Cornell University
Microsoft Research
THE UNIVERSITY OF BRITISH COLUMBIA

Procrastinating with Confidence: Near-Optimal, Anytime, Adaptive Algorithm Configuration

Robert Kleinberg, Kevin Leyton-Brown, Brendan Lucier, Devon Graham

INRODUCTION

- ***Algorithm Configuration:*** finding parameter settings for an algorithm so that it performs well on inputs drawn from a given distribution

Structured Procrastination with Confidence (SPC):

- ***anytime*** algorithm configuration procedure
- ***optimal*** runtime, up to log factors
- ***adaptive*** to be faster on easier problem instances

PROBLEM SETUP

- $i \in N$: Potential **parameters** to choose from
- $j \sim \Gamma$: Distribution over possible **input instances**
- $R(i, j)$: **Runtime** of parameter setting i on input j

GOAL: (ϵ, δ)-OPTIMALITY

- We want to find a parameter $i^* \in N$ s.t.

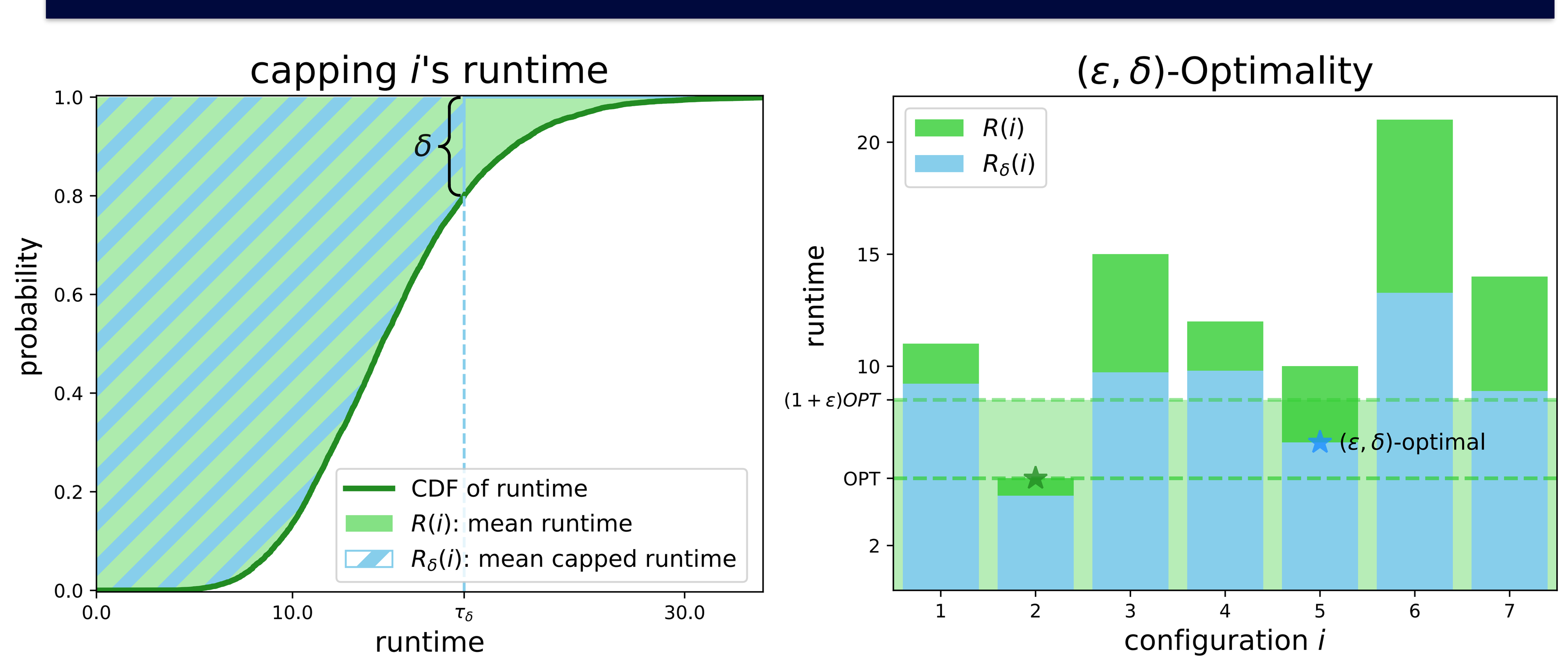
$$R_\delta(i^*) \leq (1+\epsilon) OPT$$

where:

- $R_\delta(i) = \mathbb{E}_{j\sim\Gamma}[\min\{R(i,j), \tau_\delta\}]$ is expected runtime of i capping at τ_δ, the $(1-\delta)$-quantile (i.e., $\Pr_{j\sim\Gamma}[R(i,j) > \tau_\delta] \leq \delta$)
- $OPT = \min_{i\in N} \mathbb{E}_{j\sim\Gamma}[R(i,j)]$ is expected runtime of the optimal configuration

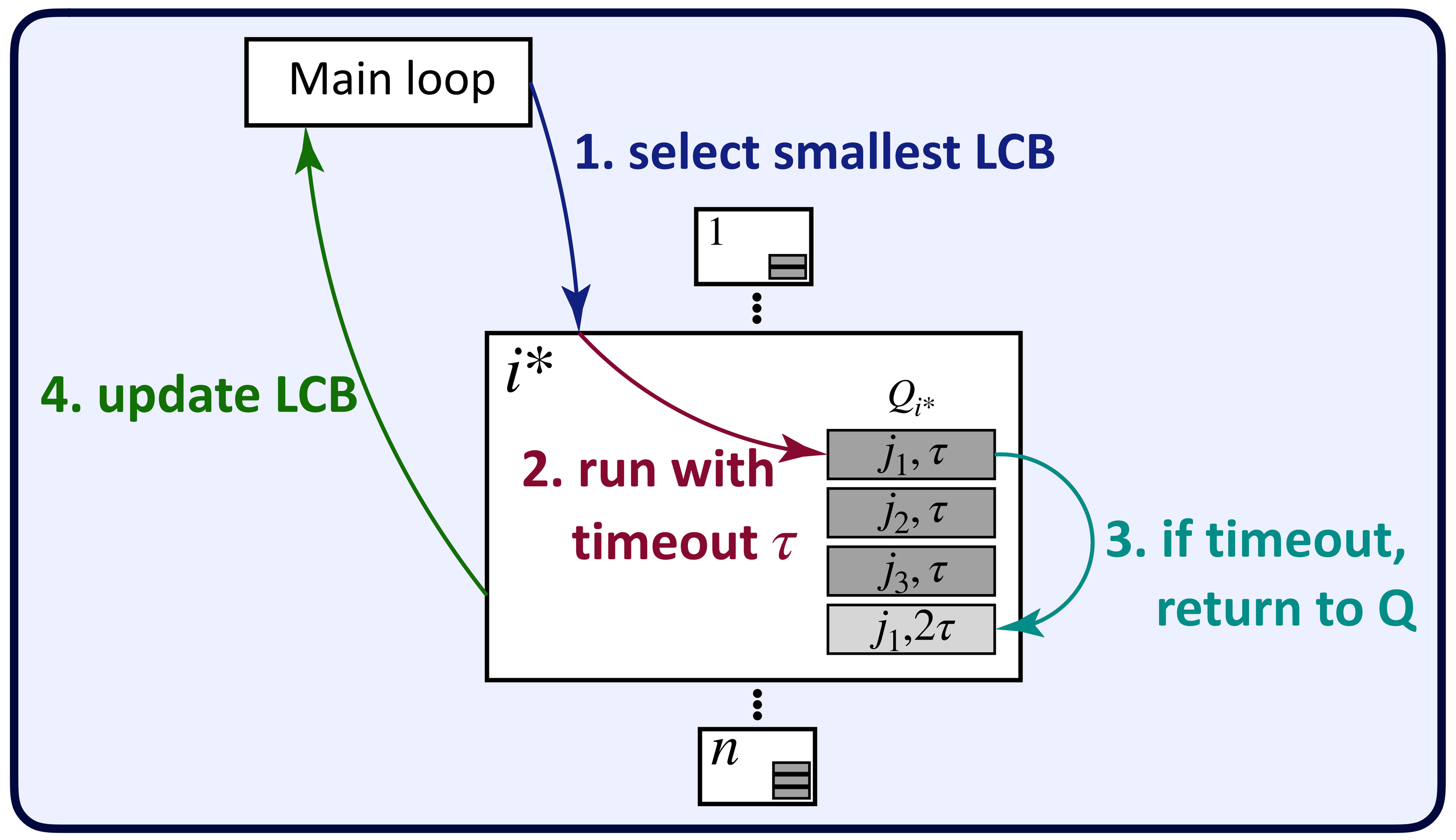
i^* is within a $(1+\epsilon)$-factor of the best configuration if we discard the worst δ-fraction of i^*'s runtimes

EXAMPLE



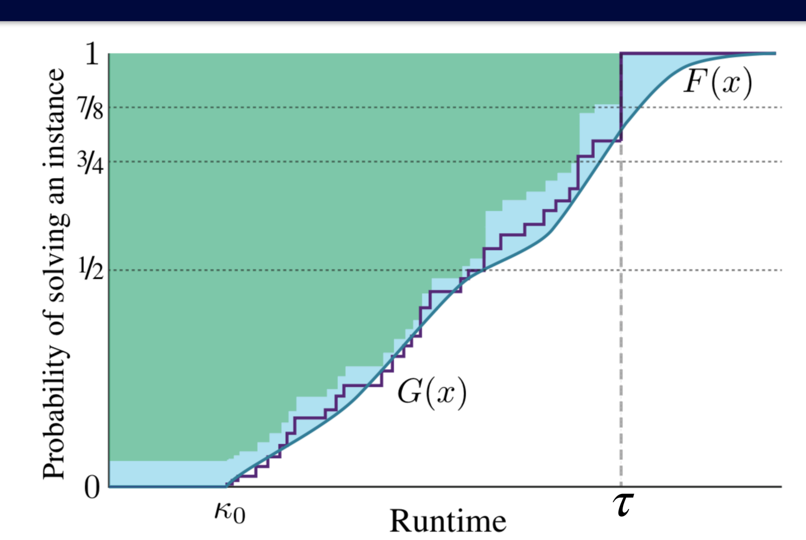
STRUCTURED PROCRASTINATION WITH CONFIDENCE

```
  for i ∈ N do
      Q_i ← queue of (input, captime) pairs sampled from Γ
  while search is not interrupted do
1     i* ← configuration with smallest LCB of mean runtime
      j, τ ← Q_{i*}.pop()
2     run i* on input j
3     if i* times out at captime τ then
          Q_{i*}.push((j, 2τ))
4     update i*'s LCB
  return configuration that ran on the most instances
```



COMPUTATION OF THE LCB

- $F(x)$ is **true CDF**
- Area above $F(x)$ is **true mean runtime**
- $G(x)$ is observed **empirical CDF**
- **GREEN** area is **LCB**



RUNTIME GUARANTEE

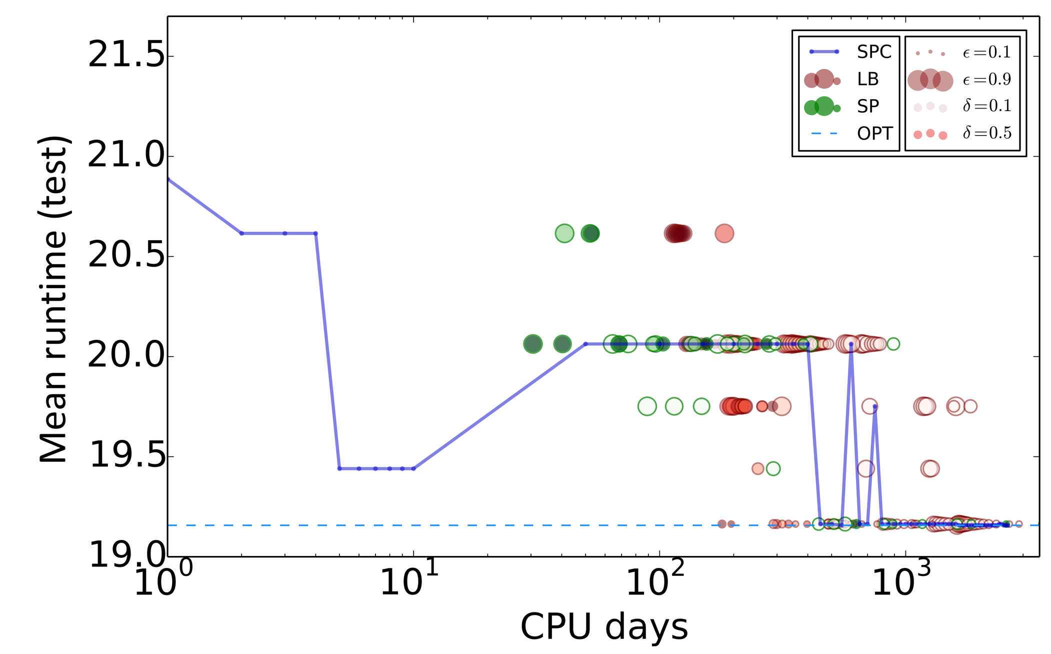
SPC returns an (ϵ, δ)-optimal configuration if its runtime is

$$\tilde{\Omega}\left(\mathrm{OPT}\left(\frac{|S|}{\epsilon^2\delta} + \sum_{i\notin S}\frac{1}{\epsilon_i^2\delta_i}\right)\right)$$

- S is the set of (ϵ, δ)-optimal configurations
- For each $i \notin S$, i is (ϵ_i, δ_i)-suboptimal ($\epsilon_i \geq \epsilon$ and $\delta_i \geq \delta$)

EXPERIMENTAL RESULTS

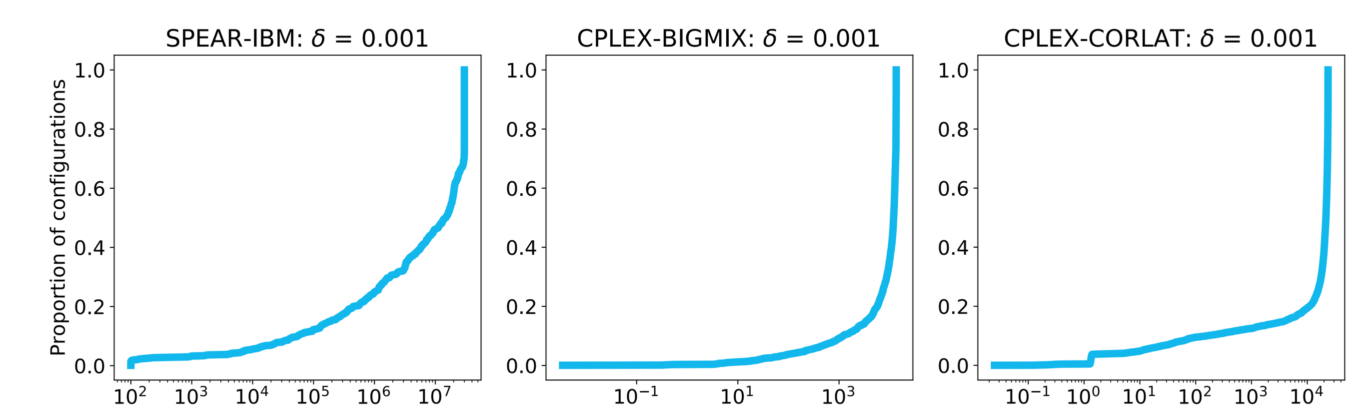
- SPC is able to find a good configuration more quickly than other methods:



✸972 configurations of minisat SAT solver on 20118 CNFuzzDD SAT instances

RUNTIME VARIATION IN PRACTICE

SPC is fastest when most configurations are far from optimal, a common scenario in practice



✸Proportion of (ϵ, δ)-optimal configurations for solver/input distribution pairs

EXTENSION FOR INFINITE N

- Find i^* that is competitive with OPT^γ, the best configuration left after excluding the fastest γ-fraction

Achieve similar runtime guarantee, with OPT^γ in place of OPT

- **Sample** a set $\hat{N}$ of size $O(1/\gamma \log(1/\gamma))$. **Run SPC** on $\hat{N}$
- Can extend this idea to **refine γ over** time in anytime setting

RELATED WORK

- *Efficiency Through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees.* Robert Kleinberg, Kevin Leyton-Brown, Brendan Lucier. IJCAI 2017.
- *LeapsAndBounds: A Method for Approximately Optimal Algorithm Configuration.* Gellért Weisz, András György, Csaba Szepesvári. ICML 2018.
- *CapsAndRuns: An Improved Method for Approximately Optimal Algorithm Configuration.* Gellért Weisz, András György, Csaba Szepesvári. ICML 2019.