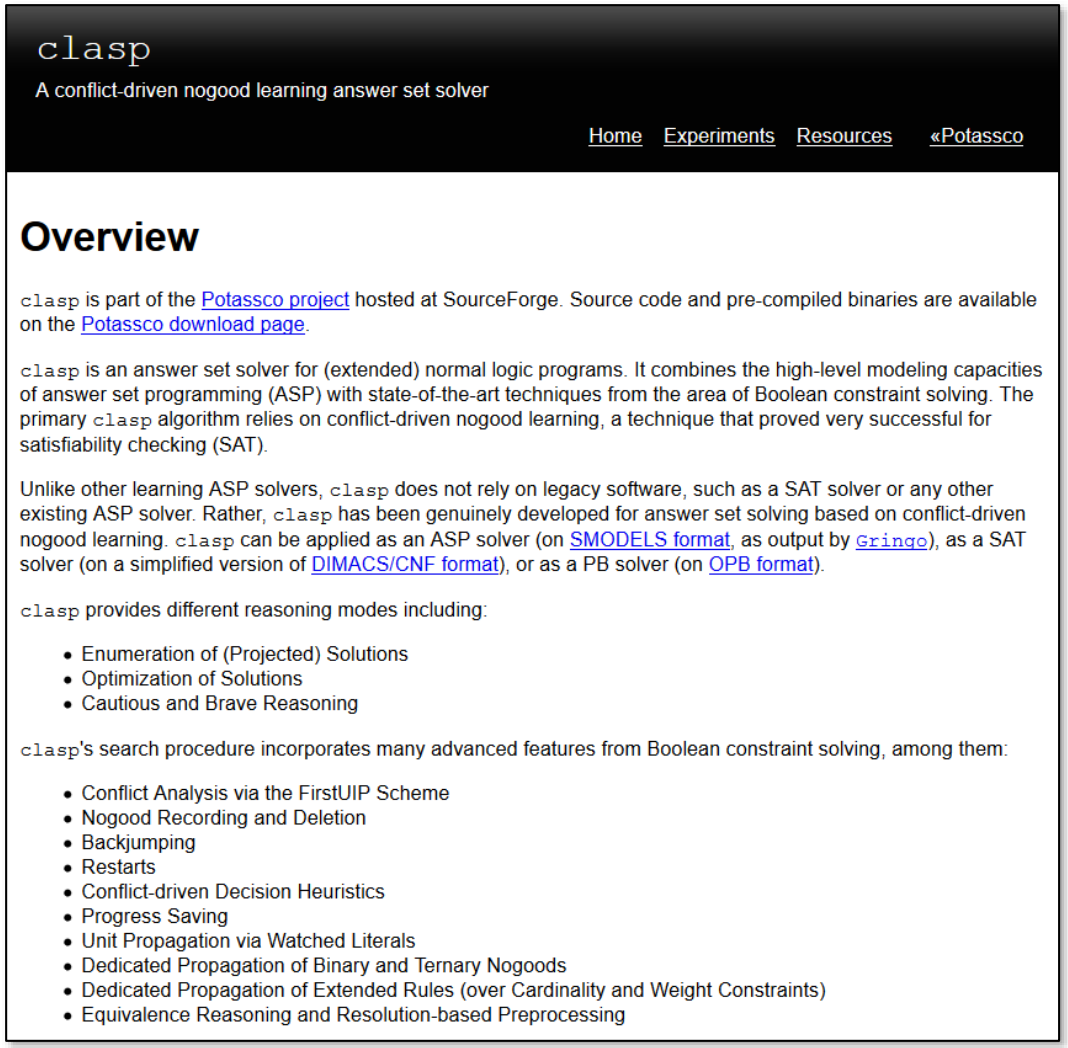
- No matter how neighbors are assigned channels, **underconstrained stations** always have a feasible channel remaining
  - remove these stations from the problem
  - solve the smaller problem
  - add them back at the end if the problem is feasible
- Find such stations via **sound but incomplete** heuristics
  - trade off quality vs running time
  - averaging across instances, our strongest heuristic identified **56% of stations** as underconstrained
- Removing these stations **enhances decomposition**
  - and hence further **enhances caching**

Portfolio Construction via Deep Optimization

- We now (effectively) have an algorithm with a large and deep parameter space:
  - Choose a **complete or local-search solver**?
    - Which one?
      - with which solver parameters
        - » and, depending on solver, conditional subparameters?

clasp: our Best Performing Complete Solver

cf. [Gebser, Kaminski, Kaufmann & Schaub, 2012]
http://www.cs.uni-potsdam.de/clasp



clasp

A conflict-driven nogood learning answer set solver

Home Experiments Resources «Potassco

Overview

clasp is part of the Potassco project hosted at SourceForge. Source code and pre-compiled binaries are available on the Potassco download page.

clasp is an answer set solver for (extended) normal logic programs. It combines the high-level modeling capacities of answer set programming (ASP) with state-of-the-art techniques from the area of Boolean constraint solving. The primary clasp algorithm relies on conflict-driven nogood learning, a technique that proved very successful for satisfiability checking (SAT).

Unlike other learning ASP solvers, clasp does not rely on legacy software, such as a SAT solver or any other existing ASP solver. Rather, clasp has been genuinely developed for answer set solving based on conflict-driven nogood learning. clasp can be applied as an ASP solver (on SMODELS format, as output by Gringo), as a SAT solver (on a simplified version of DIMACS/CNF format), or as a PB solver (on OPB format).

clasp provides different reasoning modes including:

- Enumeration of (Projected) Solutions
- Optimization of Solutions
- Cautious and Brave Reasoning

clasp's search procedure incorporates many advanced features from Boolean constraint solving, among them:

- Conflict Analysis via the FirstUIP Scheme
- Nogood Recording and Deletion
- Backjumping
- Restarts
- Conflict-driven Decision Heuristics
- Progress Saving
- Unit Propagation via Watched Literals
- Dedicated Propagation of Binary and Ternary Nogoods
- Dedicated Propagation of Extended Rules (over Cardinality and Weight Constraints)
- Equivalence Reasoning and Resolution-based Preprocessing

Very highly configurable: ideal for deep optimization

SATenstein: a highly parameterized LS framework

[Khudabukhsh, Xu, Hoos, L-B; 2009, 2016]

- **Frankenstein's** goal:
  - Create "perfect" human being from scavenged body parts
- **SATenstein's** goal:
  - Create high-performance SAT solvers using components scavenged from existing solvers
- Components drawn from or inspired by **existing local search algorithms** for SAT
  - parameters determine which components are selected and how they behave (41 total)
  - designed for use with deep optimization (3 levels of conditional params)
- SATenstein can instantiate:
  - most high-performance solvers **previously proposed** in the literature
    - at least 26 distinct solvers; for this project we added 3 more, including DCCA
  - trillions of **novel** solver strategies

Portfolio Construction via Deep Optimization

- We now (effectively) have an algorithm with a large and deep parameter space:
  - Choose a **complete or local-search solver**?
    - clasp or SATenstein?
      - with which solver parameters
        - » and, depending on solver, conditional subparameters?
  - Which **problem-specific speedups**?
    - …again with their own parameters
  - Plus some **generic speedups** not yet mentioned:
    - AC3 (arc consistency)
    - Changing the SAT encoding

- And, is a **single algorithm** enough?

Algorithm Portfolios

[L-B, Nudelman, Shoham, 2002-2009; Xu, Hutter, Hoos, L-B, 2007-12]

- 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
- Or, just run all the algorithms in the portfolio together **in parallel**

Hydra: Automatic Portfolio Synthesis

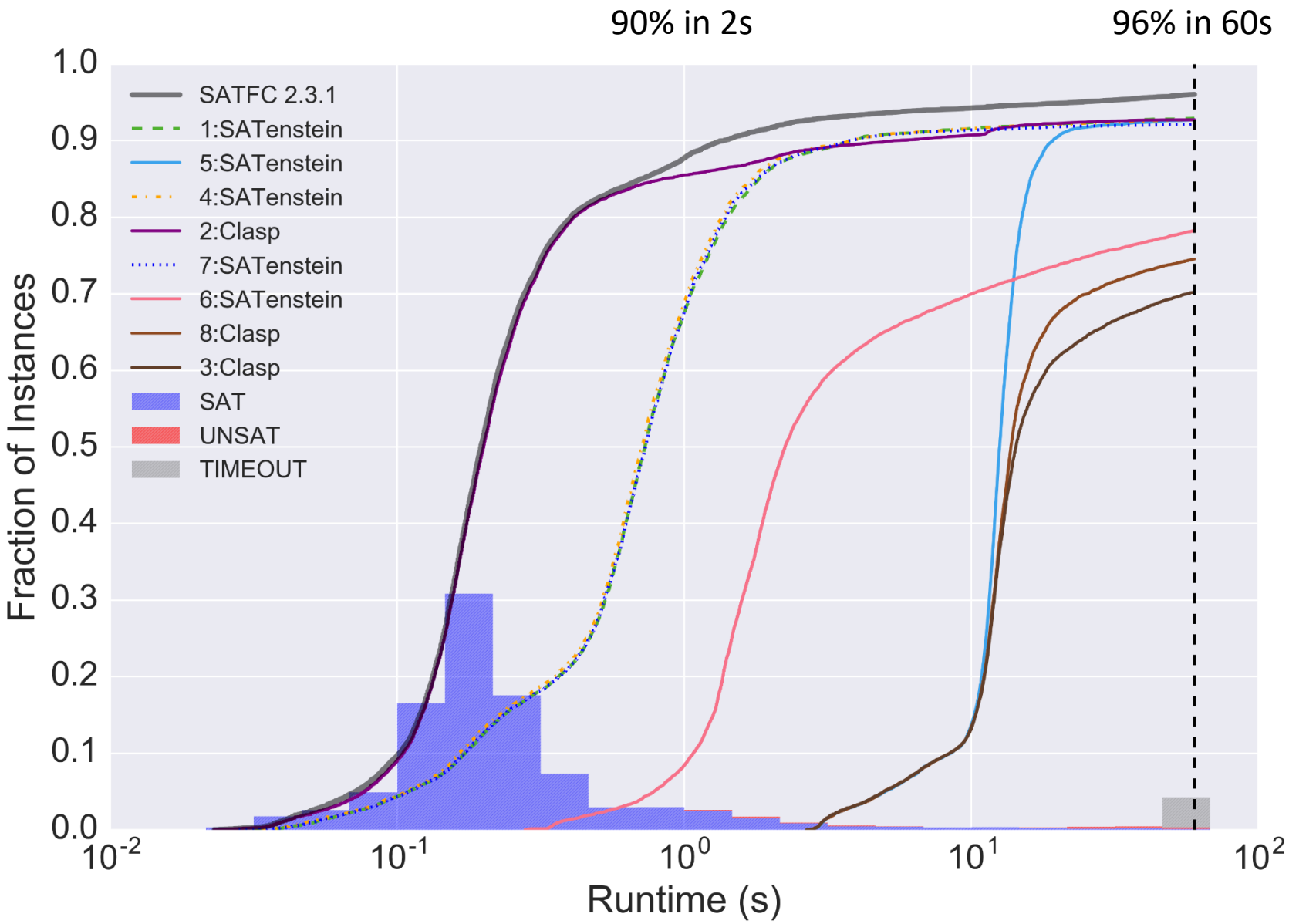
[Xu, Hoos, L-B, 2010; Xu, Hutter, Hoos, L-B, 2011; Lindauer, Hoos, L-B, Schaub, 2016]

- Hydra: augment an additional portfolio *P* by targeting instances **on which *P* performs poorly**
- Give SMAC a dynamic performance metric:
  - performance of alg *s* when *s* outperforms *P;* performance of *P* otherwise
  - Intuitively: *s* scored for **marginal contribution** to *P*

Iteratively optimize this metric, given clasp and Satenstein:

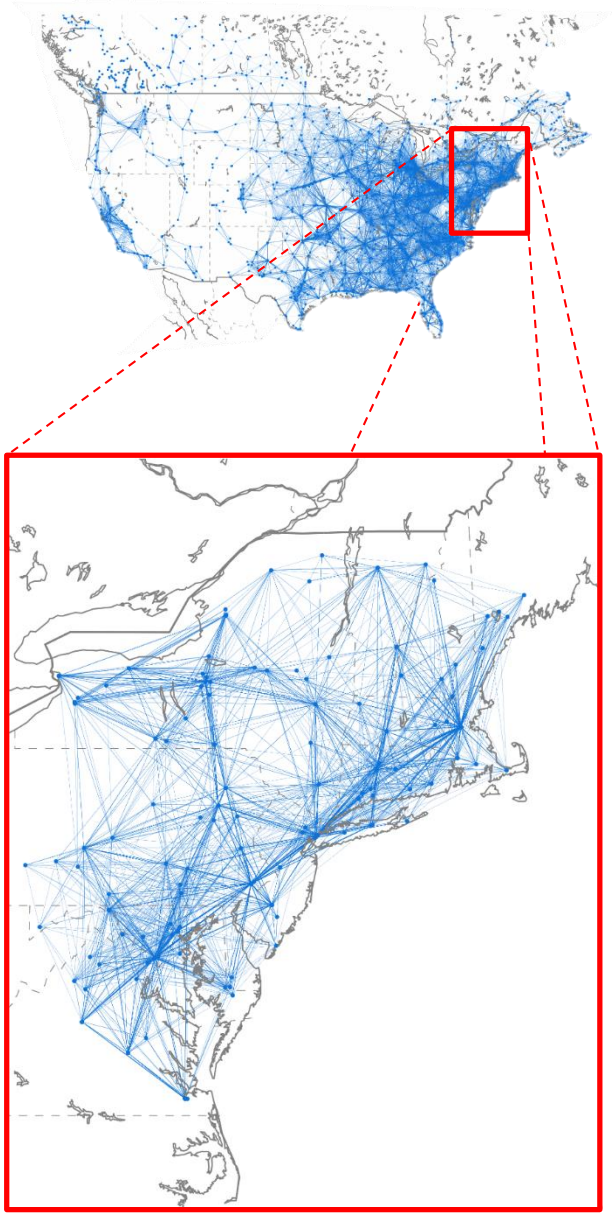
#	Solver	Solved (%)	Δ(%)	Time (s)
1	Satenstein+ (No 1-channel constraints, Components)	92.91	–	53 874
2	Clasp (Expanding presolver)	94.42	1.51	39 415
3	Clasp (AC3, Underconstrained, Components)	95.14	0.72	37 751
4	Satenstein+ (No 1-channel constraints, Components)	95.59	0.45	34 967
5	Satenstein+ (AC3, Underconstrained, Components, No 1-channel constraints)	95.78	0.19	34 329
6	Satenstein+	95.92	0.14	33 650
7	Satenstein+ (No 1-channel constraints, Components)	96.01	0.09	32 569
8	Clasp (AC3, Underconstrained, Components)	96.03	0.02	32 503

Putting It All Together

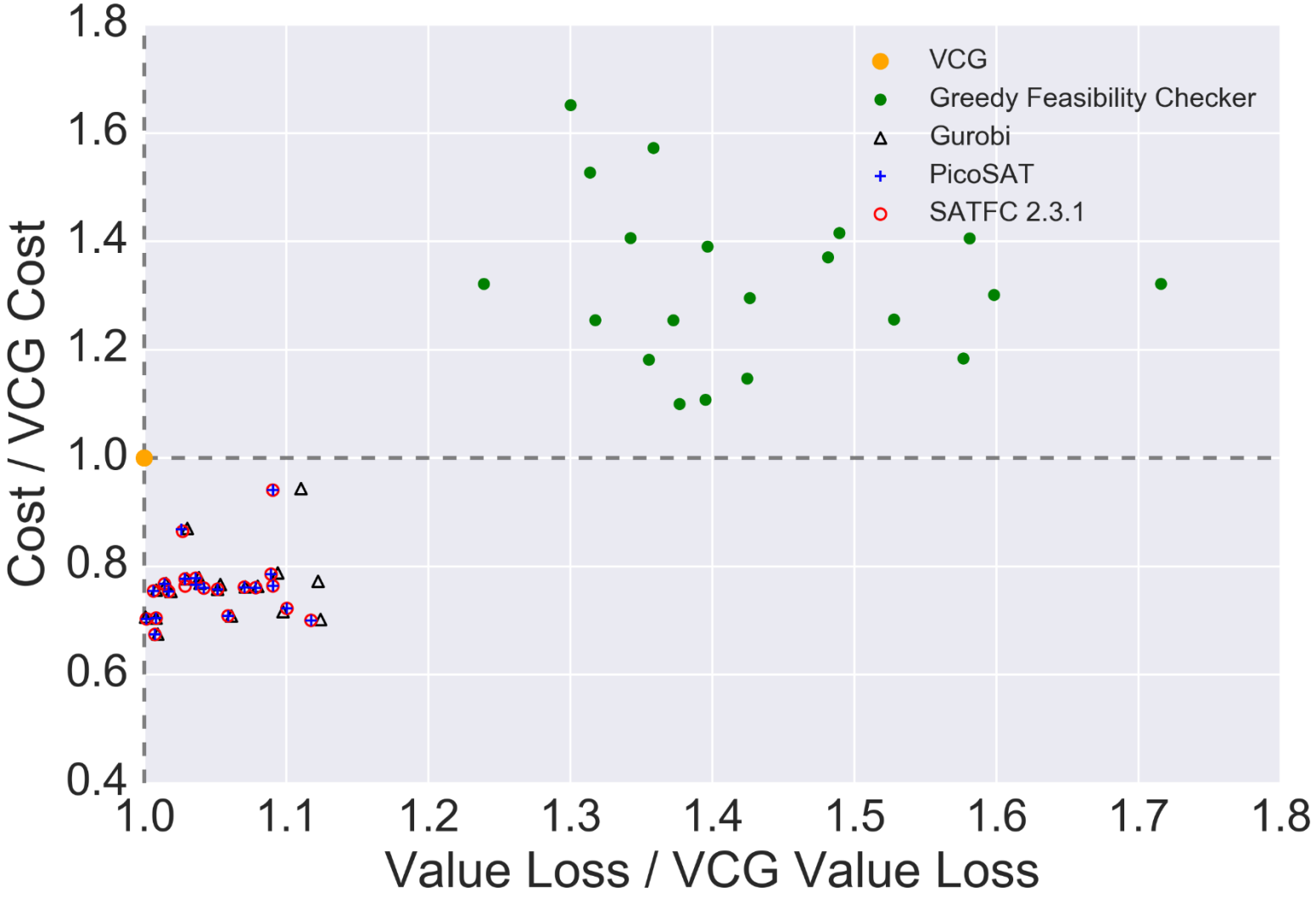


Evaluating the Design on Practical Problems

- Can't run VCG on national-scale problems: can't find optimal packings
- To make a realistic problem we could solve exactly, we restricted to all stations within **two constraint hops of New York**
  - a very densely connected region
  - 218 stations met this criterion

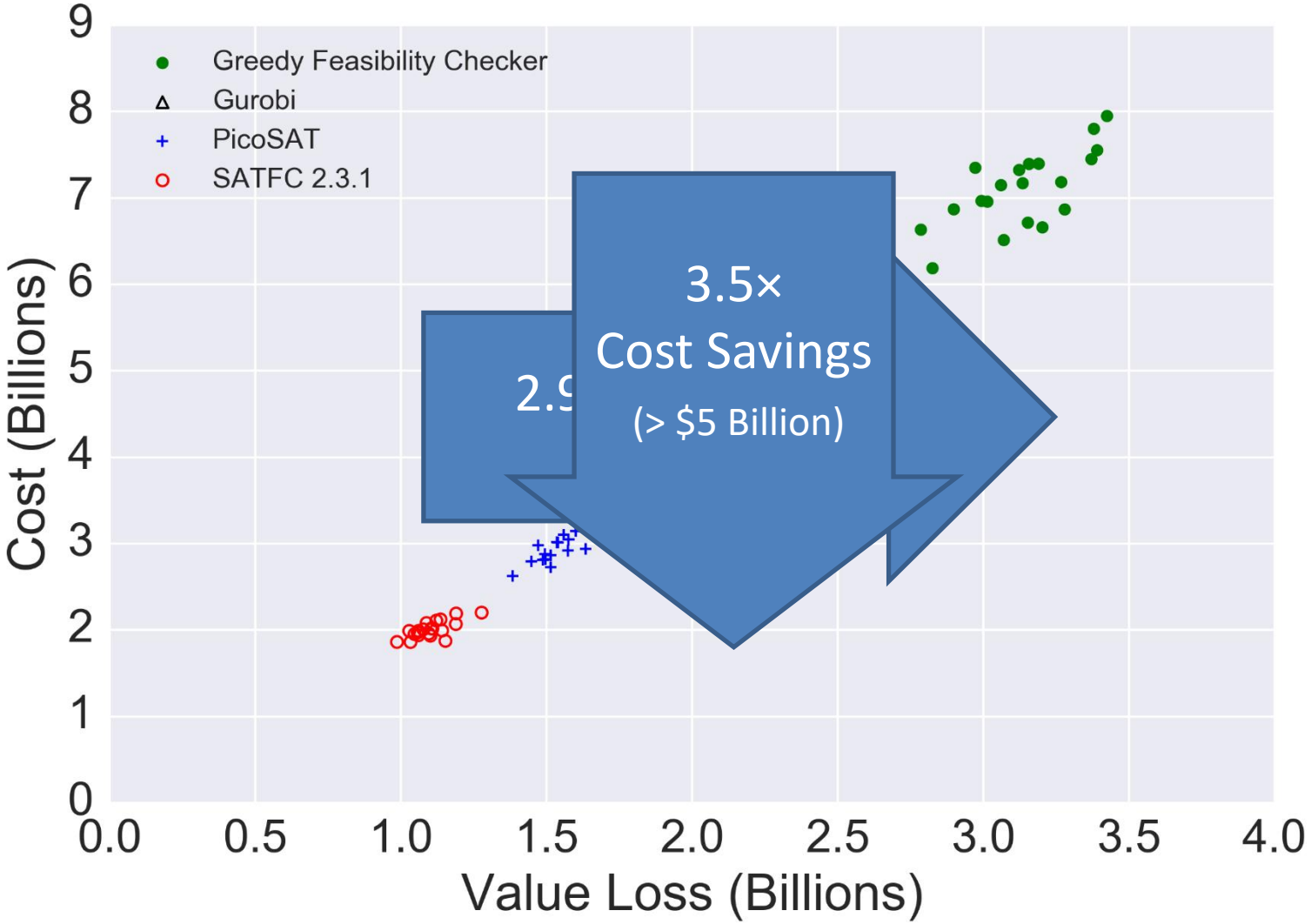


Outcome Quality (NYC + 2 Hops)



"Greedy": check whether existing solution can be directly augmented with new station

Outcome Quality (National)



"Greedy": check whether existing solution can be directly augmented with new station

Conclusions

- Spectrum reallocation is a **socially important problem** that posed interesting new challenges for auction theory
  - defining **property rights**
  - expressing **constraints about externalities** in a tractable way
  - determining **amount of spectrum** to repurpose
  - finding a computationally tractable, robust, budget balanced, and easy to understand mechanism
- The FCC used **descending auctions** to buy back spectrum from TV broadcasters
  - **advantages:** simple, robust, many good economic properties
  - **a key challenge:** ~100,000 NP-complete problems must be solved in real time; auction revenue suffers when they can't be
    - I described how this repacking problem was **solved at national scale**, via "**deep optimization**" (algorithm configuration; algorithm portfolios); SATenstein; problem-specific speedups; caching