CapsAndRuns: An Improved Method for Approximately Optimal Algorithm Configuration

Gellért Weisz [1] **András György** [1 2] **Csaba Szepesvári** [1 3]

Abstract

We consider the problem of configuring general-purpose solvers to run efficiently on problem instances drawn from an unknown distribution, a problem of major interest in solver autoconfiguration. Following previous work, we focus on designing algorithms that find a configuration with near-optimal expected *capped* runtime while doing the least amount of work, with the cap chosen in a configuration-specific way so that most instances are solved. In this paper we present a new algorithm, CapsAndRuns, which finds a near-optimal configuration while using time that scales (in a problem dependent way) with the optimal expected capped runtime, significantly strengthening previous results which could only guarantee a bound that scaled with the potentially much larger optimal expected *uncapped* runtime. The new algorithm is simpler and more intuitive than the previous methods: first it estimates the optimal runtime cap for each configuration, then it uses a Bernstein race to find a near optimal configuration given the caps. Experiments verify that our method can significantly outperform its competitors.

1. Introduction

For computational problems of major practical interest (satisfiability, planning, etc.) the computing science community has developed a large number of "solvers", which tend to have many configuration parameters. The plethora of solver parameters is explained by the diversity of applications; experience suggests that there is much to be gained by selecting the solver configuration in an application-specific way. The problem of finding the right configuration can then be treated as a learning problem, where one can sample instances from the distribution underlying an application, the learning algorithm can run any configuration on any sampled instance until a timeout of its choice, and the goal is to find a configuration with nearly optimal expected runtime while using the least amount of time during the search. (Related, but different problems are considered, e.g., by Luby et al., 1993; Adam, 2001; Mnih et al., 2008; Audibert & Bubeck, 2010; György & Kocsis, 2011; Li et al., 2016). There has been much practical success on designing such black-box configuration search methods, especially in the context of satisfiability problems. Although there is plenty of empirical evidence "proving" the utility of methods such as SMAC (Hutter et al., 2011; 2013), ParamILS (Hutter, 2007; Hutter et al., 2009), GGA (Ansótegui et al., 2009; 2015), and irace (Birattari et al., 2002; López-Ibánez et al., 2011), little effort went into creating a theory that would explain why and when one should expect one of these methods to outperform another one.

In their recent paper, Kleinberg et al. (2017) initiated a line of research with the goal to reach this understanding. In their work, they suggest to take a step back and consider an unstructured version of the problem which can be thought to be at the heart of algorithm autoconfiguration. For this setting, they presented a general-purpose configuration optimizer called Structured Procrastination, with guarantees both on how close to the optimal configuration the algorithm's result is, and how long it takes to find such a configuration. They also demonstrated that the gap between worst-case runtimes of existing algorithms (SMAC, ParamILS, GGA, irace) and their solution can be arbitrarily large; a vivid justification of the need for a theoretical approach. Their algorithm, Structured Procrastination, attempts to refine the runtime guarantee for the empirically fastest solver and solves tasks in increasing order of difficulty, postponing difficult tasks until all simpler tasks have been solved.

Another key innovation of Kleinberg et al. (2017) is the 'proper' problem formulation. In particular, they suggest that algorithms should aim to minimize the expected capped

[1] DeepMind, London, UK. [2] On leave from Imperial College London, London, UK. [3] On leave from University of Alberta, Edmonton, AB, Canada. Correspondence to: Gellért Weisz <gellert@google.com>, András György <agyorgy@google.com>, Csaba Szepesvári <szepi@google.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

runtime where for each configuration, the cap is set to a quantile of a fixed order. This objective expresses the view of a user who faces problems when runtime distributions are often heavy-tailed and, as such, is willing to skip a fixed percentage of the instances to gain on the number of instances that can be solved in a given unit time. The main result of their paper is an upper bound on the total runtime of their proposed autoconfiguration methods (which matches their worst-case lower bound up to a logarithmic factor) as a function of the expected *uncapped* runtime of the *optimal* configuration.

In a recent paper (Weisz et al., 2018), we extended the results of Kleinberg et al. (2017), by providing an arguably simpler algorithm (LEAPSANDBOUNDS) with better runtime guarantees while considering a broader class of problems (we removed their assumption on the finiteness of the maximum runtime). We also presented instance-dependent runtime bounds that showed that LEAPSANDBOUNDS finishes faster if the runtime of the configurations over different problem instances has small variance.

In this paper we improve this line of work in two ways: (i) we present an algorithm that is simpler and more intuitive than both STRUCTURED PROCRASTINATION and LEAPSANDBOUNDS, and (ii) we provide a problem-dependent upper bound on the runtime of the algorithm that is significantly better than the previous ones. In particular, the runtime scales with the expected *capped* runtime of the best configuration, rather than with the expected *uncapped* runtime, making the algorithm the first that essentially matches a previous lower bound. This also means that the algorithm is essentially unimprovable in a strong sense. Experiments on configuring the open-source `minisat` solver (Sorensson & Een, 2005) on a randomly generated set of SAT problems (Weisz et al., 2018) demonstrate the advantages of CAPSANDRUNS over STRUCTURED PROCRASTINATION and LEAPSANDBOUNDS.

The rest of the paper is organized as follows: The problem is introduced formally in Section 2. Our algorithm is presented in Section 3, together with its theoretical performance guarantees. The proof of the latter is deferred to Section 4. Experiments are presented in Section 5, while conclusions are drawn and future work is discussed in Section 6.

2. The model

In this section we present the formal definition of the problem we consider, based on the work of Kleinberg et al. (2017). The algorithm configuration problem is defined by a triplet $(\mathcal{N}, \Gamma, R)$: Here, $\mathcal{N}$ is a family of configurations and Γ is a distribution over the set $\mathcal{J}$ of input instances.[1] For now, we consider the case when $\mathcal{N}$ is a finite set, and enumerate the configurations as $\mathcal{N} = [n]$.[2] For configuration $i \in \mathcal{N}$ and instance $j \in \mathcal{J}$, $R(i,j) \in [0,\infty]$ is the runtime of configuration i on instance j. For simplicity, we will assume that for any $i \in \mathcal{N}$, $R(i,J)$ has a continuous distribution when J is drawn from Γ; extending the results to other distributions (e.g., discrete) are straightforward but significantly complicates the notation.

We let $R(i) = \mathbb{E}_{J\sim\Gamma}\left[R(i,J)\right]$ denote the average runtime of configuration i on instances distributed according to Γ, and define $\mathrm{OPT} = \min_i\{R(i)\}$ as the mean runtime of an optimal configuration. *The goal is defined to find such an optimal, or at least a nearly optimal configuration while spending as little time as possible–proportional to the runtime of the optimal configuration–on this task.* For this, a search algorithm can *(i)* sample instances J at random from Γ; *(ii)* enumerate the configurations in $\mathcal{N}$; *(iii)* run a configuration i on an instance j until it finishes, or the execution time exceeds a fixed timeout $\tau \geq 0$, chosen by the search algorithm. Practically, this means observing $R(i,j,\tau) \doteq \min(R(i,j),\tau)$ after time $R(i,j,\tau)$, and also whether the calculation has finished with a solution or it timed out.

The main difficulty in organizing the search is that some configurations may take a long, or even infinite time to execute on some instances. Since an algorithm that claims to *find* a near-optimal configuration must verify that no other configuration can finish significantly faster than the chosen configuration, the total runtime is at least proportional to $n \times \mathrm{OPT}$, where $n = |N|$ is the number of configurations to be tested. Since knowing the mean runtime up to a multiplicative accuracy of $(1+\varepsilon)$ requires $1/\varepsilon^2$ samples even when the runtime distributions are light-tailed, relaxing the requirement to find a configuration i with runtime $R(i) \leq (1+\varepsilon)\mathrm{OPT}$, we get that the total runtime is at least $\Omega(n \times \mathrm{OPT}/\varepsilon^2)$. The situation worsens for heavy-tailed runtime distributions: If the runtime of an algorithm is $b > 1$ with probability $1/b$ and 0 otherwise, with b unknown, all sampling methods need to see at least one positive runtime to estimate the expected runtime up to any fixed accuracy. Thus, any sampling method needs to use at least $\Omega(b)$ time, despite that the expected runtime is constant. This implies that in the face of heavy-tailed runtime distributions, the runtime of any sound configuration search algorithm would be unbounded in the worst-case, regardless the value of OPT, n and ε. Since heavy tailed runtime distributions are quite common in practice, rather than constraining the problem by ruling these out, following Kleinberg et al. (2017), we relax the search criterion to that of finding an (ε,δ)-optimal configuration.

To this end, for any $\tau \geq 0$, we introduce the τ-capped version of $R(i)$ as $R_\tau(i) = \mathbb{E}_{J\sim\Gamma}\left[R(i,J,\tau)\right]$. Also, for any

[1] For randomized solvers, input instances can mean (input instance, random seed) pairs.

[2] For any positive integer a, we define $[a] = \{1,\ldots,a\}$.

$\delta \in [0, 1]$, define the δ-quantile of configuration i as $t_\delta(i) = \inf_{t\in\mathbb{R}}\{t : \Pr_{J\sim\Gamma}(R(i, J) > t) \le \delta\}$. Note that since $R(i, J)$ has a continuous distribution, $\Pr_{J\sim\Gamma}(R(i, J) > t_\delta(i)) = \delta$.

We define the mean runtime below the δ-quantile as $R^\delta(i) = R_{t_\delta(i)}(i)$, and $\mathrm{OPT}_\delta = \min_i R^\delta(i)$. Kleinberg et al. (2017) defined a configuration to be (ε, δ)-optimal if $R_\tau(i^*) \le (1 + \varepsilon)\mathrm{OPT}$, and $\Pr_{J\sim\Gamma}(R(i^*, J) > \tau) \le \delta$. Our goal would be to change OPT to OPT_δ; however, one can show that the resulting property would be impossible to verify with high probability. However, allowing a slight slack by using $\mathrm{OPT}_{\alpha\delta}$ instead for some $0 < \alpha < 1$ makes the definition tractable. For simplicity, we choose $\alpha = 1/2$, which gives rise to the following definition:

Definition 1 ((ε, δ)-optimality)**.** *A configuration i is (ε, δ)-optimal if $R^\delta(i) \le (1 + \varepsilon)\mathrm{OPT}_{\delta/2}$. Otherwise, we say i is (ε, δ)-suboptimal.*

In words, given (ε, δ), a sound configuration search algorithm must find a configuration i and a runtime cap τ such that the configuration's τ-capped mean runtime is at most $(1+\varepsilon)\mathrm{OPT}_{\delta/2}$, with τ at least as large as the δ-quantile of the runtime distribution of configuration i.

3. The algorithm

In this section we describe our algorithm, CapsAndRuns, to find an (ε, δ)-optimal configuration. CapsAndRuns works in two phases: In the first phase, for each configuration $i \in \mathcal{N}$, we estimate a runtime cap τ_i that ensures that i can solve at least $1 - \delta$ fraction of the instances within time τ_i. This is implemented in QuantileEst in Algorithm 2: b random instances are solved in parallel,[3] and the runtime cap is selected so that $1 - \frac{3}{4}\delta$ fraction of the instances are completed. This ensures (with the proper choice of b), that with high probability, $t_\delta(i) \le \tau_i \le t_{\delta/2}(i)$ and hence $R^\delta(i) \le R_{\tau_i}(i) \le R^{\frac{\delta}{2}}$.

Once the caps are found, in the second phase of the algorithm (RuntimeEst given in Algorithm 3) we solve random instances one by one for each configuration i, while continuously updating upper and lower estimates on the expected runtime. In particular, with high probability, the difference of the average capped runtime $\bar{Y}_j(i)$ and the expected capped runtime $R_{\tau_i}(i)$ can be bounded as $|R_{\tau_i}(i) - \bar{Y}_j(i)| \le C_{i,j}$ for all i and j where $C_{i,j}$ is defined in Algorithm 3.

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 $\Gamma$
$b \leftarrow \left\lceil \frac{48}{\delta} \log\left(\frac{3n}{\zeta}\right) \right\rceil$
$T \leftarrow \infty$          ▷ 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 $(i, \tau_i)$, abort if $|\mathcal{N}'| = 1$
        if RuntimeEst $(i, \tau_i)$ rejected $i$ then
            Remove $i$ from $\mathcal{N}'$
        else
            $\bar{Y}(i) \leftarrow$ return value of RuntimeEst$(i, \tau_i)$
        end if
    end if
end for
return $i^* = \mathrm{argmin}_{i\in\mathcal{N}'} \bar{Y}(i)$ and $\tau_{i^*}$.
```

The final ingredient of the algorithm is that we eliminate configurations that are redundant (i.e., we can be certain that there are other (ε, δ)-optimal configurations): We maintain a global upper estimate T of $\min_i R_{\tau_i}(i)$, which is continuously updated by each estimation process and is used to (i) stop searching for the cap in phase one if a configuration i is too slow, and (ii) eliminate configurations in the second phase if their average runtime estimate is too high. (i) ensures that if a cap is not found in $O(b\mathrm{OPT}_{\delta/2})$ total work, we can exclude that configuration, implying that the total work in Phase I (QuantileEst) is at most $O(bn\mathrm{OPT}_{\delta/2})$. Phase II (RuntimeEst) essentially implements a slightly modified version of the Bernstein race of Mnih et al. (2008), and our analysis is based on theirs.

In order to be able to state the resulting performance guarantees for CapsAndRuns, we need to introduce a few definitions. Let $\bar{\mathcal{N}}_1$ and $\bar{\mathcal{N}}_2$ denote the set of configurations rejected by CapsAndRuns in Phase I (Line 3 of QuantileEst) and Phase II (Line 8 of RuntimeEst), respectively. Let $i_* = \mathrm{argmin}_{i\in\mathcal{N}\setminus\bar{\mathcal{N}}_1} R_{\tau_i}(i)$; note that i_* is an essentially optimal configuration with high probability, as evidenced by Lemma 10. The performance of CapsAndRuns depends on the individual behavior of the different configurations. To quantify this depen-

[3] Emulating parallelization on a single CPU can be implemented in a straightforward way if stopping and resuming jobs is possible. If this is problematic (e.g., due to memory constraints), it can be avoided by restarting from scratch every time while doubling the maximum runtime, which only results in a factor of 2 slowdown.

$$\mathcal{O}\left(\mathrm{OPT}_{\frac{\delta}{2}}\left[\frac{n}{\delta}\log\frac{n}{\zeta}+\sum_{i\in\mathcal{N}}\max\left\{\frac{\max\{\hat{\sigma}^2(i),\hat{\sigma}^2(i_*)\}}{\max\{\varepsilon^2,\Delta_i^2\}},\frac{\max\{r(i),r(i_*)\}}{\max\{\varepsilon,\Delta_i\}}\right\}\left(\log^2\frac{n}{\zeta}+\log\frac{n}{\zeta}\log\frac{\max\{r(i),r(i_*)\}}{\max\{\varepsilon,\Delta_i^2\}}\right)\right]\right). \quad (1)$$

dence, for any $i \in \mathcal{N}$, define the suboptimality gap as $\Delta_i = 1 - \frac{\mathrm{OPT}_{\delta/2}}{R^\delta(i)}$. Furthermore, let $\sigma_\tau^2(i)$ denote the variance of $R(i, J, \tau)$, $J \sim \Gamma$, and define the maximum relative variance as $\hat{\sigma}^2(i) = \sup_{\tau\in[t_\delta(i),t_{\delta/2}(i)]} \frac{\sigma_\tau^2(i)}{R_\tau^2(i)}$ and relative range as $r(i) = \sup_{\tau\in[t_\delta(i),t_{\delta/2}(i)]} \frac{\tau}{R_\tau(i)}$. Now we are ready to present our main result (its proof is given in the next section).

Theorem 1. *For a failure parameter $\zeta \in (0, 1/6)$, with probability at least $1 - 6\zeta$, CAPSANDRUNS finds an (ε, δ)-optimal configuration with total work given in Eq. (1) on the top of the page.*

Kleinberg et al. (2017) showed that in the worst-case the total work to find an (ε, δ)-optimal solution (with OPT_∞ instead of $\mathrm{OPT}_{\delta/2}$) must be $\Omega(\frac{n\mathrm{OPT}}{\varepsilon^2\delta})$. Our bound shows that depending on the runtime distributions, one can have much better guarantees. In particular, our bound scales with the gaps Δ_i instead of ε. Furthermore, the $1/\varepsilon^2$ term is multiplied with the relative variance, which can be quite small, and only $1/\varepsilon$ multiplies the much bigger range. Furthermore, we separated the effects of ε and δ, avoiding the appearance the very large $(\varepsilon^2\delta)^{-1}$. Nevertheless, in the worst case, $\hat{\sigma}^2(i) \le \sup_{\tau\in[t_\delta(i),t_{\delta/2}(i)]} \frac{\tau}{R_\tau(i)} \le \frac{2}{\delta}$, matching the worst-case lower bound up to logarithmic factors.

In this paper we assume that the number of configurations is finite, which might seem like a big limitation since, in practice, parameter spaces are usually (uncountably) infinite. On the other hand, we can simply adopt the random sampling idea of Kleinberg et al. (2017) to overcome this difficulty: Under mild assumptions on the set of configurations, one can sample a large enough number n of configurations that guarantees that at least one of the best $\gamma \in (0, 1)$ fraction of configurations is selected with high probability (depending on n and γ). For example, if configurations can be sampled uniformly at random, the aforementioned probability is $1 - (1 - \gamma)^n$. Then, running CAPSANDRUNS over this set of configurations yields a configuration whose expected capped runtime is guaranteed to be ε-close to that of the top γ-fraction of the configurations.

4. Proof of the main theorem

In this section we present the proof of Theorem 1. This is done through a sequence of several lemmas. First we analyze QUANTILEEST (Phase I), then RUNTIMEEST (Phase II). In both cases we use concentration inequalities to show that our measurements lead to good estimates with high probability, and we use these results to argue about their correctness and the runtime of the algorithms.

In these lemmas we show that, with high probability, the measurements concentrate around their expectations. This is then used to show that QUANTILEEST is correct and is aborted for slow configurations. Next we follow the analysis of the Bernstein race, also based on concentration of measure arguments (Mnih et al., 2008; Mnih, 2008; Loh & Nowozin, 2013), to show that RUNTIMEEST returns an (ε, δ)-optimal configuration and to bound its total runtime.

We start by introducing some notation. Note that QUANTILEEST either finishes normally and returns with τ_i (for configuration i), or it aborts and rejects i. To be able to reason about τ_i even when QUANTILEEST aborts, we define τ_i to be the value QUANTILEEST would return without the abort option.

Recall that $b = \left\lceil \frac{48}{\delta} \log\left(\frac{3n}{\zeta}\right) \right\rceil$ is the number of instances run by QUANTILEEST for each configuration. Let $X_\tau(i, j)$ denote the runtime, capped at τ, of executions started by QUANTILEEST (i.e., in Phase I), where $i \in [n]$ refers to the configuration and $j \in [b]$ indexes the instances selected by QUANTILEEST to be run with configuration i. Let $\bar{X}_\tau(i) = \frac{1}{b}\sum_{j\in[b]} X_\tau(i, j)$ be the empirical average of capped runtime measurements. Note that only $\bar{X}_t(i)$ for all $t \le \tau_i$ are observed. Let $X^\delta(i, j) = X_{t_\delta(i)}(i, j)$.

Let $Y(i, j)$ denote the τ_i-capped runtime of the j^{th} execution started by RUNTIMEEST (in Phase II), noting that these are independent and identically distributed for all j. Let $\bar{Y}_j(i) = \frac{1}{j}\sum_{t\in[j]} Y(i, t)$ be their average up to the j^{th} measurement. Let $\bar{Y}(i)$ denote $\bar{Y}_j(i)$ for the final measurement j performed in Phase II before returning.

4.1. Analysis of QUANTILEEST

Our first result in this section shows that if QUANTILEEST finishes then the estimated runtime cap τ_i is reasonable. Below, and in the rest of the paper, for any event A, A^c denotes its complement.

Lemma 2 (Correctness of QUANTILEEST)**.** *For any configuration $i \in \mathcal{N} \setminus \bar{\mathcal{N}}_1$, define $E_{1,i} = \{t_\delta(i) \le \tau_i < t_{\delta/2}(i)\}$. Then $\Pr(E_{1,i}^c) \le \frac{\zeta}{n}$.*

The proof of the lemma is based on standard concentration results (in particular, the Chernoff bound), and is given in Appendix A.

The result above depends on the specific choice of b. Taking

Algorithm 2 QUANTILEEST

```
1: Inputs: $i$
2: Initialize: $m \leftarrow \lceil (1 - \frac{3}{4}\delta) b \rceil$
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 $\tau$
```

Algorithm 3 RUNTIMEEST

```
1: Inputs: $i$, $\tau_i$
2: Initialize: $j \leftarrow 0$
3: while True do
4:     Sample $j^{th}$ instance $J$ from $\Gamma$
5:     $Y_{i,j} \leftarrow$ runtime of config $i$ on instance $J$, with timeout $\tau_i$
6:     Compute the sample mean $\bar{Y}_j(i)$ and variance $\bar{\sigma}^2_{i,j}$ of $\{Y_{i,1}, \dots, Y_{i,j}\}$, and the confidence-interval width
       $$C_{i,j} = \bar{\sigma}_{i,j} \sqrt{\frac{2 \log(\frac{3nj(j+1)}{\zeta})}{j}} + \frac{3\tau_i \log(\frac{3nj(j+1)}{\zeta})}{j} .$$
7:     if $\bar{Y}_j(i) - C_{i,j} > T$ then
8:         return reject $i$
9:     end if
10:    if j=b then
11:        $T \leftarrow \min\{T, 2\bar{Y}_j(i)\}$.
12:    end if
13:    $T \leftarrow \min\{T, \bar{Y}_j(i) + C_{i,j}\}$    ▷ lowest upper confidence
14:    if $C_{i,j} \leq \frac{\varepsilon}{2+2\varepsilon} \bar{Y}_j(i)$ then
15:        return accept $i$ with runtime estimate $\bar{Y}_j(i)$.
16:    end if
17:    $j \leftarrow j + 1$
18: end while
```

measurements on b instances in our problem also guaranties that the expected runtime can be estimated up to constant accuracy; this is described in the next lemma.[4] Below we refer to measurements by Z (as introduced in the lemma), and we will instantiate this result with measurements performed both in Phase I and Phase II. The lemma is a simple consequence of the more general Lemma 14 given in the appendix.

Lemma 3. *Let τ be a constant satisfying $0 < \tau < t_{\delta/2}(i)$, and let $Z_\tau(i,j)$, $j \in [b]$, be b runtime measurements of configuration i with timeout τ. Let $\bar{Z}_\tau(i)$ be their average and $R_\tau(i)$ their expectation, and define the event $S_i = \{\frac{1}{2} R_\tau(i) \leq \bar{Z}_\tau(i) \leq 2R_\tau(i)\}$. Then $\Pr(S_i^c) \leq \frac{\zeta}{n}$.*

For each configuration i, we instantiate the above lemma for the b measurements in Phase I. We do this twice, both with timeouts $t_\delta(i)$ and $t_{\delta/2}(i)$, calling the associated S_i events $S_{1,i}$ and $S_{2,i}$. The measurements corresponding to $t_{\delta/2}(i)$ are $X^{\frac{\delta}{2}}(i,j)$, with average $\bar{X}^{\frac{\delta}{2}}(i)$ and expectation $R^{\frac{\delta}{2}}(i)$. The measurements corresponding to $t_\delta(i)$ are $X^\delta(i,j)$, with average $\bar{X}^\delta(i)$ and expectation $R^\delta(i)$.[5]

For each configuration, we also instantiate Lemma 3 for the first b measurements in Phase II (with timeouts τ_i). We call the associated events $S_{3,i}$. The measurements are denoted by $Y(i,j)$, with average $\bar{Y}_b(i)$ and expectation $R_{\tau_i}(i)$.[6] Let $E_2 = \bigcap_i (S_{1,i} \cap S_{2,i} \cap S_{3,i})$ and $E_1 = \bigcap_i E_{1,i}$. The next lemma provides crude high-probability bounds on the above defined estimates.

Lemma 4. *$P(E_1 \cap E_2) \geq 1 - 4\zeta$. Under $E_1 \cap E_2$, $\frac{1}{2} R^\delta(i) \leq \bar{X}_{\tau_i}(i) \leq 2R^{\frac{\delta}{2}}(i)$ and $\frac{1}{2} R_{\tau_i}(i) \leq \bar{Y}_b(i) \leq 2R_{\tau_i}(i)$.*

Proof. Using a union bound, $P(E_1 \cap E_2) \geq 1 - 4\zeta$ by Lemma 2 and Lemma 3. E_2 guarantees that $\frac{1}{2} R_{\tau_i}(i) \leq \bar{Y}_b(i) \leq 2R_{\tau_i}(i)$, and also that $\frac{1}{2} R^{\frac{\delta}{2}}(i) \leq \bar{X}^{\frac{\delta}{2}}(i) \leq 2R^{\frac{\delta}{2}}(i)$, $\frac{1}{2} R^\delta(i) \leq \bar{X}^\delta(i) \leq 2R^\delta(i)$, and . Since E_1 guarantees that $t_\delta(i) \leq \tau_i(i) \leq t_{\delta/2}(i)$, we also have $\bar{X}^\delta(i) \leq \bar{X}_{\tau_i}(i) \leq \bar{X}^{\frac{\delta}{2}}(i)$, which completes the proof. □

Based on the above, we are ready to prove an upper bound on the runtimes in Phase I (i.e., in QUANTILEEST).

Lemma 5 (Runtime bounds for Phase I)**.** *If $E_1 \cap E_2$ holds, each configuration performs up to $16b\mathrm{OPT}_{\delta/2}$ work in Phase I. For any configuration i that completes Phase I (i.e., $i \in \mathcal{N} \setminus \bar{\mathcal{N}}_1$), $R_{\tau_i}(i) = \mathcal{O}\left(\mathrm{OPT}_{\delta/2} \log\left(\frac{3n}{\zeta}\right)\right)$.*

Proof. For any configuration i' that completes Phase I and b measurements of Phase II, $\bar{X}_{\tau_{i'}}(i') \leq \bar{X}^{\frac{\delta}{2}}(i') \leq 2R^{\frac{\delta}{2}}(i')$ and $\bar{Y}_b(i') \leq 2R_{\tau_{i'}}(i') \leq 2R^{\frac{\delta}{2}}(i')$ by Lemma 4. Let $w = \bar{X}_{\tau_{i'}}(i') + \bar{Y}_b(i')$. After at most bw work, T is set to at most $2\bar{Y}_b(i') \leq 2w$ in Line 11 of RUNTIMEEST. If i' completes this bw work first, $w \leq \min_{\hat{i}} 4R^{\frac{\delta}{2}}(\hat{i}) = 4\mathrm{OPT}_{\delta/2}$ and therefore T is set to at most $8\mathrm{OPT}_{\delta/2}$ after at most $4b\mathrm{OPT}_{\delta/2}$ work. During this time, every other configuration has been evaluated with the same amount of work. Phase I is terminated in Line 3 of QUANTILEEST for any configuration that reaches $16b\mathrm{OPT}_{\delta/2}$ work.

To prove the second part of this Lemma, take any configuration i that completes Phase I. To bound $R_{\tau_i}(i)$, notice that $R^\delta(i)$ has at least a $1 - \delta$ fraction of the runs included in the average, and the timeout τ_i applies to the rest of the runs. Therefore, $R_{\tau_i}(i) \leq R^\delta(i) + \delta\tau_i$. Since each configuration performs up to $16b\mathrm{OPT}_{\delta/2}$ work, $\tau_i \leq 16b\mathrm{OPT}_{\delta/2}$. Furthermore, from Lemma 4 and the first part of the lemma (since we make exactly b measurements), $R^\delta(i) \leq 2\bar{X}_{\tau_i}(i) \leq 32\mathrm{OPT}_{\delta/2}$. Combining

[4] Notice that both b and $t_{\delta/2}(i)$ depend on δ.

[5] Note that we do not actually observe $t_{\delta/2}(i)$, $t_\delta(i)$ or the related measurements.

[6] Note that the $Y_{i,j}$ are independent of τ_i.

these, $R_{\tau_i}(i) \le 32\mathrm{OPT}_{\delta/2} + 16\cdot 48\log\left(\frac{3n}{\zeta}\right)\mathrm{OPT}_{\delta/2} = \mathcal{O}\left(\mathrm{OPT}_{\delta/2}\log\left(\frac{3n}{\zeta}\right)\right)$. □

4.2. Analysis of RuntimeEst

Next we analyze the behavior of RuntimeEst and the total runtime in Phase II. Let $\bar{\sigma}^2_{i,j} = \frac{1}{j}\sum_{t\in[j]}\left(Y(i,j)-\bar{Y}_j(i)\right)^2$ denote the empirical variance of $\{Y(i,j)\}_{j\in[j]}$ after j measurements in Phase II. We will use $C_{i,j}$ defined in Algorithm 3 to create confidence intervals for our estimates (the n and $j(j+1)$ factors in the logarithmic term in $C_{i,j}$ enable taking union bounds over configurations and instances).

The next result shows that $C_{i,j}$ is a valid confidence width, as well as some of its properties. Its proof follows from the analysis of the Bernstein race by Mnih (2008) (Theorem 2 in their paper), and is given in Appendix A.3.

Lemma 6. *There exists an event E_3 with $\Pr(E_3) \ge 1-2\zeta$ and a universal constant C such that under $E_1\cap E_3$,*
(a) for all i, j,

$$|R_{\tau_i}(i) - \bar{Y}_j(i)| \le C_{i,j}\ ; \tag{2}$$

(b) for all configurations i and for all

$$j \ge C\cdot\max\left(\frac{\hat{\sigma}^2(i)}{\alpha^2},\frac{r(i)}{\alpha}\right)\left(\log\frac{2n}{\zeta}+\log\frac{r(i)}{\varepsilon}\right),$$

$$C_{i,j} \le \alpha\bar{Y}_j(i)\ ; \tag{3}$$

(c) for all configurations i such that $\Delta_i > 0$ and for all

$$j \ge C\cdot\max\left(25\frac{\hat{\sigma}^2(i)}{\Delta_i^2},5\frac{r(i)}{\Delta_i}\right)\left(\log\frac{2n}{\zeta}+\log\frac{5r(i)}{\Delta_i}\right),$$

$$C_{i,j} < \frac{\Delta_i}{4}R_{\tau_i}(i), \tag{4}$$

and for all

$$j \ge C\cdot\max\left(25\frac{\hat{\sigma}^2(i_*)}{\Delta_i^2},5\frac{r(i_*)}{\Delta_i}\right)\left(\log\frac{2n}{\zeta}+\log\frac{5r(i_*)}{\Delta_i}\right),$$

$$C_{i_*,j} < \frac{\Delta_i}{4}R_{\tau_{i_*}}(i_*). \tag{5}$$

Let $E = E_1\cap E_2\cap E_3$ be the intersection of all "good" events; note that E guarantees that our average runtime estimates are close to their expectations. Furthermore, from Lemma 4 and Lemma 6, $\Pr(E^c) \le 6\zeta$.

The next lemma shows that if E holds, the time limit T is an upper bound on the smallest expected capped runtime.

Lemma 7 (Consistency of T)**.** *If E holds throughout the execution of the algorithm, T is always an upper bound on the minimum τ_i-capped expected runtime of configurations i not rejected in Phase I. In other words, at every time there exists an $i\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ such that $R_{\tau_i}(i)\le T$.*

Proof. T is only updated during RuntimeEst evaluations of configurations that were not rejected in Phase I. Every time after T is updated, in Line 13 and Line 11 of RuntimeEst, during the evaluation of configuration i, $R_{\tau_i}(i)\le T$. This is because under E, Lemma 6 ensures $R_{\tau_i}(i)\le\bar{Y}_j(i)+C_{i,j}$ for $C_{i,j}$ defined in Algorithm 3, and $R_{\tau_i}(i)\le 2\bar{Y}_j(i)$ according to Lemma 4. □

The next result shows that a configuration is only rejected in QuantileEst if there is a good enough configuration which is not rejected.

Lemma 8. *If E holds, for any configuration i rejected in Phase I (Line 3 in QuantileEst), there exists an $i'\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ such that $R_{\tau_{i'}}(i') < R^{\frac{\delta}{2}}(i)$.*

Proof. If configuration i is rejected in Phase I, then for the actual T, $\bar{X}_{\tau_i}(i)\ge 2T$. By Lemma 4, $\bar{X}_{\tau_i}(i)\le 2R^{\frac{\delta}{2}}(i)$, so $T\le R^{\frac{\delta}{2}}(i)$. By Lemma 7, there exists an $i'\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ such that $R_{\tau_{i'}}(i')\le T\le R^{\frac{\delta}{2}}(i)$. □

The following two lemmas show that i_* is "good enough" and it is not rejected in RuntimeEst.

Lemma 9 (Consistency of RuntimeEst)**.** *If E holds, i_* will not be rejected: $i_*\notin\bar{\mathcal{N}}_2$.*

Proof. If a configuration i is rejected in Phase II (Line 8 in RuntimeEst), then $\bar{Y}_j(i)-C_{i,j}>T$. Since E holds, Lemma 6 implies $R_{\tau_i}(i)\ge\bar{Y}_j(i)-C_{i,j}$. Combining these gives $R_{\tau_i}(i)>T$. Combining with Lemma 7 implies that there exists an $i'\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ such that $R_{\tau_{i'}}(i')\le T<R_{\tau_i}(i)$. Since this cannot hold for $i=i_*$, we obtain $i\ne i_*$. □

Lemma 10 (i_* is good enough)**.** *If E holds, $R_{\tau_{i_*}}(i_*)\le\mathrm{OPT}_{\delta/2}$.*

Proof. Let $i_{\delta/2}=\mathrm{argmin}_{i\in\mathcal{N}}R^{\frac{\delta}{2}}(i)$. We prove that there exists an $i'\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ such that $R_{\tau_{i'}}(i')\le R^{\frac{\delta}{2}}(i_{\delta/2})$. If $i_{\delta/2}\in\mathcal{N}\setminus\bar{\mathcal{N}}_1$ (i.e., it is not rejected), choosing $i'=i_{\delta/2}$ gives the result. If $i_{\delta/2}$ is rejected in Phase I, then the statement immediately follows from Lemma 8 by substituting $i=i_{\delta/2}$. Then, from the definition of i_*, we have $R_{\tau_{i_*}}(i_*)\le R_{\tau_{i'}}(i')\le\mathrm{OPT}_{\delta/2}$. □

The next result bounds the total runtime of any configuration in Phase II.

Lemma 11 (Runtime in Phase II)**.** *If E holds, the evaluation of a configuration i in Phase II stops (is either rejected or accepted) after at most $\mathcal{O}\left(\mathrm{OPT}_{\delta/2}(j'+b)\log\left(\frac{3n}{\zeta}\right)\right)$ work where*

$$j' = C\cdot\max\left(\frac{\max\{\hat{\sigma}^2(i),\hat{\sigma}^2(i_*)\}}{\max\{\frac{\varepsilon^2}{(2+2\varepsilon)^2},\frac{\Delta_i^2}{25}\}},\frac{\max\{r(i),r(i_*)\}}{\max\{\frac{\varepsilon}{2+2\varepsilon},\frac{\Delta_i}{5}\}}\right)$$
$$\cdot\left(\log\frac{2n}{\zeta}+\log\frac{\max\{r(i),r(i_*)\}}{\max\{\frac{\varepsilon}{2+2\varepsilon},\frac{\Delta_i}{5}\}}\right).$$

If i is accepted, then $|R_{\tau_i}(i) - \bar{Y}_j(i)| \le \frac{\varepsilon}{2+\varepsilon} R_{\tau_i}(i)$.

Proof. Due to the definition of j' and Lemma 6, either (3) holds for any $j \ge j'$ with $\alpha = \frac{\varepsilon}{2+2\varepsilon}$, or both (4) and (5) hold for any $j \ge j'$. We investigate these two cases separately.

If (3) holds, RUNTIMEEST returns in Line 14, accepting the configuration after j' samples, and we have $C_{i,j} \le \alpha \bar{Y}_j(i) = \frac{\varepsilon}{2+2\varepsilon} \bar{Y}_j(i)$. Combining with (2), we have

$$\alpha R_{\tau_i}(i) \ge \alpha(\bar{Y}_j(i) - C_{i,j}) \ge C_{i,j}(1-\alpha)$$

implying

$$|R_{\tau_i}(i) - \bar{Y}_j(i)| \le C_{i,j} \le \frac{\alpha}{1-\alpha} R_{\tau_i}(i) = \frac{\varepsilon}{2+\varepsilon} R_{\tau_i}(i). \quad (6)$$

To finish the first case, we need to bound the total amount of work $\bar{Y}_{j'}(i) j'$ (since the algorithm evaluates i on at most j' instances). Since by (6) and Lemma 5, $\bar{Y}_{j'}(i) \le \left(1 + \frac{\varepsilon}{2+\varepsilon}\right) R_{\tau_i}(i) = \mathcal{O}(R_{\tau_i}(i)) = \mathcal{O}\left(\mathrm{OPT}_{\delta/2} \log\left(\frac{3n}{\zeta}\right)\right)$, the total work in the this case is $\mathcal{O}\left(\mathrm{OPT}_{\delta/2} j' \log\left(\frac{3n}{\zeta}\right)\right)$, as desired.

Next we analyze the second case, where both (4) and (5) hold for any $j \ge j'$. We first bound from above the time taken by RUNTIMEEST to reject configuration i in Line 8. Let j_i and j_{i_*} be the number of instances ran in Phase II, with configurations i and i_*, respectively, and consider the first time when both $j_i \ge j'$ and $j_{i_*} \ge j'$. $R_{\tau_{i_*}}(i_*) \le \mathrm{OPT}_{\delta/2}$ (by Lemma 10), and $\Delta_i R_{\tau_{i_*}}(i_*) \le \Delta_i R_{\tau_i}(i) \le R_{\tau_i}(i) - \mathrm{OPT}_{\delta/2} \le R_{\tau_i}(i) - R_{\tau_{i_*}}(i_*)$ as $R^\delta(i) \le R_{\tau_i}(i)$ under E. Thus,

$$\begin{aligned}
T &\le \bar{Y}_{j_{i_*}}(i_*) + C_{i_*,j_{i_*}} \le R_{\tau_{i_*}}(i_*) + 2C_{i_*,j_{i_*}} \\
&< R_{\tau_{i_*}}(i_*) + \frac{\Delta_i}{2} R_{\tau_{i_*}}(i_*) < R_{\tau_{i_*}}(i_*) + \frac{\Delta_i}{2} R_{\tau_i}(i) \\
&\le R_{\tau_{i_*}}(i_*) + \frac{R_{\tau_i}(i) - R_{\tau_{i_*}}(i_*)}{2} \\
&= R_{\tau_i}(i) - \frac{R_{\tau_i}(i) - R_{\tau_{i_*}}(i_*)}{2} \le R_{\tau_i}(i) - \frac{\Delta_i}{2} R_{\tau_i}(i) \\
&< R_{\tau_i}(i) - 2C_{i,j_i} \le \bar{Y}_{j_i}(i) - C_{i,j_i}
\end{aligned}$$

So i is rejected by RUNTIMEEST in Line 8, as soon as both configuration i and i_* have been run on at least j' instances. Since $\Delta_i < 1$, $\bar{Y}_{j'}(i) \le \frac{5}{4} R_{\tau_i}(i) = \mathcal{O}\left(\mathrm{OPT}_{\delta/2} \log\left(\frac{3n}{\zeta}\right)\right)$ by (4) and Lemma 5, and $\bar{Y}_{j'}(i_*) \le \frac{5}{4} R_{\tau_{i_*}}(i_*) = \mathcal{O}\left(\mathrm{OPT}_{\delta/2} \log\left(\frac{3n}{\zeta}\right)\right)$ by (5) and Lemma 10. Thus, configuration i will have been run on at least j' instances after at most $\bar{Y}_{j'}(i) j' = \mathcal{O}\left(\mathrm{OPT}_{\delta/2} \log\left(\frac{3n}{\zeta}\right) j'\right)$ Phase II work, and configuration i_* will have been run on at least j' instances after at most $16b\mathrm{OPT}_{\delta/2}$ Phase I work (by Lemma 5), and $\bar{Y}_{j'}(i_*) j' \le \frac{5}{4} \mathrm{OPT}_{\delta/2} j'$ Phase II work. Configurations i and i_* are evaluated in parallel, so at most $\mathcal{O}\left(\mathrm{OPT}_{\delta/2}(j' + b) \log\left(\frac{3n}{\zeta}\right)\right)$ work is performed for configuration i in Phase II, before the configuration is rejected. □

Our last two lemmas show the correctness of CAPSANDRUNS and bound its runtime, proving Theorem 1.

Lemma 12 (Correctness of CAPSANDRUNS). *If E holds and* CAPSANDRUNS *returns with a configuration I, then I is (ε, δ)-optimal.*

Proof. We wish to show that $R^\delta(I) \le (1+\varepsilon)\mathrm{OPT}_{\delta/2}$.

By Lemma 9, i_* is not rejected. Furthermore, by definition, I is not rejected either. So when the algorithm stops, by Lemma 11, $|R_{\tau_{i_*}}(i_*) - \bar{Y}(i_*)| \le \frac{\varepsilon}{2+\varepsilon} R_{\tau_{i_*}}(i_*)$ and $|R_{\tau_I}(I) - \bar{Y}(I)| \le \frac{\varepsilon}{2+\varepsilon} R_{\tau_I}(I)$. Since I was returned by CAPSANDRUNS, $\bar{Y}(I) \le \bar{Y}(i_*)$. For any configuration i,

$$\begin{aligned}
&\left(1 - \frac{\varepsilon}{2+\varepsilon}\right) R^\delta(I) \le \left(1 - \frac{\varepsilon}{2+\varepsilon}\right) R_{\tau_I}(I) \le \bar{Y}(I) \\
&\le \bar{Y}(i_*) \le \left(1 + \frac{\varepsilon}{2+\varepsilon}\right) R_{\tau_{i_*}}(i_*) \le \left(1 + \frac{\varepsilon}{2+\varepsilon}\right) \mathrm{OPT}_{\delta/2} \,,
\end{aligned}$$

where the last inequality comes from by Lemma 10. Thus,

$$R^\delta(I) \le \frac{1 + \frac{\varepsilon}{2+\varepsilon}}{1 - \frac{\varepsilon}{2+\varepsilon}} \mathrm{OPT}_{\delta/2} = (1+\varepsilon)\mathrm{OPT}_{\delta/2},$$

finishing the proof. □

Lemma 13 (Runtime of CAPSANDRUNS). *If E holds, the total work done by* CAPSANDRUNS *is bounded by* (1).

Proof. Combining the upper bound on the Phase II work for each configuration from Lemma 11 with the Phase I work from Lemma 5, it follows that the total work performed by CAPSANDRUNS for all configurations is bounded by (1). □

5. Experiments

In the experiments we used the same benchmark dataset as in our previous paper (Weisz et al., 2018). The dataset contains runtimes of 972 different configurations of `minisat` on a set of 20118 SAT problems generated using CNFuzzDD,[7] with a timeout of 15 CPU minutes. We simulated runs of STRUCTURED PROCRASTINATION (Kleinberg et al., 2017), LEAPSANDBOUNDS (Weisz et al., 2018),[8] and CAPSANDRUNS (this work), with parameters $\varepsilon = 0.05$, $\delta = 0.2$,

[7] http://fmv.jku.at/cnfuzzdd/

[8] There is a slight mistake in the derivation of LEAPSANDBOUNDS, affecting the stopping criterion in Line 15 of Algorithm 3 in their paper: specifically, instead of $c \le \frac{\varepsilon}{3}\left(\bar{Q} + \mathrm{LB}\right)$, the criterion should be $c \le \frac{\varepsilon}{2+2\varepsilon}\bar{Q}$. The same criterion appears in Line 14 of RUNTIMEEST. We ran the experiments with this correction.

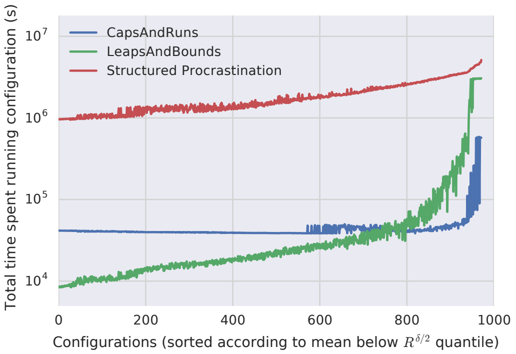

Figure 1: Amount of time the three methods run each configuration, on a logarithmic scale, in an environment that supports resuming runs.

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

Table 1: Total CPU time (in days) for the three methods to find an (ε, δ)-optimal configuration. 95% confidence intervals for LEAPSANDBOUNDS and CAPSANDRUNS were calculated from 10 measurements.

and error probability 0.1 ($\zeta = \frac{1}{60}$), which are the only algorithms we know of that can guarantee finding an (ε, δ)-optimal solution for a given $(\varepsilon, \delta, \zeta)$ triple. In particular, the earlier mentioned heuristic methods (such as SMAC, ParamILS, GGA or irace) do not have such a guarantee and they also need a distance function whose 'proper' selection heavily influences their efficiency, and thus comparing to them would be quite complicated and is outside of the scope of the present paper.

All algorithms returned the same configuration in our simulation, but after a varying amount of computation. Table 1 lists the total CPU runtimes in a simulated environment. Figure 1 shows that both LEAPSANDBOUNDS and CAPSANDRUNS run every configuration for a significantly shorter amount of time than STRUCTURED PROCRASTINATION. Although CAPSANDRUNS runs bad configurations significantly longer than LEAPSANDBOUNDS (because of estimating the caps), it needs to run the hardest configurations for significantly shorter time (observe the logarithmic scale in Figure 1), achieving the shortest total runtime. Fig. 2 shows that CAPSANDRUNS is faster than LEAPSANDBOUNDS for most settings of ε and δ, particularly so for small ε and δ values. Also note that in the cases when LEAPSANDBOUNDS is better, the difference is of the same order as the measurement error, while the advantages of CAPSANDRUNS are much more pronounced, especially when ε and δ are small.

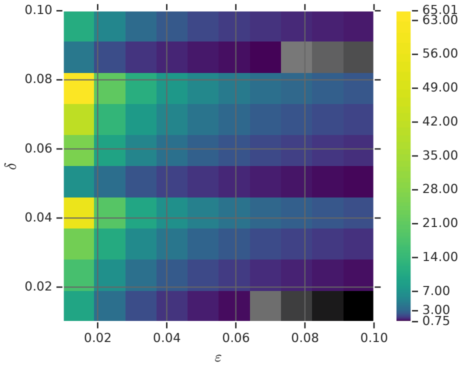

Figure 2: The ratio of the total runtime of LEAPSANDBOUNDS and CAPSANDRUNS for various values of ε and δ. Ratios greater than 1 mean CAPSANDRUNS is faster; values lower than 1 are shown in greyscale.

6. Conclusion and future work

We considered the algorithm configuration problem for a finite number of configurations and a possibly infinite set of random instances with the goal of finding a configuration that can solve a large $(1 - \delta)$ fraction of problem instances with near-optimal capped runtime. We presented an algorithm, CAPSANDRUNS, that finds a near-optimal configuration with high probability. CAPSANDRUNS first estimates the runtime cap for each configuration then performs a Bernstein race over the configurations to find the best one. On the theoretical side, an upper bound is presented on the total work of the algorithm, which significantly improves upon existing runtime bounds (Kleinberg et al., 2017; Weisz et al., 2018), in particular, by improving the dependence on the problem parameters ε and δ, and also matches the worst-case lower bound of Kleinberg et al. (2017). Comparing to existing algorithm configuration methods with theoretical guarantees, our method achieved significant speedup in a recent benchmark problem concerned with configuring SAT solvers.

Potential future work includes modifying our algorithm to incorporate some smoothness assumption on the runtime with respect to some parametrization of the space of configurations, as used in most existing heuristic methods. One idea here is to use ideas borrowed from the bandit literature where similar generalizations have been done. After such an extension, it will be interesting to compare to the existing heuristic methods on existing benchmark problems. Another interesting line of direction is to extend the method to compete with combinations of configurations.

References

Adam, K. Learning while searching for the best alternative. *Journal of Economic Theory*, 101:252–280, 2001.

Ansótegui, C., Sellmann, M., and Tierney, K. A gender-based genetic algorithm for the automatic configuration of algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pp. 142–157. Springer, 2009.

Ansótegui, C., Malitsky, Y., Samulowitz, H., Sellmann, M., and Tierney, K. Model-based genetic algorithms for algorithm configuration. In *IJCAI*, pp. 733–739, 2015.

Audibert, J.-Y. and Bubeck, S. Best arm identification in multi-armed bandits. In *Proceedings of Conference on Learning Theory (COLT)*, 2010.

Audibert, J.-Y., Munos, R., and Szepesvári, C. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902, 2009.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 11–18. Morgan Kaufmann Publishers Inc., 2002.

György, A. and Kocsis, L. Efficient multi-start strategies for local search algorithms. *Journal of Artificial Intelligence Research (JAIR)*, 41:407–444, 2011.

Hutter, F. On the potential of automatic algorithm configuration. In *SLS-DS2007: Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, pp. 36–40, 2007.

Hutter, F., H. Hoos, H., Leyton-Brown, K., and Stützle, T. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1): 267–306, 2009.

Hutter, F., H. Hoos, H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Bayesian optimization with censored response data. *CoRR*, abs/1310.1947, 2013.

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

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016.

Loh, P.-L. and Nowozin, S. Faster hoeffding racing: Bernstein races via jackknife estimates. In *International Conference on Algorithmic Learning Theory*, pp. 203–217. Springer, 2013.

López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. The irace package, iterated race for automatic algorithm configuration. Technical report, Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

Luby, M., Sinclair, A., and Zuckerman, D. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

Mnih, V. Efficient stopping rules. Master's thesis, University of Alberta, Edmonton, AB, Canada, 2008.

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

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

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

A. Technical results

A.1. Proof of Lemma 2

For any $p \in [0,1]$, Let $B^p_{i,j}$ be the Bernoulli random variable indicating whether $X_{\tau_i}(j) \geq t_p(i)$ (value 1) or not (value 0). $\bar{B}^p_i$ is the average of $B^p_{i,j}$. Then, $\mathbb{E}(\bar{B}^\delta_i) = \delta$.

$t_\delta(i) > \tau_i$ means that the runtime of the m^{th} longest run on configuration i, where $m = \lceil (1 - \frac{3}{4}\delta) b \rceil$, is less than $t_\delta(i)$, so at most $b - m$ of the B^δ_j variables are 1. Thus,

$$\bar{B}^\delta_i \leq \frac{b-m}{b} \leq \frac{3}{4}\delta = \frac{3}{4}\mathbb{E}(\bar{B}^\delta_i) \,.$$

By Chernoff's bound, because the $B^\delta_{i,j}$ are i.i.d. Bernoulli random variables,

$$\Pr\left(\bar{B}^\delta_i \leq \frac{3}{4}\mathbb{E}(\bar{B}^\delta_i)\right) \leq \exp\left(-b\delta\frac{1}{32}\right) \leq \frac{\zeta}{3n} \,.$$

Similarly, $\tau_i \geq t_{\delta/2}(i)$ means that at least $b - m$ of the $B^{\frac{\delta}{2}}_j$ variables are 1.

$$\bar{B}^{\frac{\delta}{2}}_i \geq \frac{b-m}{b} = 1 - \frac{\lceil (1 - \frac{3}{4}\delta) b \rceil}{b} \geq \frac{3}{4}\delta - \frac{1}{b} \geq \frac{3}{4}\delta - \frac{1}{32}\delta = \frac{23}{16}\frac{\delta}{2} = \frac{23}{16}\mathbb{E}(\bar{B}^{\frac{\delta}{2}}_i) \,.$$

By Chernoff's bound, because the $B^{\frac{\delta}{2}}_{i,j}$ are i.i.d. Bernoulli random variables,

$$\Pr\left(\bar{B}^{\frac{\delta}{2}}_i \geq \frac{23}{16}\mathbb{E}(\bar{B}^{\frac{\delta}{2}}_i)\right) \leq \exp\left(-b\delta\frac{1}{26}\right) \leq \frac{\zeta}{3n} \,.$$

A union bound completes the proof.

A.2. Generalization of Lemma 3

Lemma 14. *Let τ be a constant satisfying $0 < \tau < t_{\delta/2}(i)$, and let $Z_\tau(i,j)$, $j \in [b]$, be b runtime measurements of configuration i with timeout τ. Let $\bar{Z}_\tau(i)$ be their average and $R_\tau(i)$ their expectation, and define the event $\widehat{S}_i = \{|\bar{Z}_\tau(i) - R_\tau(i)| \leq \widehat{C}_i\}$ where $\widehat{C}_i = \bar{\sigma}_i\sqrt{\frac{2\log(\frac{3n}{\zeta})}{b}} + \frac{3\tau\log(\frac{3n}{\zeta})}{b}$ and $\bar{\sigma}^2_i = \frac{1}{b}\sum_{j\in[b]}\left(Z_\tau(i,j) - \bar{Z}_\tau(i)\right)^2$ is the empirical variance of $\left(Z_\tau(i,j)\right)_j$. Then $\Pr(\widehat{S}^c_i) \leq \frac{\zeta}{n}$.Furthermore, if $\widehat{S}_i$ holds then $\frac{1}{2}R_\tau(i) \leq \bar{Z}_\tau(i) \leq 2R_\tau(i)$.*

The lemma follows from Lemma 15 and Lemma 18 below.

Lemma 15. $\Pr(\widehat{S}^c_i) \leq \frac{\zeta}{n}$.

Proof. The samples $\left(Z_\tau(i,j)\right)_j$ are independent and identically distributed with mean $\bar{Z}_\tau(i)$, expectation $R_\tau(i)$ and range $[0,\tau]$. Thus, the lemma holds by the empirical Bernstein bound (cf. Audibert et al., 2009, Theorem 1 and Appendix B). □

Lemma 16. $\bar{\sigma}^2_i \leq \frac{1}{\delta}(\bar{Z}_\tau(i) + R_\tau(i))^2$.

Proof. First we show that for any $c > 0$,

$$\bar{\sigma}^2_{i,k} \leq c\left(\bar{Z}_\tau(i) + \frac{1}{2c}R_\tau(i)\right)^2 \,. \tag{7}$$

Notice that

$$\begin{aligned}\bar{\sigma}^2_i &= \frac{1}{b}\sum_{j\in[b]}(Z_\tau(i,j) - \bar{Z}_\tau(i))^2 \leq \frac{1}{b}\sum_{j\in[b]} Z_\tau(i,j)^2 \\ &\leq \frac{1}{b}\sum_{j\in[b]} \tau\, Z_\tau(i,j) = \tau\,\bar{Z}_\tau(i) \,.\end{aligned}$$

because $Z_\tau(i,j) \le \tau$ by definition. Now (7) follows from the obvious $\tau\,\bar{Z}_\tau(i) \le c\left(\bar{Z}_\tau(i) + \frac{1}{2c}\tau\right)^2$.

Since $\tau < t_{\delta/2}(i)$, more than $\frac{\delta}{2}$ fraction of the instances time out at τ, so we have $\frac{\delta}{2}\tau \le R_\tau(i)$. Setting $c = \frac{1}{\delta}$, we get $\bar{\sigma}_i^2 \le \frac{1}{\delta}\left(\bar{Z}_\tau(i) + R_\tau(i)\right)^2$. □

Lemma 17. $C_i \le \frac{1}{3}\left(\bar{Z}_\tau(i) + R_\tau(i)\right)$.

Proof. Using Lemma 16 and that $\frac{\delta}{2}\tau \le R_\tau(i)$,

$$\begin{aligned}C_i &= \bar{\sigma}_i\sqrt{\frac{2\log(\frac{3n}{\zeta})}{b}} + \frac{3\tau\log(\frac{3n}{\zeta})}{b}\\ &\le \left(\bar{Z}_\tau(i) + R_\tau(i)\right)\sqrt{\frac{2\log(\frac{3n}{\zeta})}{\delta b}} + R_\tau(i)\frac{6\log(\frac{3n}{\zeta})}{\delta b}\\ &\le \left(\bar{Z}_\tau(i) + R_\tau(i)\right)\left(\sqrt{\frac{2\log(\frac{3n}{\zeta})}{\delta b}} + \frac{6\log(\frac{3n}{\zeta})}{\delta b}\right) .\end{aligned}$$

For $b = \left\lceil \frac{48}{\delta}\log\left(\frac{3n}{\zeta}\right)\right\rceil$, $\sqrt{\frac{2\log(\frac{3n}{\zeta})}{\delta b}} + \frac{6\log(\frac{3n}{\zeta})}{\delta b} \le \frac{1}{3}$. □

Lemma 18. *Under* $\widehat{S}_i$, $\frac{1}{2}R_\tau(i) \le \bar{Z}_\tau(i) \le 2R_\tau(i)$.

Proof. We have from Lemma 15 and Lemma 17, that $|\bar{Z}_\tau(i) - R_\tau(i)| \le \frac{1}{3}\left(\bar{Z}_\tau(i) + R_\tau(i)\right)$. Thus, $\bar{Z}_\tau(i) \ge R_\tau(i) - \frac{1}{3}\left(\bar{Z}_\tau(i) + R_\tau(i)\right)$ and $\bar{Z}_\tau(i) \le R_\tau(i) + \frac{1}{3}\left(\bar{Z}_\tau(i) + R_\tau(i)\right)$. Reordering the inequalities gives the result. □

A.3. Proof of Lemma 6

We first define a good event E_3 (in Lemma 19) that guarantees the accuracy of mean and variance estimates of runtimes. Properties of this event are given in Lemma 19 and Lemma 20. Finally, we prove Lemma 6.

Lemma 19. *Let E_3 be the intersection of the events*

$$|R_{\tau_i}(i) - \bar{Y}_j(i)| \le C_{i,j}\ , \textit{ and} \tag{8}$$

$$\bar{\sigma}_{i,j}^2 \le \sigma_{\tau_i}^2(i) + \sigma_{\tau_i}(i)\sqrt{\frac{2\log(\frac{nj(j+1)}{\zeta})}{j}} + \frac{\log(\frac{nj(j+1)}{\zeta})}{3j}\ , \tag{9}$$

for all configurations i and $j = 1, 2, \ldots$. Then $\Pr(E_3) \ge 1 - 2\zeta$.

Lemma 20. *If (8) and (9) hold for all i, j, then there exists a universal constant C such that for any $\alpha > 0$,*

$$C_{i,j} < \alpha\bar{Y}_j(i) \tag{10}$$

holds for all i and all

$$j \ge C\cdot\max\left(\frac{\sigma_{\tau_i}^2(i)}{\alpha^2 R_{\tau_i}^2(i)}, \frac{\tau_i}{\alpha R_{\tau_i}(i)}\right)\left(\log\frac{2n}{\zeta} + \log\frac{\tau_i}{\alpha R_{\tau_i}(i)}\right)\ . \tag{11}$$

The proofs of these lemmas are based on Bernstein's inequality and can be extracted from the analysis of the Bernstein race by Mnih (2008) (Part 1 of Theorem 2 in their paper). Indeed, Lemma 19 is essentially the same as showing that the events (3.18) and (3.20) in their paper hold with high probability. Condition (10) is a slight variation of the stopping rule of algorithm EBStop of Mnih (2008) (our α is essentially the same as their ε), and the main difference between Lemma 20 and Part 1 of their Theorem 2 is that they only state that the stopping condition holds for the smallest j satisfying (11), but a thorough inspection of their proof shows that the statement also holds for all larger j values. Also note that in our case the range of the runtimes corresponding to configuration i is $[0, \tau_i]$, which is a slight change compared to the range $[0, 1]$ used by Mnih (2008).

Proof of Lemma 6. Let E_3 be defined as in Lemma 19. Then $\Pr(E_3) \ge 1 - 2\zeta$ and (2) follows directly from the lemma (and the definition of E_3). Also, the conditions of Lemma 20 hold under E_3, implying (3) for all i since $\hat{\sigma}(i) \ge \frac{\sigma_{\tau_i}(i)}{R_{\tau_i}(i)}$ and $r(i) \ge \frac{\tau_i}{R_{\tau_i}(i)}$ under E_1 by Lemma 2. If $\Delta_i > 0$, then with $\alpha = \frac{\Delta_i}{5}$, Lemma 19 and Lemma 20 also imply that for all j satisfying the conditions in part (c), $C_{i,j} < \frac{\Delta_i}{5}\bar{Y}_j(i) \le \frac{\Delta_i}{5}(R_{\tau_i}(i) + C_{i,j})$, which then gives $C_{i,j} < \frac{\frac{\Delta_i}{5}}{1-\frac{\Delta_i}{5}} R_{\tau_i}(i) \le \frac{\Delta_i}{4} R_{\tau_i}(i)$ (as $\Delta_i \le 1$ by definition), giving (4) and, similarly, (5). □

B. Empirical Bernstein Bound

Let $X_1, \dots X_t$ be i.i.d. random variables with range R and mean μ. Let the empirical mean be $\bar{X}$ and the empirical variance be $\bar{\sigma}^2 = \frac{1}{t}\sum_{i=1}^{t}(X_i - \bar{X})^2$. Applying Bernstein's inequality to the sum and the sum of the squares of these random variables, we get the empirical Bernstein bound (Audibert et al., 2009), which states that with probability at least $1 - \zeta$,

$$|\bar{X} - \mu| \le \sqrt{\frac{2\bar{\sigma}^2 \log(3/\zeta)}{t}} + \frac{3R\log(3/\zeta)}{t}.$$