**INTRODUCTION**

- **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**

- \( i \in \mathbb{N} \): 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**: \((\epsilon, \delta)\)-OPTIMALITY

- We want to find a parameter \( \hat{i}^* \in \mathbb{N} \) s.t.
  \[
  R_S(\hat{i}^*) \leq (1 + \epsilon) \text{OPT}
  \]
  where:
  - \( R_S(i) = \mathbb{E}_{j \sim \Gamma}[\min(R(i, j), \tau_j)] \) is expected runtime of \( i \) capping at \( \tau_j \), the \((1 - \delta)\)-quantile, i.e., \( \mathbb{P}(R(i, j) > \tau_j) \leq \delta \)
  - \( \text{OPT} = \min_{i \in \mathbb{N}} \mathbb{E}_{j \sim \Gamma}[R(i, j)] \) is expected runtime of the optimal configuration

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

**EXAMPLE**

- **capping \( i \)'s runtime**
- \((\epsilon, \delta)\)-Optimality

**STRUCTURED PROCRASTINATION WITH CONFIDENCE**

for \( i \in \mathbb{N} \) do
  \( Q_i \leftarrow \) queue of \((\text{input}, \text{captime})\) pairs sampled from \( \Gamma \)
  while search is not interrupted do
    \( i^* \leftarrow \) configuration with smallest LCB of mean runtime
    \( j, \tau \leftarrow Q_{i^*}.\text{pop}() \)
    run \( i^* \) on input \( j \)
    if \( i^* \) times out at captime \( \tau \) then
      \( Q_{i^*}.\text{push}(j, 2\tau) \)
    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 \((\epsilon, \delta)\)-optimal configuration if its runtime is

\[
\hat{\Omega}\left(\text{OPT} + \frac{|S|}{\epsilon^2 \delta^2} + \sum_{i \in S} \frac{1}{\epsilon^2 \delta_i^2}\right)
\]

- \( S \) is the set of \((\epsilon, \delta)\)-optimal configurations
- For each \( i \not\in S \), \( i \) is \((\epsilon_i, \delta_i)\)-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 mizzaSAT solver on 20118 CNFuzzDD SAT instances
  - Proportion of \((\epsilon, \delta)\)-optimal configurations for solver/input distribution pairs

**RUNTIME VARIATION IN PRACTICE**

- **SPC** is fastest when most configurations are far from optimal, a common scenario in practice
  - Find \( \hat{i}^* \) that is competitive with \( \text{OPT}' \), the best configuration left after excluding the fastest \( \gamma \)-fraction
  - Achieve similar runtime guarantee, with \( \text{OPT}' \) in place of \( \text{OPT} \)
  - Sample a set \( \hat{N} \) of size \( O(1/\gamma \log(1/\gamma)) \). Run SPC on \( \hat{N} \)
  - Can extend this idea to refine \( \gamma \) over time in anytime setting

**RELATED WORK**