PROVABLY EFFICIENT ALGORITHM CONFIGURATION WITH LOCAL STRUCTURE AND REUSE OF OBSERVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of algorithm configuration (AC) under realistic performance landscapes that exhibit local structure, but lack strong global regularity. While recent theoretically grounded AC methods provide PAC-style guarantees, they often rely on global exploration or restrictive assumptions on the reward distributions, limiting their practical efficiency. In this work, we introduce a new AC framework based on a relaxed structural assumption that captures local smoothness of the performance landscape. Under this assumption, we design an algorithm that identifies an ϵ-optimal configuration with probability of at least $(1 - \delta)$ using predominantly local sampling. Our method efficiently reuses problem instances, narrowing the performance gap to heuristic AC methods. A key feature of our approach is the principled and consistent reuse of previously observed evaluations. In contrast to prior methods, this induces an implicit, data-dependent capping mechanism that reduces evaluation cost while preserving statistical guarantees. We provide theoretical bounds on the sample complexity of our algorithm and show that it improves over existing approaches under our structural assumptions. Empirical results further support our theory, demonstrating competitive or superior performance on benchmark configuration tasks. The code and experimental data are available at anonymous.4open.science/r/LoRe-AC-Band-045E.

1 INTRODUCTION

Algorithm Configuration (AC) has become a central paradigm for automatically tuning parameterized algorithms to achieve strong empirical performance across diverse instance distributions (Schede et al., 2022). While the parameters can have a strong influence on the performance or even the correctness of the target algorithm (Hutter et al., 2009b), it is usually a daunting or even infeasible task to tune the potentially huge amount of parameters in large or even infinite spaces by hand. Consequently, there is a great need to automate the search for promising parameters, in particular due to the growing demand for automated algorithmic solutions in industry and academia.

While many practical AC methods have demonstrated impressive results, their theoretical understanding remains limited, particularly in settings where performance landscapes are heterogeneous, noisy and potentially heavy-tailed. Motivated by this, a line of recent works on theoretically grounded AC methods that provide provable guarantees on identifying near-optimal configurations with high probability under specific assumptions on the underlying performance distributions has been published (Kleinberg et al., 2019; Weisz et al., 2020). However, none of these works used an assumption about the structure of the performance landscape since classical structural assumptions like continuity are too strong and not valid in practice anyway.

Existing theoretical approaches typically rely on global exploration strategies or strong structural assumptions, such as independence across configurations or well-behaved runtime distributions. However, in many realistic settings, the performance landscape exhibits local structure, where similar configurations yield similar performance, but without satisfying strong global regularity conditions. Leveraging such structure algorithmically while maintaining theoretical guarantees remains a key challenge. In addition, most of the theoretically grounded AC methods have a performance gaps to heuristic AC methods and are more instance-inefficient due to the forced extensive global exploration.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we show that provably efficient AC does not require global regularity of the performance landscape. We introduce LORE-AC-BAND (**LO**cal structure and **RE**-use of observations), which extends AC-BAND (Brandt et al., 2023) by exploiting a weaker local structural assumption. This enables our algorithm to identify a near-optimal configuration with high probability using local sampling. LORE-AC-BAND systematically reuses previously evaluated problem instances, improving instance efficiency while retaining theoretical guarantees. This reuse induces a data-dependent capping mechanism that avoids unnecessarily expensive evaluations based on information from previous runs. Our approach thus combines weaker structural assumptions, local exploration, and principled reuse of observations to achieve both theoretical guarantees and efficient resource usage.

2 ALGORITHM CONFIGURATION

Problem formulation. We follow the notation from Schede et al. (2022). Let $\mathcal{A}$ be a parameterized algorithm with parameters p_i from a domain Θ_i for each $i \in \{1, \dots, m\}$. Then we call $\mathcal{A}$ the *target algorithm* and denote the space of all feasible parameter configurations for $\mathcal{A}$ as $\Theta = \Theta_1 \times \dots \times \Theta_m$ which we call the *configuration space*. Furthermore, let $\mathcal{I}$ be the space of problem instances that $\mathcal{A}$ can solve, and let $\mathcal{P}$ be an unknown probability distribution over $\mathcal{I}$. To measure the cost of running $\mathcal{A}$ with parameter configuration $\theta \in \Theta$ on a problem instance $i \in \mathcal{I}$, we can observe a real value $c(i, \theta)$ from an unknown, possibly stochastic function $c : \mathcal{I} \times \Theta \to \mathbb{R}$. The most common example for such a cost function is the measured runtime of $\mathcal{A}$ with configuration θ on instance i, but it is also possible to regard other target metrics such as solution quality, memory usage, or a task-specific utility function (Graham et al., 2023), can also be considered. If we consider a duel of a set of configurations in each time step, the cost function, e.g. a binary value whether the considered configuration won the duel or not, can depend on the set of competitors we regard. To cover this potential dependence in our notation, we write the set $\bar{\Theta} \subset \Theta$ as an index to the observed cost function $c_{\bar{\Theta}}(i, \theta)$. We denote in the following the set of all possible subsets of parameter configurations that have at least size 2 and at most size $k \in \mathbb{N}$ as $\Theta_{[2,k]} := \{\bar{\Theta} \subset \Theta | 2 \leq |\bar{\Theta}| \leq k\}$. In addition, let $s_{\theta|\bar{\Theta}}(t) = s(c_{\bar{\Theta}}(i_1, \theta), \dots, c_{\bar{\Theta}}(i_t, \theta))$ be a statistic to aggregate the observed costs for configuration θ on a set of problem instances $i_1, \dots, i_t$ in a comparison set $\bar{\Theta} \subset \Theta$. A classical example for the statistic s is to take the arithmetic average over all observations, i.e., $s_{\theta|\bar{\Theta}}(t) = s(c_{\bar{\Theta}}(i_1, \theta), \dots, c_{\bar{\Theta}}(i_t, \theta)) := t^{-1} \sum_{t'=1}^{t} c_{\bar{\Theta}}(i_{t'}|\theta)$.

As in Li et al. (2016), we **assume** that for each possible set of configurations the statistic of each configuration contained converges against some limit value if we run it on infinitely many instances.

$$(A1) \ : \ \forall \bar{\Theta} \in \Theta_{[2,k]} \forall \theta \in \bar{\Theta} \ : \ S_{\theta|\bar{\Theta}} = \lim_{t \to \infty} s_{\theta|\bar{\Theta}}(t) \text{ exists.}$$

Note that this assumption is rather mild and implicitly given in most AC problems by the i.i.d. setting.

The intuitive **goal** is to find the best configuration θ^* that has the largest set-dependent limit value over all configurations. However, this configuration may not exist, and even if it exists, it is infeasible to find it in a very large or even infinite configuration space. This problem is resolved by relaxing the goal to finding a configuration that is near-optimal. We use the definition from Bengs et al. (2021) and call $\theta \in \Theta$ an **ϵ-optimal configuration** for an $\epsilon > 0$ iff

$$\forall \bar{\Theta} \in \Theta_{[2,k]} \text{ s.t. } \theta \in \bar{\Theta} \ : \ S_{\theta|\bar{\Theta}} \geq S_{(1)|\bar{\Theta}} - \epsilon,$$

where we sort the limits $S_{(1)|\bar{\Theta}} \geq \dots \geq S_{(|\bar{\Theta}|)|\bar{\Theta}}$. To avoid that finding an ϵ-optimal configuration equals the search for a needle in the haystack, we assume as in de Heide et al. (2021) that a specific proportion of ϵ-optimal configurations is contained in Θ.

$$(A2) \ : \ \text{the proportion of } \epsilon\text{-optimal configurations is } \alpha \in (0, 1).$$

Another challenge is that we cannot compare every set in $\Theta_{[2,k]}$, which risks finding only *locally* ϵ-optimal arms in only a finite number of sets $\bar{\Theta} \in \Theta_{[2,k]}$, but not on all possible subsets. Thus, inspired by Brandt et al. (2023), we assume that the probability that a configuration is a global ϵ-optimal configuration increases with the number of sets on which it is a local ϵ-optimal configuration.

$$\begin{aligned}(A3) \ : \forall M \in \mathbb{N}, \forall \theta \in \Theta : \ \mathbb{P}\Big(&(\forall i \in \{1, \dots, M\} \ : \ S_{\theta|\bar{\Theta}_i} \geq S_{(1)|\bar{\Theta}_i} - \epsilon) \\ &\Rightarrow (\forall \bar{\Theta} \in \Theta_{[2,k]} \text{ s.t. } \theta \in \bar{\Theta} \ : \ S_{\theta|\bar{\Theta}_i} \geq S_{(1)|\bar{\Theta}_i} - \epsilon)\Big) \geq 1 - \psi(M),\end{aligned}$$

where $\psi : \mathbb{N} \to [0, 1]$ is a strictly monotone decreasing function and $\bar{\Theta}_1, \dots, \bar{\Theta}_M \sim \text{Unif}(\Theta_{[2,k]})$.

Non-stochastic Combinatorial Bandits. In non-stochastic combinatorial bandits (Brandt et al., 2022), we consider a set of n arms $[n] := \{1, \dots, n\}$ and a fixed subset size $k \in \mathbb{N}$ that represents the maximal number of arms we can compare in parallel. We are also given a fixed budget B, that corresponds to the maximum number of allowed comparisons of arms. Each time a subset of arms $Q \subset [n]$ with $2 \leq |Q| \leq k$ is queried, we observe feedback $o_{i|Q}$ for each arm $i \in Q$, e.g. a numerical reward or a preference-based winner information. Note that this observation does not have to be sampled from an underlying stochastic distribution, which is why this setting is called *non-stochastic*. Similar to above, by means of a given statistic s, the observations of an arm $i \in Q$ are aggregated and mapped to a numerical value representing the performance of arm i.

We assume again that s converges against a limit denoted by $s_{i|Q}(t) \overset{t\to\infty}{\longrightarrow} S_{i|Q}$. The **goal** in a best-arm identification problem in Combinatorial Bandits is to find the *generalized Condorcet winner* (GCW) i^* defined as the arm that has the highest limit statistic in each subset in which it is included. As is common in the literature, e.g., in Bengs et al. (2021), we assume that a GCW exists.

An algorithm configuration problem with finitely many configurations can be formulated as a non-stochastic Combinatorial Bandit problem, treating each configuration as an arm and each observed cost function as bandit feedback. After selecting a finite subset of Θ, the AC problem reduces to best-arm identification, allowing us to apply existing Combinatorial Bandit methods and, typically, transfer their theoretical guarantees.

3 Related Work

The field of AC has evolved in many different directions over the past decades; for a comprehensive overview, see Schede et al. (2022). While heuristic methods based on strategies such as local neighborhood search (e.g., ParamILS (Hutter et al., 2009b)) or genetic approaches (e.g., GGA (Ansótegui et al., 2009) and GGA++ (Ansótegui et al., 2015)) have shown strong empirical performance, they all completely lack theoretical guarantees. Moreover, since the search space in AC typically does not exhibit guaranteed structure (such as continuity), classical structural assumptions like Lipschitz continuity are generally not satisfied in practice and have therefore not been adopted.

Nevertheless, several works address the AC problem from a theoretical perspective. For example, Structured Procrastination with Confidence (Kleinberg et al., 2019) postpones the evaluation of difficult instances, while CapsAndRuns (CAR) and ImpatientCapsAndRuns (iCAR) (Weisz et al., 2019; 2020) estimate configuration-specific timeouts and perform a Bernstein race. Utilitarian AC (Graham et al., 2023) and its follow-up work in Graham et al. (2026) instead evaluate the performance of the configurations via a bounded utility function. In the related setting of Hyperparameter Optimization, the multi-fidelity method Hyperband by Li et al. (2016) builds on Successive Halving by Jamieson & Talwalkar (2016) from the multi-armed bandit literature and extends its theoretical guarantees to its domain. Inspired by these ideas, Brandt et al. (2023) propose AC-Band, a more general AC framework that leverages techniques from the Combinatorial Bandit setting, in particular the *Combinatorial Successive Elimination* (CSE) method from Brandt et al. (2022). As these approaches are central to our work, we explain them in more detail.

A prominent method for AC that our work builds on and improves is the AC-Band algorithm from Brandt et al. (2023). This approach has a theoretical foundation like a derived sufficient budget to guarantee the identification of an ϵ-optimal configuration. This is done by extending and iteratively applying a theoretical grounded method from the field of MABs, called CSE.

CSE. Brandt et al. (2022) solves the best arm identification problem in non-stochastic Combinatorial Bandits using CSE. The main idea is to iteratively (in rounds) partition all arms that are still possible winner candidates into query sets of size k. Then, in each round, each query set is queried a carefully chosen number of times b_r before the worst $f(k)$ arms regarding the statistic s are discarded from each query set. The discarding function $f : [k] \to [k]$ has to be defined by the user, while b_r guarantees that the overall budget is (i) not exceeded, and (ii) distributed evenly among the queried subsets and the number of overall rounds. It is worth mentioning that CSE offers generality: on the one hand, due to the variable discard portion $f(k)$ and the statistic s, and, on the other hand, due to the possibility to consider numerical rewards or preference-based feedback as observations.

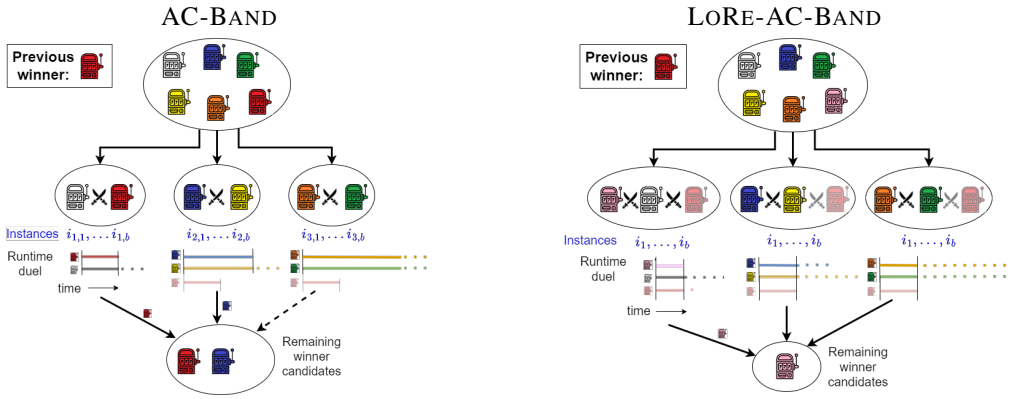


Figure 1: Left: Design of an epoch of AC-Band, in which CSE is called and treats the previous winner in the same way as all other newly sampled configurations. Right: Design of an epoch of LORE-AC-BAND, in which LORE-CSE is called and treats the previous winner as a dummy arm, inducing a capping time for each upcoming duel.

AC-Band. To solve the AC problem the CSE algorithm is extended in Brandt et al. (2023) to handle an infinite number of possible configurations and used as a subroutine in the AC-Band algorithm. The main idea is to call CSE on an epoch-by-epoch basis, sampling new previously unused configurations of Θ in each epoch and using them as input to CSE. The proposed winner from CSE is then used as input for CSE in the next epoch together with new sampled configurations. The size of newly sampled configurations as well as the epoch-wise budget for CSE decrease geometrically over the epochs to reduce the degree of exploration of the configuration space.

4 LORE-CSE

The main idea of our proposed algorithm LORE-CSE is to extend the CSE algorithm to compare each of the arm sets during the run of the algorithm to a predefined performance value of a dummy arm. This can be already observed feedback from a previous run of (LORE-)CSE or a prior guess for a good performance value. This might make sense, e.g., in the case the observed feedback is a runtime of a target algorithm with a specific parameter configuration that corresponds to the notion of an arm here. In the case the chosen subset of parameter configurations we compare is bad, we might have to wait a long time to observe the first configuration to finish, and in this case it makes sense to cap it earlier, as soon as we reach the runtime (= predefined performance value) of the dummy arm. Algorithm 1 gives high-level pseudocode of LORE-CSE. A detailed version is in Appendix A.

Algorithm 1 LORE-CSE

Input: set of arms $[n]$, subset size $k \leq n$, discard portion $f(\cdot)$, statistic of dummy arm s^*
Initialization: possible winner candidates $W \leftarrow [n]$

while $|W| > 1$ **do**
  if $|W| > k$ **then**
    Partition W into non-overlapping subsets S of size k (and if necessary a set with size $|W|$ mod k)
  else
    $S \leftarrow$ Remaining winner candidates W
  for each subset S **do**
    $W \leftarrow W \backslash \{\text{LORE-ArmElimination}(S, f(\cdot), s^*)\}$
return W

5 LORE-AC-BAND

Similar to Brandt et al. (2023), we extend the LORE-CSE algorithm from Algorithm 1 that solves the best-arm identification problem in non-stochastic combinatorial bandits to the AC setting with

Algorithm 2 LoRE-ArmElimination

Input: subset of arms $S \subset [n]$, discard portion $f(\cdot)$, statistic of dummy arm s^*

for $i = 1, \ldots,$ budget_per_subset **do**
  Duel subset S but cap on performance s^*
  if no arm has performed better than s^* **then**
    Consider no winner in S
  Update performance statistics
return worst $f(|S|)$ arms of S

Algorithm 3 ConfigCreation

Input: total number of configurations to be created n, exploration fraction constant $x \in [0, 1)$, initial configuration $\hat{\theta}$, neighborhood size r
Initialization: Configs $= \emptyset$

Configs $\leftarrow$ sample $\theta_1, \ldots, \theta_{\lfloor n \cdot x \rfloor} \sim \text{Unif}(\Theta)$
Configs $\leftarrow$ Configs $\cup$ sample $\theta_{\lfloor n \cdot x \rfloor + 1}, \ldots, \theta_n \sim B_r(\hat{\theta})$
return Configs

Algorithm 4 LoRE-AC-Band

Input: configuration space Θ, budget B, number of initial configurations n_0 and at epoch n_e
Initialization: epoch $e = 0$

Sample configurations $\theta_{0,1}, \ldots, \theta_{0,n_0} \sim \text{Unif}(\Theta)$
Run CSE on $\theta_{0,1}, \ldots, \theta_{0,n_0}$
Store returned winner θ_0^* with its performance statistic s_0^*
while B is not reached **do**
  Start next epoch $e = e + 1$
  Create configurations $\theta_{e,1}, \ldots, \theta_{e,n_e} \leftarrow$ ConfigCreation() around θ_{e-1}^*
  Run LoRE-CSE on $\theta_{e,1}, \ldots, \theta_{e,n_e}$ with arm θ_{e-1}^* and its performance statistic s_{e-1}^*
  if returned winner is better than arm θ_{e-1}^* **then**
    Store returned winner as θ_e^* and its performance statistic s_e^*
  else
    $\theta_e^* \leftarrow \theta_{e-1}^*, s_e^* \leftarrow s_{e-1}^*$
return θ_e

a potential infinitely large configuration space Θ. Unlike classical AC-Band, we do not treat the winner from the previous (LoRE-)CSE run like a newly sampled configuration. Instead, we use it as a dummy arm that acts as a capping time in the next LoRE-CSE call. A high-level description of the LoRE-AC-Band algorithm can be found in Algorithm 4. A more detailed version is given in Appendix A. Figure 1 illustrates how the previous winner is treated differently in AC-Band and LoRE-AC-Band: as a regular sampled configuration in and as a dummy arm, respectively.

5.1 Creation of new configurations

In comparison to the classical AC-Band algorithm, we will not only create new configurations by completely sampling uniformly at random in the configuration space Θ. Instead, we split the creation of new configurations between uniform exploration and intensification in a promising area. We have an exploration constant that corresponds to the proportion of the configurations to be created that are still sampled uniformly at random from Θ. The rest of the configurations are sampled in a local neighborhood of the best configuration we have already seen until now like shown in Algorithm 3. Formally, we first decompose the configuration space into categorical and continuous counterparts as $\Theta = \Theta_{\text{cat}} \times \Theta_{\text{cont}}$. We use the Hamming distance (i.e., the number of mismatches) $\mathbf{d}_H$ on Θ_{cat} and the Euclidean distance on Θ_{cont}, where each continuous variable is normalized to $[0, 1]$. The r-neighborhood around $\hat{\boldsymbol{\theta}}$ is then defined as

$$B_r(\hat{\boldsymbol{\theta}}) = \{\boldsymbol{\theta} \in \Theta \mid w \cdot \mathbf{d}_H(\boldsymbol{\theta}_{\text{cat}}, \hat{\boldsymbol{\theta}}_{\text{cat}}) + (1 - w) \cdot \|\boldsymbol{\theta}_{\text{cont}} - \hat{\boldsymbol{\theta}}_{\text{cont}}\|_2 \leq r\},$$

where $w \in [0, 1]$ is a balancing weight.

Given the current incumbent configuration $\boldsymbol{\theta}$, we generate new configurations in its r-neighborhood by perturbing the continuous and categorical parts jointly. We first draw M points $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M)$, each $\boldsymbol{u} \in [0, 1]^{|\boldsymbol{\theta}_{\text{cont}}|}$ using Latin Hypercube Sampling (LHS) (McKay et al., 1979), and map each to a perturbation within half-width κ_1 of the incumbent: for each $i \in \{1, \ldots, M\}$,

$$\tilde{\boldsymbol{\theta}}_{(i)}^{\text{cont}} = \text{clip}\left(\boldsymbol{\theta}^{\text{cont}} + (2\boldsymbol{u}_{(i)} - 1)\,\kappa_1;\ 0, 1\right).$$

Each candidate inherits the incumbent's categorical values and re-samples at most κ_2 of them at random. A candidate is retained only if it lies within the radius r of $\boldsymbol{\theta}$; otherwise it is discarded and

will be resampled again. Retained candidates are deduplicated against previously seen configurations, and generation stops once the requested number of new configurations is reached. The parameters κ_1, κ_2, and the radius r jointly control the search around the incumbent. This strategy allows us to exploit a promising region of configurations while still ensuring enough exploration in the rest of the configuration space. Under specific assumptions about the structure in the performance landscape, we are able to derive a bound for a good exploration constant x in Section 6.

6 Local Sampling with Probabilistic Continuity

Since structural assumptions about the configuration space Θ are usually not valid in practice, a theoretical foundation of heuristic search methods that rely on such assumptions is still missing. Prominent examples that implicitly assume an underlying structure in the search space are local search algorithms like ParamILS (Hutter et al., 2009a) or evolutionary approaches like GGA (Ansótegui et al., 2009). However, these methods usually perform well in practice, which indicates that a loose structure in the search space does exist in practice. To cover this formally, we will relax the following definition of Lipschitz-continuity to a probabilistic version, such that the cost function for each two configurations is Lipschitz-continuous only with a specific probability in Definition 1.

Definition 1 (Probabilistic Lipschitz-continuity)**.** Let $f : X \to \mathbb{R}$. Then we say that f is γ-*probabilistic Lipschitz-continuous* if and only if there exists a constant $L \in \mathbb{R}$ s.t.

$$\mathbb{P}\left(|f(x) - f(y)| \leq \ L \cdot |x - y|\right) \geq \min\left\{1, \gamma/(|x - y|)\right\} \tag{1}$$

for any two points $x, y \in X$ and a constant $\gamma \in (0, 1)$.

The above definition of γ-probabilistic Lipschitz-continuity allows us to have some sudden jumps in the search space with a specific probability. For two configurations that are close to each other, the probability that the cost function of these two points is continuous is higher than for two configurations that lie further apart. Such assumptions about the structure in the search space were lacking in the theoretical grounded literature in the field of AC. In contrast, they usually assumed just a fixed portion of the search space to be ϵ-optimal like in Assumption 2 and sampled often enough uniformly at random to find one of these near-optimal solutions with high probability, e.g. in Brandt et al. (2023).

Assumption 2 (Proportion of near-optimal configurations)**.** The proportion of ϵ-best configurations in Θ is $\alpha \in (0, 1)$.

If we assume in addition to Assumption 2 a γ-probabilistic Lipschitz-continuity in the search space, we can prove that sampling uniformly at random in the local neighborhood of the best configuration seen until now is superior in comparison to global sampling. The necessary conditions for this theoretical guarantee are discussed in Theorem 3, the proof of which is deferred to Appendix C.

Theorem 3 (Local sampling vs. global sampling)**.** *Assume a γ-probabilistic Lipschitz-continuity in the search space Θ. Let the proportion of $\frac{\epsilon}{2}$-optimal solutions in Θ be $\tilde{\alpha}$ and the proportion of ϵ-optimal solutions α. Then, with probability at least $(1 - \delta)^{\frac{2\gamma L}{\epsilon}}$, the probability to find an ϵ-optimal configuration by local sampling in an r-Ball around the best solution found until now in n samples is higher than with global sampling if*

$$r = \min\left\{\frac{\epsilon}{2L}, \left(1 - \left(\frac{\gamma}{r}\right)^m\right)\frac{\gamma}{\alpha}, \gamma(m+1)^{1/m}\right\} \quad \textit{and} \quad n \geq \frac{\log(\delta)}{\log(1 - \tilde{\alpha})}.$$

Intuitively, we need to see enough random samples n to guarantee that a "close to optimal" solution is found. However, the radius r for local sampling around the best solution seen until now has to be small enough to guarantee a higher concentration of ϵ-optimal configurations in the r-neighborhood than in the global search space. For our proposed ConfigCreation algorithm in Algorithm 3 we can derive the following corollary for the exploration factor. The proof can be found in Appendix C.

Corollary 4. *Assume a γ-probabilistic Lipschitz-continuity, a proportion of ϵ-optimal configurations of α, and of $\frac{\epsilon}{2}$-optimal configurations of $\tilde{\alpha}$. Then, LoRe-AC-Band with an overall number of n_0 configurations and E epochs creates an ϵ-optimal configuration with Algorithm 3 with higher probability than by sampling uniformly at random in Θ with probability at least $(1 - \delta)^{\frac{2\gamma L}{\epsilon}}$ if the exploration factor is larger than*

$$x \geq \frac{1}{n_0\left(2\left(1 - \frac{1}{2^{E+1}}\right) - \frac{3}{2}\right)}\left(\frac{\log(\delta)}{\log(1 - \tilde{\alpha})} - \frac{n_0}{2}\right).$$

7 THEORETICAL GUARANTEES FOR LORE-AC-BAND

Since we do not run the dummy arm again in each subset to save resources, the previously seen observations for it must be stored to be able to compare it to the newly sampled configurations. We call the stored information for the dummy arm $o_1, o_2, \dots$ in the following.

Example. An intuitive way to store the observations of the dummy arm is to compute the arithmetic average of all its observed costs. In this case, the information we use in the next runs of CSE is the same for each sampled problem instance, we compare the dummy arm with its newly sampled competitors. Thus, we have $o_1 = o_2 = \dots$. However, for a huge variance in the observations of the dummy arm, it might be better to learn the distribution $\hat{D}$ from which the observations are sampled and use this estimated distribution to sample $o_1, o_2, \dots \sim \hat{D}$. If enough memory is available, we can simply store the previously observed costs for the dummy arm and look them up if necessary.

Assumption 5. Assume that for a given distribution D over $\mathbb{R}$ $x_1, x_2, \dots x_n \sim D$ is a series of samples and $o_1, o_2, \dots, o_n$ is the previously stored information for the dummy arm. Then for a given $\delta \in (0,1)$, there exists a decreasing sequence $(c_n)_{n\in\mathbb{N}} \in \mathbb{R}$ with $\lim_{n\to\infty} c_n \to 0$, depending on the number of samples $n \in \mathbb{N}$, such that the statistic s fulfills

$$\mathbb{P}\left(\left|s(x_1, x_2, \dots, x_n) - s(o_1, o_2, \dots, o_n)\right| \leq c_n\right) \geq 1 - \delta.$$

However, since the performance values $o_1, o_2, \dots$ are estimated, we need to ensure that the statistic applied to the estimated performance values is close enough to the statistic we would see if we would have real observations of the dummy arm again. This is given with high probability if assumption 5 holds. We can thus guarantee to find an ϵ-optimal arm with high probability with Dummy-CSE shown in Algorithm 5, which is stated in Theorem 6, the proof of which is given in Appendix D.

Theorem 6 (Sufficient budget for LORE-CSE for finding an ϵ-best arm)**.** *Let $R^{\rho,k,n}$ be the number of rounds of a run of* LORE-CSE *for discard portion $\rho \in [0,1]$, subset size $k \in \mathbb{N}$ and number of arms $n \in \mathbb{N}$. Let θ^* be the previously found best arm, and $\tilde{\Delta}_{i|A} = \max\{S_{1|A}, S_{\theta^*|A}\} - \min\{S_{i|A}, S_{\theta^*|A}\}$ and $\tilde{\tau} := \max\{\tau \in \mathbb{N} | \min\{\epsilon, \min_{i\in[n]} \max_{r\in[R]} \tilde{\Delta}_{i|\mathbb{A}_r(i^*)}\} > c_\tau\}$. Using Algorithm 5 with n arms, a discard function f_ρ, and a subset size k returns an arm i^* with probability at least $(1-\delta)^{R_{\rho,k,n}}$, which is an ϵ-best arm if the budget B is larger than*

$$z(\rho,k,n,\epsilon) := R^{\rho,k,n} \max_{r\in[R^{\rho,k,n}]} P_r^{\rho,k,n} \cdot \max\left\{\tilde{\tau}, \max_{r\in[R^{\rho,k,n}]}\left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\max\left\{\frac{\epsilon - c_{\tilde{\tau}}}{2}, \max_{r\in[R^{\rho,k,n}]} \frac{\tilde{\Delta}_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)} - c_{\tilde{\tau}}}{2}\right\}\right)\right)\right\},$$

where $\mathbb{A}_r(i^*)$ is the partition in round r of LORE-CSE *containing i^* and $\tilde{\tau} \in \mathbb{N}$ is random but fixed.*

In comparison to the sufficient budget of the classical CSE algorithm in Brandt et al. (2022), we need a sufficient budget that is larger than $\tilde{\tau}$ in each round of LORE-CSE. With this, we ensure that the difference between the stored information for the dummy arm and the ground truth statistic of the dummy arm is small enough and does not have an influence on the performance estimation of the arms. While the rest of the budget looks similar to the one for CSE, we have the additional subtraction of c_τ in the numerators, which can be seen as the cost for using the stored and potentially noisy statistic instead of running the dummy arm again. However, LORE-CSE needs a smaller sufficient budget than CSE if the budget for CSE is greater than $\tilde{\tau}$ and

$$\tilde{\Delta}_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)} - \Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)} > \max\{\epsilon, c_{\tilde{\tau}}\}$$

or in other words, if the dummy arm has either a more than $\max\{\epsilon, c_{\tilde{\tau}}\}$ faster runtime than the best arm of the n competitors or if the dummy arm has a more than $\max\{\epsilon, c_{\tilde{\tau}}\}$ slower runtime than the $f_\rho(|\mathbb{A}_r(i^*)|)$-best arm. We extend these results to the LORE-AC-BAND algorithm in the following. The proof can be found in Appendix D.

Theorem 7 (Sufficient Budget for LORE-AC-BAND)**.** *Let*

$$m_{\text{LORE-CSE}} := \max\left\{\max_{e\in[E], r\in[R^e]}\left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\max\left\{\frac{\epsilon - c_{\tilde{\tau}}}{2}, \max_{r\in[R^e]} \frac{\tilde{\Delta}_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)} - c_{\tilde{\tau}}}{2}\right\}\right)\right), \tilde{\tau}\right\}, N_{\tilde{\alpha},\delta} = \frac{\log(\delta)}{\log(1-\tilde{\alpha})\left(2x\left(1 - \frac{1}{2^{E+1}}\right) - \frac{3x}{2} + \frac{1}{2}\right)}.$$

Furthermore, let R^e be the number of rounds of LoRe-CSE *in epoch $e \in \{1, \dots, E\}$ and $\mathbb{A}_r(\theta^*)$ the set of the partition in round r of* LoRe-CSE *containing θ^*. Under assumptions $(A1) - (A3)$, $(A5)$, Algorithm 4 used with budget B, a subset size k, and exploration factor x and an $\epsilon/2$-best configuration proportion of $\tilde{\alpha}$ finds an ϵ-best configuration θ^* with probability at least $\min\{1-\delta, 1-\psi((R^E)^{-1})\}$ for $C_1 = \log_q(2), C_2 = 1 + \log_q\left(n_0 + \frac{4n_0}{n_0 - N_{\tilde{\alpha},\delta}}\right), C_3 = \lceil \log_q(k) \rceil$ if $B \geq \frac{n_0 + 2^{E+1}}{k} \cdot \frac{-eC_1 + C_2 + C_3}{2^e} \cdot m_{\text{LoRe-CSE}}$.*

In addition to the differences in the $m_{\text{LoRe-CSE}}$-term and m_{CSE}-term, which are discussed after Theorem 6, LoRe-AC-Band requires fewer samples than AC-Band Brandt et al. (2023) if the following condition on the proportion of $\epsilon/2$-optimal configurations in Θ holds:

$$N_{\tilde{\alpha},\delta} \leq \frac{\log(\delta)}{\log(1-\alpha)} \Leftrightarrow \tilde{\alpha} \geq 1 - \exp\left(\frac{\log(1-\alpha)}{2x\left(1 - \frac{1}{2^{E+1}}\right) - \frac{3x}{2} + \frac{1}{2}}\right).$$

The constant C_2 is smaller than for AC-Band, which leads to a smaller factor in the overall budget.

While both LoRe-AC-Band and utilitarian algorithm configuration Graham et al. (2023) provide PAC-style guarantees for identifying near-optimal configurations, their results are not directly comparable because they optimize different objectives. LoRe-AC-Band operates in the classical AC setting and guarantees near-optimality in expected runtime, whereas utilitarian configuration optimizes expected utility and provides $(\epsilon, \delta, \gamma)$-type guarantees that account for a controlled probability of missing superior configurations. Thus, the two approaches are comparable only at a high level; relating their approximation quality or sample complexity requires additional assumptions linking utility and runtime.

8 Empirical Evaluation

We evaluate LoRe-AC-Band and the baselines on eight surrogate scenarios from AClib covering five target algorithms from AClib, in which the runtimes are predicted by empirical performance models (EPMs) (Hutter et al., 2014; Eggensperger et al., 2018).[1] Unlike tabular runtime data, which are usually restricted to a fixed set of configurations (e.g., in Weisz et al. (2020)), EPMs use random forest regression models to predict the runtime of arbitrary configurations on each instance. The number of instances per scenario is split into training and test sets and ranges from 240 to 1988. Details on the EPMs and scenarios are provided in Appendix B. All baselines are run on the training instances and evaluate the final reported incumbent configuration on 100 random test instances.

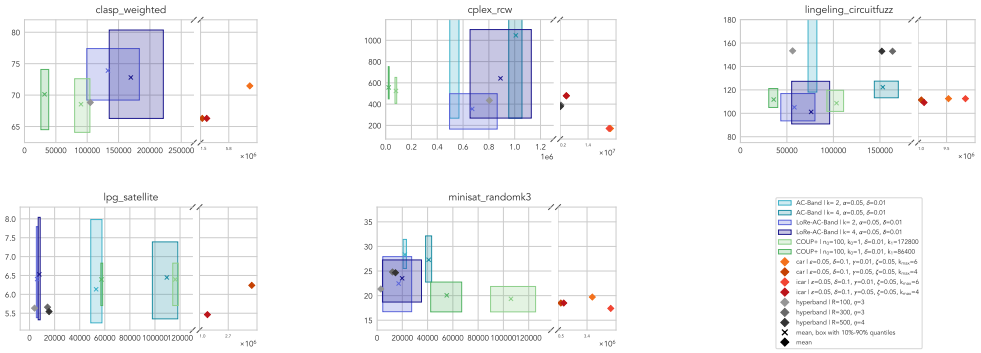
8.1 Baselines

For **AC-Band** (Brandt et al., 2023), we use the maximum number of configurations run in parallel to $k \in \{2, 4\}$ and $\alpha \in \{0.05, 0.01\}$, which determines the number of initial configurations. LoRe-AC-Band uses the same settings. For **Practical COUP (COUP+)** (Graham et al., 2026), we follow its use case and adopt the default Log-Laplace utility function, $n_0 \in \{100, 500\}$ initial configurations, and a total time budget $\kappa_1 \in \{1, 2\}$ day(s). Unlike its variant, our COUP+ does *not* use surrogate models to generate new configurations. For **CAR** and its variant **iCAR** (Weisz et al., 2020), we use $\gamma \in \{0.05, 0.01\}$. Finally, for **HyperBand** (Li et al., 2016), we use $(R, \eta) \in \{(100, 3), (300, 3), (500, 4)\}$ to match the computational budgets of the other baselines. Table 2 (Appendix) lists all parameter settings.

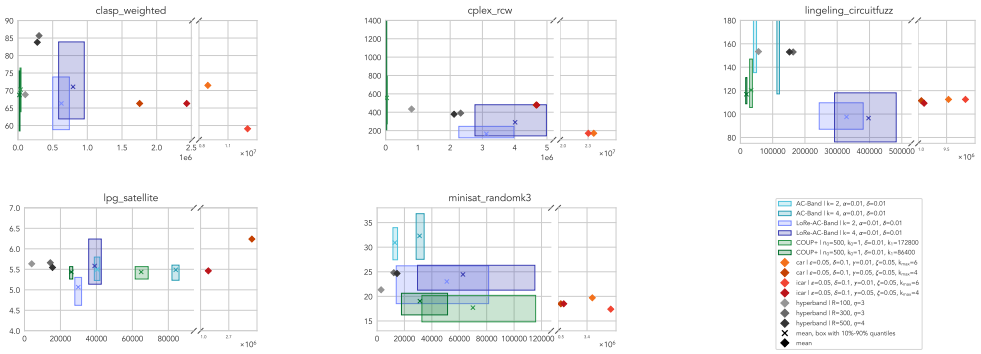
8.2 Comparison with Baselines

Figure 2 reports the total instance CPU time consumed and the incumbent's average runtime on the test instances. We define the total instance CPU time as the sum of the predicted runtimes over all queried configuration–instance pairs. Repeated queries are cached and counted once. We consider two groups, with a smaller (100) and a larger number (500) of initial configurations for baselines.

[1]The datasets are from bitbucket.org/mlindauer/aclib2/src/master/, and the trained models are originally from bitbucket.org/mlindauer/aclib2/src/Surrogates/.



(a) Fewer initial configurations for LoRe-AC-Band (94), COUP+ (100).



(b) More initial configurations for LoRe-AC-Band (463), COUP+ (500).

Figure 2: Total instance CPU time consumed (x-axis) vs. the incumbent's average runtime on the 100 test instances (y-axis) for LoRe-AC-Band (purple), COUP+ (green), AC-Band (cyan), CAR and iCAR (orange), and HyperBand (gray). Partial results of AC-Band are out of scope. Boxes indicate 10%–90% quantiles over 10 seeds. Time unit: seconds (s).

In general, LoRe-AC-Band is competitive with COUP+. In some scenarios, e.g., *lpg_satellite*, LoRe-AC-Band attains better incumbents while requiring less total instance CPU time. LoRe-AC-Band outperforms its predecessor AC-Band. Compared to **iCAR** and **HyperBand**, LoRe-AC-Band consumes less time on instances and gains lower incumbent runtime across the scenarios. LoRe-AC-Band is robust to the choice of its key parameters, though the best choices depend on the target algorithm. The number of instances required by LoRe-AC-Band depends only on the per-round instance budget in LoRe-ArmElimination since instances are reused across rounds. Experimental results suggest that the usage is comparable to or lower than that of COUP+. The full results can be found in Figure 5. Extended experiments are reported in the appendix, including wall-clock time and instance usage comparisons (Section B.3) and sensitivity analysis (Section B.4).

9 Conclusion

We introduced LoRe-AC-Band, an algorithm configuration method that exploits local structure and reuses previous observations. Under a new local structure assumption, we showed that sampling near the incumbent yields an ϵ-optimal configuration with higher probability than uniform sampling, and derived a corresponding lower bound on the exploration factor. Treating the previous winner as a dummy arm with stored observations induces a data-dependent capping mechanism, which allows instance reuse while retaining a sufficient-budget guarantee. On EPM-based surrogate scenarios from AClib, LoRe-AC-Band is competitive with COUP+ and outperforms AC-Band.

REFERENCES

Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In Principles and Practice of Constraint Programming, pp. 142–157. Springer Berlin Heidelberg, 2009.

Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney. Model-Based Genetic Algorithms for Algorithm Configuration. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), IJCAI'15, pp. 733–739. AAAI Press, 2015.

Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. pp. 142–157, 09 2009. ISBN 978-3-642-04243-0. doi: 10.1007/978-3-642-04244-7_14.

Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: a survey. J. Mach. Learn. Res., 22(1), January 2021.

Jasmin Brandt, Viktor Bengs, Björn Haddenhorst, and Eyke Hüllermeier. Finding optimal arms in non-stochastic combinatorial bandits with semi-bandit feedback and finite budget. In Advances in Neural Information Processing Systems, 2022.

Jasmin Brandt, Elias Schede, Viktor Bengs, Björn Haddenhorst, Eyke Hüllermeier, and Kevin Tierney. Ac-band: A combinatorial bandit-based approach to algorithm configuration. In 37th Conference AAAI Conference on Artificial Intelligence, 2023.

Rianne de Heide, James Cheshire, Pierre Ménard, and Alexandra Carpentier. Bandits with many optimal arms. In Proceedings NeurIPS (Annual Conference on Neural Information Processing Systems), pp. 22457–22469, December 2021.

Katharina Eggensperger, Marius Lindauer, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Efficient benchmarking of algorithm configurators via model-based surrogates. Machine Learning, 107(1):15–41, January 2018. ISSN 1573-0565. doi: 10.1007/s10994-017-5683-z.

Devon R. Graham, Kevin Leyton-Brown, and Tim Roughgarden. Utilitarian algorithm configuration. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Devon R. Graham, Eros Rojas Velez, and Kevin Leyton-Brown. Practical, utilitarian algorithm configuration. Proceedings of the AAAI Conference on Artificial Intelligence, 40(43):36937–36945, Mar. 2026. doi: 10.1609/aaai.v40i43.41021.

Frank Hutter, Holger Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An Automatic Algorithm Configuration Framework. Journal of Artificial Intelligence Research, 36:267–306, 10 2009a. doi: 10.1613/jair.2861.

Frank Hutter, Holger Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An Automatic Algorithm Configuration Framework. J. Artif. Intell. Res. (JAIR), 36:267–306, 10 2009b.

Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence, 206:79–111, 2014.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert (eds.), Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, volume 51 of Proceedings of Machine Learning Research, pp. 240–248, Cadiz, Spain, 09–11 May 2016. PMLR.

Robert D. Kleinberg, Kevin Leyton-Brown, Brendan Lucier, and Devon R. Graham. Procrastinating with Confidence: Near-Optimal, Anytime, Adaptive Algorithm Configuration. In Neural Information Processing Systems (NeurIPS), 2019.

Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. Singularity: Scientific containers for mobility of compute. PloS one, 12(5):e0177459, 2017.

Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In International Conference on Learning Representations, 2016.

M. McKay, Richard Beckman, and William Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 21: 239–245, 05 1979. doi: 10.1080/00401706.1979.10489755.

Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. A survey of methods for automated algorithm configuration. Journal of Artificial Intelligence Research, 75:425–487, 2022.

Gellert Weisz, Andras Gyorgy, and Csaba Szepesvari. CapsAndRuns: An improved method for approximately optimal algorithm configuration. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 6707–6715. PMLR, 2019.

Gellért Weisz, András György, Wei-I Lin, Devon Graham, Kevin Leyton-Brown, Csaba Szepesvári, and Brendan Lucier. Impatientcapsandruns: approximately optimal algorithm configuration from an infinite pool. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20. Curran Associates Inc., 2020. ISBN 9781713829546.