Learning near-optimal hyperparameters with minimal overhead

Gellért Weisz András György **Csaba Szepesvári**

DeepMind

Workshop on Automated Algorithm Design

(TTIC 2019)

August 7, 2019

Introduction

- Problem: find good parameter settings (configurations) for general purpose solvers.
  - No structure assumed over the parameter space.

Introduction

- Problem: find good parameter settings (configurations) for general purpose solvers.
  - No structure assumed over the parameter space.

- Zillions of practical algorithms $\Leftrightarrow$ Little theory

  Want theoretical guarantees on the runtime of
  - the chosen configuration; and
  - the configuration process.

Introduction

- Problem: find good parameter settings (configurations) for general purpose solvers.
  - No structure assumed over the parameter space.

- Zillions of practical algorithms $\Leftrightarrow$ Little theory

  Want theoretical guarantees on the runtime of
  - the chosen configuration; and
  - the configuration process.

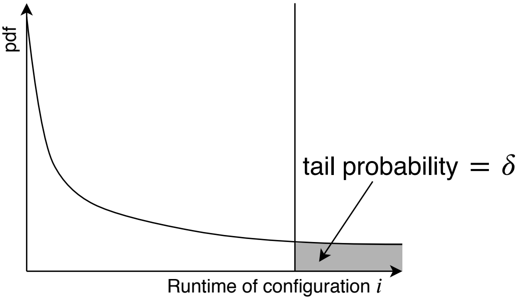
- Goal: find a near-optimal configuration solving $1 - \delta$ fraction of the problems in the least expected time.
  - Since some instances (δ fraction) are hopelessly hard; don't want to solve those.

Problem formulation

Given: n configurations, distribution Γ of problem instances.

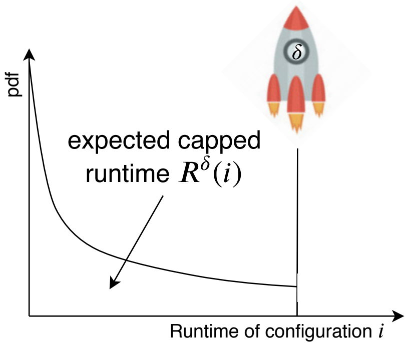
Problem formulation

Given: n configurations, distribution Γ of problem instances.



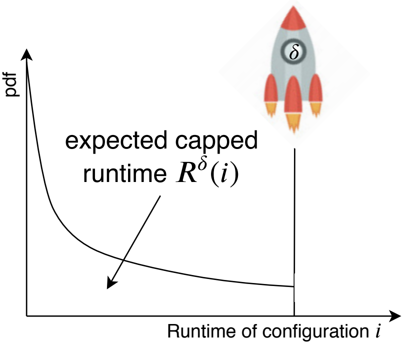
Problem formulation

Given: n configurations, distribution Γ of problem instances.



Problem formulation

Given: n configurations, distribution Γ of problem instances.

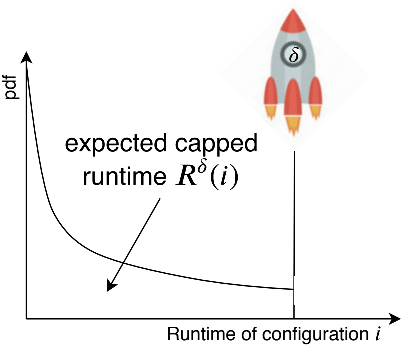


Runtime of the optimal capped configuration:

$$\mathrm{OPT}_{\delta} = \min_{i} R^{\delta}(i)$$

Problem formulation

Given: n configurations, distribution Γ of problem instances.



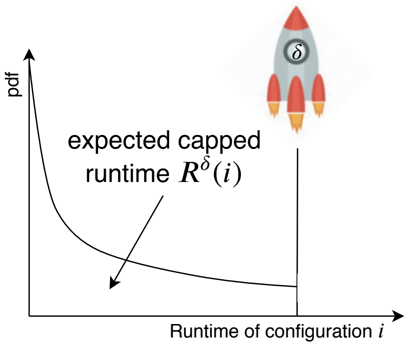
Runtime of the optimal capped configuration:

$$\mathrm{OPT}_\delta = \min_i R^\delta(i)$$

Configuration i is (ε, δ)-optimal if $R^\delta(i) \le (1+\varepsilon)\mathrm{OPT}_{\delta/2}$.

Problem formulation

Given: n configurations, distribution Γ of problem instances.



Runtime of the optimal capped configuration:

$$\mathrm{OPT}_\delta = \min_i R^\delta(i)$$

Configuration i is (ε, δ)-optimal if $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_{\delta/2}$.

Note that $\mathrm{OPT}_\delta \leq \mathrm{OPT}_{\delta/2} \leq \mathrm{OPT}_0$ – gaps can be large!

Previous work (before ICML'19)

Structured Procrastination

(Kleinberg et al., 2017)

- Relaxed goal: Find i with $R^{\delta}(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Worst-case lower bound: runtime must be at least $\Omega\left(\mathrm{OPT}_0 \frac{n}{\varepsilon^2\delta}\right)$
- With probability $1-\zeta$, returns an (ε, δ)-optimal configuration in worst-case time

$$\mathcal{O}\left(\mathrm{OPT}_0 \, \frac{n}{\varepsilon^2\delta} \log\left(\frac{n \log \bar{\kappa}}{\zeta\varepsilon^2\delta}\right)\right)$$

  - $\overline{\kappa}$: absolute upper bound on runtimes

Structured Procrastination

(Kleinberg et al., 2017)

- Relaxed goal: Find i with $R^{\delta}(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Worst-case lower bound: runtime must be at least $\Omega(\mathrm{OPT}_0 \frac{n}{\varepsilon^2\delta})$
- With probability $1-\zeta$, returns an (ε,δ)-optimal configuration in worst-case time

$$\mathcal{O}\left(\mathrm{OPT}_0 \, \frac{n}{\varepsilon^2\delta} \log\left(\frac{n \log \bar{\kappa}}{\zeta\varepsilon^2\delta}\right)\right)$$

  - $\bar{\kappa}$: absolute upper bound on runtimes

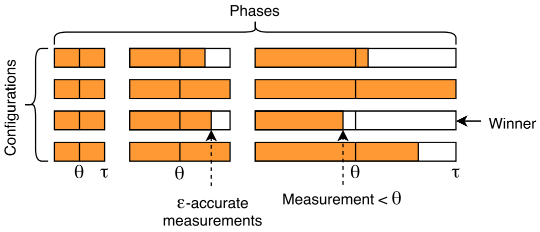
Can we remove $\bar{\kappa}$?

Can we improve runtime when problem is easier?

LEAPSANDBOUNDS

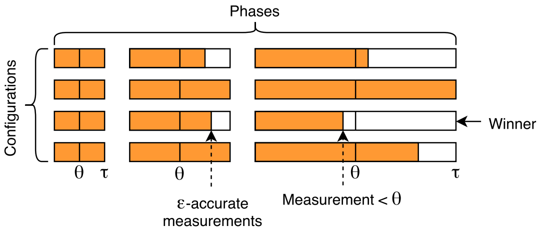
(Weisz et al., 2018)

1. Guess a value θ of OPT, starting from a low value
2. Test whether $R^\delta(i) \leq \theta$ for some configuration i:
   - For each i, run $b = \tilde{\mathcal{O}}(\frac{1}{\delta\varepsilon^2})$ instances with instance-wise timeout $\tau = \frac{4\theta}{3\delta}$, abort if empirical average exceeds θ.
3. Return the configuration with the smallest mean amongst successful configurations. If no test succeeded, double θ, continue from Step 2.



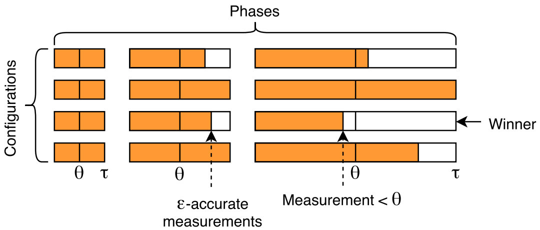
Average runtime budget and its use across different configurations and phases

Why does this work?



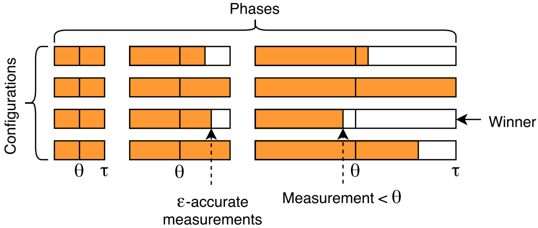
- w.h.p., for any configuration i:

Why does this work?



- w.h.p., for any configuration i:
  - if runs complete within θ average runtime:

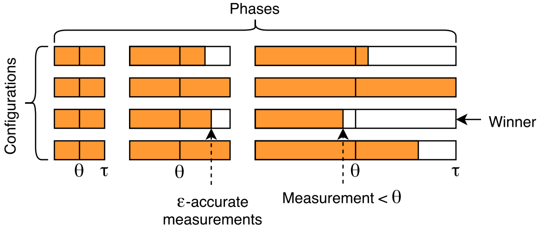
Why does this work?



- w.h.p., for any configuration i:
  - if runs complete within θ average runtime:

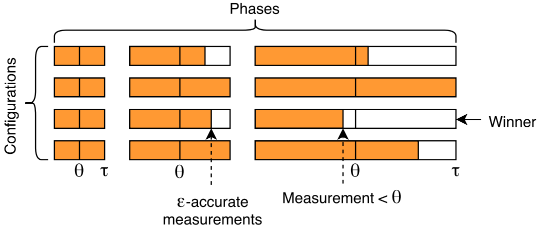
    (i) τ is above δ-quantile for configuration i

Why does this work?



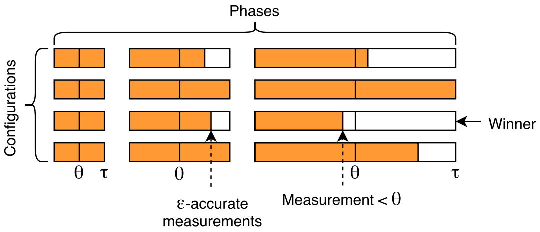
- w.h.p., for any configuration i:
  - if runs complete within θ average runtime:
    (i) τ is above δ-quantile for configuration i
    (ii) Empirical mean $\bar{R}_i$ is ε-close to $R_\tau(i) = \mathbb{E}[X(i, J) \wedge \tau]$, $J \sim \Gamma$

Why does this work?



- w.h.p., for any configuration i:
  - if runs complete within θ average runtime:
    - (i) τ is above δ-quantile for configuration i
    - (ii) Empirical mean $\bar{R}_i$ is ε-close to $R_\tau(i) = \mathbb{E}[X(i,J) \wedge \tau]$, $J \sim \Gamma$
  - otherwise, $R^\delta(i) > \theta$, hence can safely abandon i for this phase

Why does this work?



- w.h.p., for any configuration i:
  - if runs complete within θ average runtime:
    - (i) τ is above δ-quantile for configuration i
    - (ii) Empirical mean $\bar{R}_i$ is ε-close to $R_\tau(i) = \mathbb{E}[X(i,J) \wedge \tau]$, $J \sim \Gamma$
  - otherwise, $R^\delta(i) > \theta$, hence can safely abandon i for this phase

- Thus, if for any configuration i, $\bar{R}_i < \theta$, then for $i^* = \operatorname{argmin}_i \bar{R}_i$, $R^\delta(i^*) \leq (1+\varepsilon)\mathrm{OPT}_0$ w.h.p.

Guarantees

Theorem

With high probability,

(i) the algorithm finds an (ε, δ)-optimal configuration;

(ii) the worst case runtime is $\mathcal{O}\left(\mathrm{OPT}_0 \frac{n}{\varepsilon^2 \delta} \log\left(\frac{n \log \mathrm{OPT}_0}{\zeta}\right)\right)$.

Guarantees

Theorem

With high probability,

(i) the algorithm finds an (ε, δ)-optimal configuration;

(ii) the worst case runtime is $\mathcal{O}\left(\mathrm{OPT}_0 \frac{n}{\varepsilon^2 \delta} \log\left(\frac{n \log \mathrm{OPT}_0}{\zeta}\right)\right)$.

- Improvement: Empirical Bernstein stopping
  Stop testing a configuration i when confidence intervals already indicate that (a) i is not optimal with the given timeout; or (b) i is already estimated with ε accuracy.

Guarantees

Theorem

With high probability,

- *(i) the algorithm finds an (ε, δ)-optimal configuration;*
- *(ii) the worst case runtime is* $\mathcal{O}\left(\mathrm{OPT}_0 \frac{n}{\varepsilon^2 \delta} \log\left(\frac{n \log \mathrm{OPT}_0}{\zeta}\right)\right)$.

- Improvement: Empirical Bernstein stopping
  Stop testing a configuration i when confidence intervals already indicate that (a) i is not optimal with the given timeout; or (b) i is already estimated with ε accuracy.
- Runtime:

$$\mathcal{O}\left[\mathrm{OPT}_0 \sum_{i=1}^{n} \max\left(\frac{\sigma_{i,k}^2}{\varepsilon^2 R_{\tau_k}^2(i)}, \frac{1}{\varepsilon\delta}, \frac{1}{\delta}\log\frac{1}{\delta}\right)\left(\log\frac{n \log \mathrm{OPT}_0}{\zeta} + \log\frac{1}{\varepsilon R_{\tau_k}(i)}\right)\right].$$

Guarantees

Theorem

With high probability,

(i) the algorithm finds an (ε, δ)-optimal configuration;

(ii) the worst case runtime is $\mathcal{O}\left(\mathrm{OPT}_0 \frac{n}{\varepsilon^2 \delta} \log\left(\frac{n \log \mathrm{OPT}_0}{\zeta}\right)\right)$.

- Improvement: Empirical Bernstein stopping
  Stop testing a configuration i when confidence intervals already indicate that (a) i is not optimal with the given timeout; or (b) i is already estimated with ε accuracy.
- Runtime:

$$\mathcal{O}\left[\mathrm{OPT}_0 \sum_{i=1}^{n} \max\left(\frac{\sigma_{i,k}^2}{\varepsilon^2 R_{\tau_k}^2(i)}, \frac{1}{\varepsilon\delta}, \frac{1}{\delta}\log\frac{1}{\delta}\right)\left(\log\frac{n \log \mathrm{OPT}_0}{\zeta} + \log\frac{1}{\varepsilon R_{\tau_k}(i)}\right)\right].$$

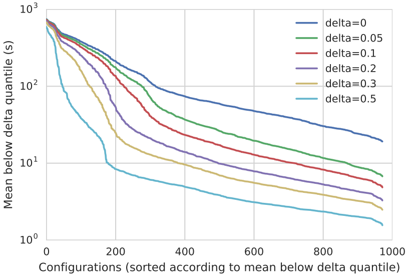
- Huge improvement if the variances are small: $\frac{\sigma_{i,k}^2}{R_{\tau_k}^2} \ll \frac{1}{\delta}$.

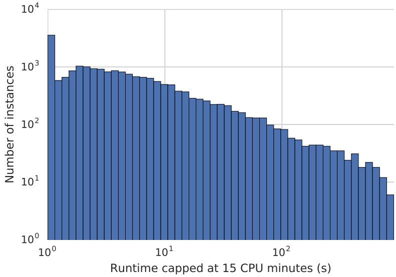
Experiments

- Configuring the minisat SAT solver (Sorensson and Een, 2005)
- 1K configurations, 20K nontrivial problem instances
- Compare with Structured Procrastination by Kleinberg et al. (2017)
- Code and (83 CPU years worth of @ year 2018 CPUs) data:
  https://github.com/deepmind/leaps-and-bounds

Experiments

- Configuring the minisat SAT solver (Sorensson and Een, 2005)
- 1K configurations, 20K nontrivial problem instances
- Compare with Structured Procrastination by Kleinberg et al. (2017)
- Code and (83 CPU years worth of @ year 2018 CPUs) data:
  https://github.com/deepmind/leaps-and-bounds



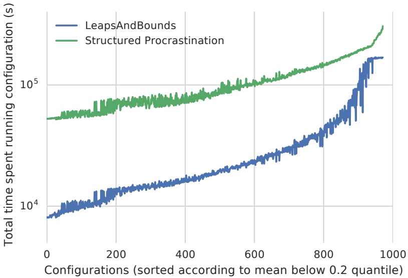


Results

- $\varepsilon = 0.2$, $\delta = 0.2$, $\zeta = 0.1$
- Instead of doubling, use $\theta := 1.25\theta$
- Runs can be stopped and resumed (ie 'continue' running on an instance)

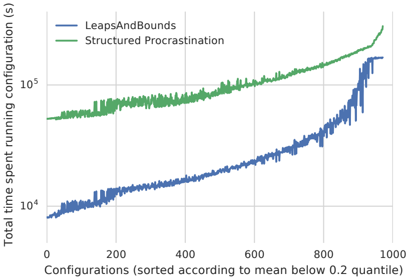
Results

- $\varepsilon = 0.2$, $\delta = 0.2$, $\zeta = 0.1$
- Instead of doubling, use $\theta := 1.25\theta$
- Runs can be stopped and resumed (ie ‘continue’ running on an instance)



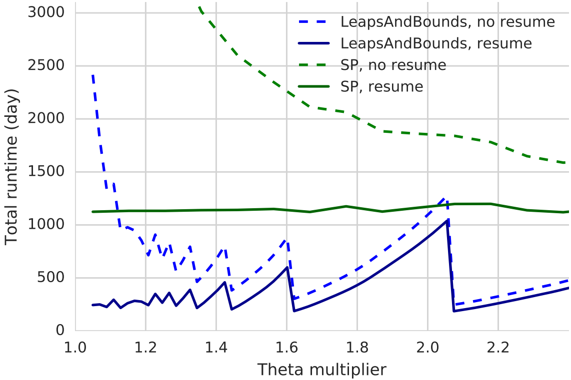
Results

- $\varepsilon = 0.2$, $\delta = 0.2$, $\zeta = 0.1$
- Instead of doubling, use $\theta := 1.25\theta$
- Runs can be stopped and resumed (ie 'continue' running on an instance)



3–20-times improvement in total work
(also across different choices of ε and δ)

Effect of the multiplier of θ



(cost of pause/resume is not modeled)

Current work (ICML'19)

CapsAndRuns algorithm (Weisz et al., 2019)

For all configurations i, in parallel:

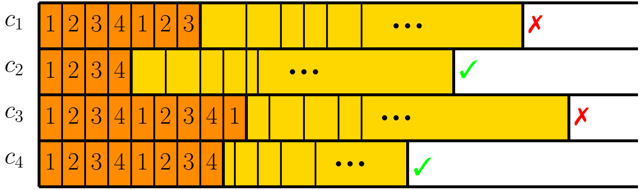
Phase I: Find $t_\delta(i) \le \tau_i \le t_{\delta/2}(i)$:

- Run $\Theta(1/\delta)$ instances in parallel until $1 - \frac{3}{4}\delta$ fraction of them finishes.

Phase II: Find $R_{\tau_i}(i)$ with ε relative accuracy:

- Run sufficiently many instances with timeout τ_i until we get an ε-accurate estimate of $R_{\tau_i}(i)$ ('Bernstein stopping' ala Mnih et al. 2008).

Return: Of the configurations not rejected, select the one with the smallest average capped runtime



CapsAndRuns algorithm (Weisz et al., 2019)

For all configurations i, in parallel:

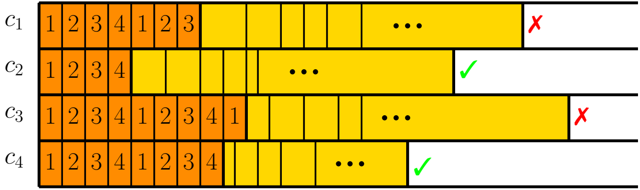
Phase I: Find $t_\delta(i) \le \tau_i \le t_{\delta/2}(i)$:

- Run $\Theta(1/\delta)$ instances in parallel until $1 - \frac{3}{4}\delta$ fraction of them finishes. Abort if taking too much time.

Phase II: Find $R_{\tau_i}(i)$ with ε relative accuracy:

- Run sufficiently many instances with timeout τ_i until we get an ε-accurate estimate of $R_{\tau_i}(i)$ ('Bernstein stopping' ala Mnih et al. 2008). Adjust best runtime UCB and abort if LCB(i)>UCB.

Return: Of the configurations not rejected, select the one with the smallest average capped runtime



Global variables

Set $\mathcal{N}$ of n algorithm configurations
Precision parameter $\varepsilon \in (0, \frac{1}{3})$
Quantile parameter $\delta \in (0, 1)$
Failure probability parameter $\zeta \in (0, \frac{1}{6})$
Instance distribution Γ
$b \leftarrow \left\lceil 48 \frac{1}{\delta} \log\left(\frac{3n}{\zeta}\right) \right\rceil$
$T \leftarrow \infty$ ▷ Time limit, updated continuously by all parallel processes

Global variables

1: Set $\mathcal{N}$ of n algorithm configurations
2: Precision parameter $\varepsilon \in (0, \frac{1}{3})$
3: Quantile parameter $\delta \in (0, 1)$
4: Failure probability parameter $\zeta \in (0, \frac{1}{6})$
5: Instance distribution Γ
6: $b \leftarrow \left\lceil 48 \frac{1}{\delta} \log\left(\frac{3n}{\zeta}\right) \right\rceil$
7: $T \leftarrow \infty$ ▷ Time limit, updated continuously by all parallel processes

Algorithm 2 QUANTILEEST

1: **Inputs:** i
2: **Initialize:** $m \leftarrow \left\lceil (1 - \frac{3}{4}\delta) b \right\rceil$
3: Run configuration i on b instances, in parallel, until m of these complete. Abort if total work $\geq 2Tb$.
4: $\tau \leftarrow$ runtime of m^{th} completed instance
5: **return** τ

Global variables

Set $\mathcal{N}$ of n algorithm configurations
Precision parameter $\varepsilon \in (0, \frac{1}{3})$
Quantile parameter $\delta \in (0, 1)$
Failure probability parameter $\zeta \in (0, \frac{1}{6})$
Instance distribution Γ
$b \leftarrow \left\lceil 48\frac{1}{\delta}\log\left(\frac{3n}{\zeta}\right)\right\rceil$
$T \leftarrow \infty$ ▷ Time limit, updated continuously by all parallel processes

Algorithm 2 QUANTILEEST

Inputs: i
Initialize: $m \leftarrow \lceil (1 - \frac{3}{4}\delta)b \rceil$
Run configuration i on b instances, in parallel, until m of these complete. Abort if total work $\geq 2Tb$.
$\tau \leftarrow$ runtime of m^{th} completed instance
return τ

Algorithm 3 RUNTIMEEST

Inputs: i, τ
Initialize: $j \leftarrow 0$
while True **do**
    Sample j^{th} instance J from Γ
    Let Y be τ capped runtime of i on J
    Update $\bar{Y}$, $\bar{\sigma}^2$, sample mean and variance
    $C = c(\bar{\sigma}, n, j, \zeta, \tau)$
    if $\bar{Y} - C > T$ **then**
        return reject i
    end if
    $T \leftarrow \min\{T, \bar{Y} + C\}$ ▷ lowest upper confidence
    if $C \leq \frac{\varepsilon}{3}(2\bar{Y} - C)$ **then**
        return accept i with runtime estimate $\bar{Y}$
    end if
    $j \leftarrow j + 1$
end while

Global variables

Set $\mathcal{N}$ of n algorithm configurations
Precision parameter $\varepsilon \in (0, \frac{1}{3})$
Quantile parameter $\delta \in (0, 1)$
Failure probability parameter $\zeta \in (0, \frac{1}{6})$
Instance distribution Γ
$b \leftarrow \left\lceil 48 \frac{1}{\delta} \log\left(\frac{3n}{\zeta}\right) \right\rceil$
$T \leftarrow \infty$ ▷ Time limit, updated continuously by all parallel processes

Algorithm 1 CAPSANDRUNS

$\mathcal{N}' \leftarrow \mathcal{N}$ ▷ Pool of competing configurations
for configuration $i \in \mathcal{N}$, in parallel **do**
// Phase I:
Run $\tau_i \leftarrow$ QUANTILEEST (i)
// Phase II:
if QUANTILEEST (i) aborted **then**
Remove i from $\mathcal{N}'$
else
Run RUNTIMEEST (i, τ_i), abort if $|\mathcal{N}'| = 1$
if RUNTIMEEST (i, τ_i) rejected i **then**
Remove i from $\mathcal{N}'$
else
$\bar{Y}(i) \leftarrow$ return value of RUNTIMEEST (i, τ_i)
end if
end if
end for
return $i^* = \text{argmin}_{i \in \mathcal{N}'} \bar{Y}(i)$ and τ_{i^*}

Algorithm 2 QUANTILEEST

Inputs: i
Initialize: $m \leftarrow \left\lceil (1 - \frac{3}{4}\delta) b \right\rceil$
Run configuration i on b instances, in parallel, until m of these complete. Abort if total work $\geq 2Tb$.
$\tau \leftarrow$ runtime of m^{th} completed instance
return τ

Algorithm 3 RUNTIMEEST

Inputs: i, τ
Initialize: $j \leftarrow 0$
while True **do**
Sample j^{th} instance J from Γ
Let Y be τ capped runtime of i on J
Update $\bar{Y}$, $\bar{\sigma}^2$, sample mean and variance
$C = c(\bar{\sigma}, n, j, \zeta, \tau)$
if $\bar{Y} - C > T$ **then**
return reject i
end if
$T \leftarrow \min\{T, \bar{Y} + C\}$ ▷ lowest upper confidence
if $C \leq \frac{\varepsilon}{3}(2\bar{Y} - C)$ **then**
return accept i with runtime estimate $\bar{Y}$
end if
$j \leftarrow j + 1$
end while

CapsAndRuns theory

Theorem

With probability $1-\zeta$,

(i) the algorithm finds an (ε,δ)*-optimal configuration;*

(ii) the total work is

$$\tilde{O}_\zeta\left(n\,\mathrm{OPT}_{\delta/2}\left(\frac{1}{\delta}+\max\left\{\frac{\sigma^2}{\max\{\varepsilon^2,\Delta^2\}},\frac{r}{\max\{\varepsilon,\Delta\}}\right\}\right)\right),$$

Refined result

- Gap: $\Delta_i = 1 - \frac{\mathrm{OPT}_{\delta/2}}{R^{\delta}(i)}$.
- Variance of $R(i, j, \tau), j \sim \Gamma$: $\sigma_\tau^2(i)$.
- Maximum relative variance: $\hat{\sigma}^2(i) = \sup_{\tau \in [t_\delta(i), t_{\delta/2}(i)]} \frac{\sigma_\tau^2(i)}{R_\tau^2(i)}$.
- Relative range $r(i) = \sup_{\tau \in [t_\delta(i), t_{\delta/2}(i)]} \frac{\tau}{R_\tau(i)}$.
- Among the set of configurations $\mathcal{N}_1$ not rejected by QuantileEst, let $i_* = \mathrm{argmin}_{i \in \mathcal{N}_1} R_{\tau_i}(i)$.

Refined result

- Gap: $\Delta_i = 1 - \frac{\mathrm{OPT}_{\delta/2}}{R^\delta(i)}$.
- Variance of $R(i,j,\tau), j \sim \Gamma$: $\sigma_\tau^2(i)$.
- Maximum relative variance: $\hat{\sigma}^2(i) = \sup_{\tau \in [t_\delta(i), t_{\delta/2}(i)]} \frac{\sigma_\tau^2(i)}{R_\tau^2(i)}$.
- Relative range $r(i) = \sup_{\tau \in [t_\delta(i), t_{\delta/2}(i)]} \frac{\tau}{R_\tau(i)}$.
- Among the set of configurations $\mathcal{N}_1$ not rejected by QUANTILEEST, let $i_* = \operatorname{argmin}_{i \in \mathcal{N}_1} R_{\tau_i}(i)$.

Theorem

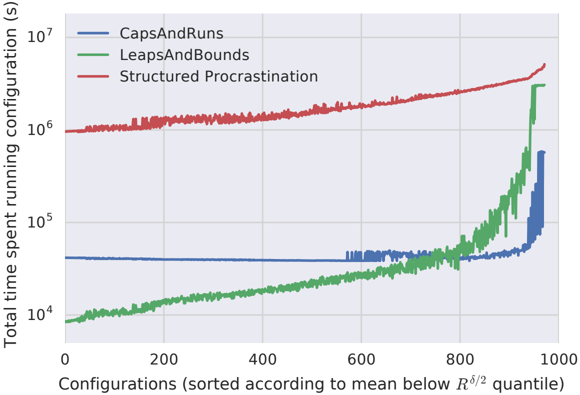
With probability $1 - \zeta$,

(i) *the algorithm finds an* (ε, δ)*-optimal configuration;*

(ii) *the total work is*

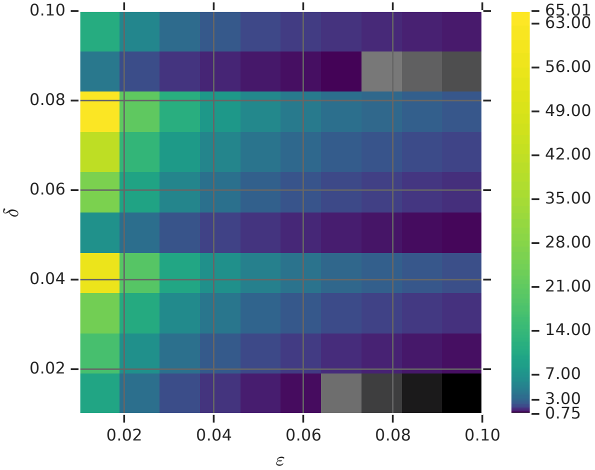
$$\tilde{O}_\zeta\left(\mathrm{OPT}_{\frac{\delta}{2}}\left[\frac{n}{\delta} + \sum_{i \in \mathcal{N}} \max\left\{\frac{\max\left\{\hat{\sigma}^2(i), \hat{\sigma}^2(i_*)\right\}}{\max\{\varepsilon^2, \Delta_i^2\}}, \frac{\max\left\{r(i), r(i_*)\right\}}{\max\{\varepsilon, \Delta_i\}}\right\}\right]\right)$$

Experiments I



STRUCTURED PROCRASTINATION	LEAPSANDBOUNDS	CAPSANDRUNS
20643 (± 5) days	1451 (± 83) days	586 (± 7) days

Experiments II: Speedup compared to LEAPSANDBOUNDS



Recent work (after ICML'19)

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CAPSANDRUNS (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!
- Questions:

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$

- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!

- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$

- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!

- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!
- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)
  - Algorithm takes ζ vs. it returns ζ (‘fixed budget’?)

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!
- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)
  - Algorithm takes ζ vs. it returns ζ (‘fixed budget’?)
  - What guarantees can we get?

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\,n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!
- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)
  - Algorithm takes ζ vs. it returns ζ (‘fixed budget’?)
  - What guarantees can we get?
  - Does the new LCB used by SPC help?

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$

- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!

- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)
  - Algorithm takes ζ vs. it returns ζ (‘fixed budget’?)
  - What guarantees can we get?
  - Does the new LCB used by SPC help?
  - Does it make sense to decrease ζ?

Structured proscrastination with confidence

(Kleinberg et al., 2019)

- Anytime guarantee: With some $c, p > 0$ universal, for any (ε, δ), for $t \geq c\,\mathrm{OPT}_0\, n/(\delta\varepsilon^2)$, SPC returns with probability $1 - ct^{-p}$ a config i such that $R^\delta(i) \leq (1+\varepsilon)\mathrm{OPT}_0$
- Making CapsAndRuns (more) anytime: Fix ε; add outer loop that decreases δ
  - Works because $\mathrm{OPT}_{1/2} \leq \mathrm{OPT}_{1/4} \leq \mathrm{OPT}_{1/8} \leq \cdots \leq \mathrm{OPT}_0$
  - Guarantee against OPT_0 is easier to get
  - Note: optimal configuration can switch back and forth when δ is decreased!
- Questions:
  - Improving guarantee from $1/(\delta\varepsilon^2)$ to problem-dependent $1/\delta + 1/\varepsilon^2$
  - Simultaneous guarantee against $\mathrm{OPT}_{\delta/2}$ for any (ε, δ)
  - Algorithm takes ζ vs. it returns ζ (‘fixed budget’?)
  - What guarantees can we get?
  - Does the new LCB used by SPC help?
  - Does it make sense to decrease ζ?
  - Continuous setting?

Thank you!

References I

R. Kleinberg, K. Leyton-Brown, and B. Lucier. Efficiency through procrastination: Approximately optimal algorithm configuration with runtime guarantees. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2017.

R. Kleinberg, K. Leyton-Brown, B. Lucier, and D. Graham. Procrastinating with confidence: Near-optimal, anytime, adaptive algorithm configuration. arXiv preprint arXiv:1902.05454, 2019.

V. Mnih, C. Szepesvári, and J.-Y. Audibert. Empirical Bernstein stopping. In Proceedings of the 25th international conference on Machine learning, pages 672–679. ACM, 2008.

N. Sorensson and N. Een. Minisat v1. 13-a sat solver with conflict-clause minimization. SAT, 2005 (53):1–2, 2005.

G. Weisz, A. György, and C. Szepesvári. Leapsandbounds: A method for approximately optimal algorithm configuration. In Proceedings of the International Conference on Machine Learning (ICML), 2018.

G. Weisz, A. György, and C. Szepesvári. CapsAndRuns: An improved method for approximately optimal algorithm configuration. In Proceedings of the International Conference on Machine Learning, pages 6707–6715, 2019.