Multi-Armed Bandits for Trustworthy and Resource-Efficient Algorithm Configuration

---

Jasmin Brandt

January 13, 2025

Department of Computer Science,
Warburger Straße 100
33098 Paderborn

Dissertation

In partial fulfillment of the requirements for the academic degree of
Doctor rerum naturalium (Dr. rer. nat.)

Multi-Armed Bandits for Trustworthy and Resource-Efficient Algorithm Configuration

Jasmin Brandt

1st Reviewer	Prof. Dr. Eyke Hüllermeier Institute of Informatics Ludwig Maximilian University of Munich
2nd Reviewer	Prof. Dr. Kevin Tierney Decision and Operation Technologies Group Bielefeld University
Supervisor	Prof. Dr. Eyke Hüllermeier

January 13, 2025

Jasmin Brandt
Multi-Armed Bandits for Trustworthy and Resource-Efficient Algorithm Configuration
Dissertation, January 13, 2025
Reviewers: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Kevin Tierney
Supervisor: Prof. Dr. Eyke Hüllermeier

Paderborn University
Department of Computer Science
Warburger Straße 100
33098 Paderborn

Abstract

The quality and runtime of an algorithm depend significantly on its internal parameters, which usually must be predefined by the user. Since identifying optimal values for these parameters is often daunting or even infeasible, the field of automatic Algorithm Configuration (AC) has emerged. However, most existing methods lack theoretical guarantees regarding the quality of their suggested parameter configurations, making it challenging for users to trust their outcomes. Conversely, the few state-of-the-art methods that provide provable quality guarantees for their configurations often fail to match the resource efficiency of heuristic methods.

One approach to derive theoretical guarantees is to adapt and apply methods from well-studied theoretical settings, such as Multi-Armed Bandits (MABs), to AC problems. Providing such guarantees instills user trust and ensures resource efficiency when the configurator's required budget aligns closely with the lower bound necessary to solve the problem. This ensures trustworthiness without excessive computational costs. However, the multitude of MAB variants — designed for different parallelization methods, feedback types, or winning-arm criteria — makes it challenging to identify the most suitable approach for AC. Additionally, an ideal configurator should handle a wide range of scenarios, such as different types of feedback. While specialized areas of AC, such as Hyperparameter Optimization (HPO), already benefit from MAB-based methods like HYPERBAND, which performs well in practice and comes with theoretical guarantees, no comparable approach exists for the more general AC setting.

A challenge in developing AC methods with theoretical guarantees is that strict adherence to theoretical limits does not always result in resource-efficient methods. In particular, the lack of knowledge about the parameter configuration space necessitates extensive exploration, often leading to conservatively high sampling budgets.

This thesis advances MAB-based AC methods in two directions to improve both resource efficiency and trustworthiness. First, we propose a general MAB framework and apply it to the broader AC setting while maintaining resource efficiency through theoretically derived budget bounds. Second, we adapt the existing HPO algorithm HYPERBAND, including its MAB subroutine SUCCESSIVE HALVING, to enhance resource efficiency while preserving trustworthiness through theoretical constraints.

Specifically, we introduce a general Combinatorial Bandit framework in which a fixed but random-sized set of arms is played in parallel at each time step. This framework accommodates both numerical rewards and preference-based feedback and provably identifies an optimal arm when the available budget of arm pulls is sufficiently large. Furthermore, the necessary budget matches the derived lower bound for such algorithms, up to a logarithmic factor. We also extended and applied this method to the AC setting, transferring the theoretical guarantees. Notably, we proved that our proposed configurator identifies a near-optimal configuration

with high probability when the budget is adequate and empirically demonstrated that it narrows the performance gap between heuristic and theoretically grounded configurators.

Additionally, this thesis presents a modification of the existing MAB method SUCCESSIVE HALVING (SHA), enabling it to reuse observations from previous runs with smaller budgets. As a result, it requires only a minimal number of new samples. We demonstrated that the same theoretical guarantees for the returned arm hold as if the algorithm were run from scratch with entirely new samples. Furthermore, we compared these guarantees to those we derived for asynchronous extensions of SHA, which had not been analyzed until now. Finally, we incorporated our algorithm as a subroutine within the HPO framework HYPERBAND, conducting a theoretical analysis and validating its improved efficiency through experimental studies.

Zusammenfassung

Die Qualität und Laufzeit eines Algorithmus hängen maßgeblich von seinen internen Parametern ab, die in der Regel vom Benutzer vorab festgelegt werden müssen. Da es oft eine herausfordernde oder sogar unlösbare Aufgabe ist, optimale Werte für diese Parameter zu identifizieren, hat sich das Forschungsfeld der automatischen Algorithmenkonfiguration (engl. Algorithm Configuration) entwickelt. Allerdings fehlen den meisten bestehenden Methoden theoretische Garantien bezüglich der Qualität ihrer vorgeschlagenen Parameterkonfigurationen, was es den Nutzern erschwert, den Ergebnissen zu vertrauen. Im Gegensatz dazu bieten die wenigen Methoden, die nachweisbare Qualitätsgarantien für ihre Konfigurationen liefern, oft nicht die Ressourceneffizienz heuristischer Ansätze.

Ein Ansatz zur Herleitung theoretischer Garantien besteht darin, Methoden aus gut erforschten theoretischen Kontexten, wie Mehrarmigen Banditen (engl. Multi-Armed Bandits), auf Algorithmenkonfigurationsprobleme anzuwenden und anzupassen. Solche Garantien stärken das Vertrauen der Nutzer in die Ergebnisse und stellen Ressourceneffizienz sicher, wenn das vom Konfigurator benötigte Budget eng mit der unteren Grenze übereinstimmt, die für die Lösung des Problems erforderlich ist. Dadurch bleibt die Vertrauenswürdigkeit gewahrt, ohne dass übermäßige Rechenkosten entstehen. Allerdings macht die Vielzahl an Varianten von Mehrarmigen Banditen – die für unterschiedliche Parallelisierungsmethoden, Beobachtungen oder Kriterien zur Auswahl des besten Arms konzipiert wurden – es schwierig, die geeignetste Methode für Algorithmenkonfiguration zu identifizieren. Zudem sollte ein idealer Konfigurator in der Lage sein, eine Vielzahl von Szenarien abzudecken, wie beispielsweise verschiedene Arten von Beobachtungen.

Während spezialisierte Bereiche der Algorithmenkonfiguration, wie die Hyperparameteroptimierung (engl. Hyperparameter Optimization), bereits von Mehrarmigen Banditen-basierten Methoden wie HYPERBAND profitieren, die in der Praxis gut funktionieren und mit theoretischen Garantien ausgestattet sind, existiert für die allgemeinere Algorithmenkonfiguration derzeit keine vergleichbare Methode.

Eine Herausforderung bei der Entwicklung von Algorithmenkonfigurationsmethoden mit theoretischen Garantien besteht darin, dass strikte Einhaltung theoretischer Grenzen nicht immer zu ressourceneffizienten Verfahren führt. Insbesondere die fehlende Kenntnis des Parameterkonfigurationsraums erfordert eine umfassende Exploration, was oft zu einer konservativ hohen Stichprobenanzahl führt.

Diese Arbeit entwickelt Mehrarmige Banditen-basierte Algorithmenkonfigurationsmethoden in zwei Richtungen weiter, um sowohl die Ressourceneffizienz als auch die Vertrauenswürdigkeit zu verbessern. Erstens schlagen wir ein allgemeines Mehrarmiges Banditen-Framework vor und wenden es auf das Algorithmenkonfigura-

tionsproblem an, während wir die Ressourceneffizienz durch theoretisch abgeleitete Budgetgrenzen wahren. Zweitens passen wir den bestehenden Hyperparameteroptimierungsalgorithmus HYPERBAND einschließlich seiner Mehrarmigen Banditen Subroutine SUCCESSIVE HALVING an, um die Ressourceneffizienz weiter zu steigern und gleichzeitig die Vertrauenswürdigkeit durch theoretische Einschränkungen sicherzustellen.

Konkret führen wir ein allgemeines kombinatorisches Banditen-Framework ein, bei dem in jedem Zeitschritt eine feste, aber zufällig große Menge von Armen parallel gespielt wird. Dieses Framework unterstützt sowohl numerische Belohnungen als auch präferenzbasierte Beobachtungen und identifiziert nachweislich einen optimalen Arm, wenn das verfügbare Budget für Armziehungen ausreichend groß ist. Darüber hinaus entspricht das erforderliche Budget der abgeleiteten unteren Grenze für solche Algorithmen, bis auf einen logarithmischen Faktor. Wir haben diese Methode zudem auf Algorithmenkonfiguration erweitert und die theoretischen Garantien übertragen. Insbesondere haben wir bewiesen, dass unser vorgeschlagener Konfigurator mit hoher Wahrscheinlichkeit eine nahezu optimale Konfiguration findet, wenn das Budget ausreichend ist, und empirisch gezeigt, dass unsere Methode die Leistungslücke zwischen heuristischen und theoretisch fundierten Konfiguratoren weiter schließt.

Darüber hinaus stellt diese Arbeit eine Modifikation der bestehenden Mehrarmigen Banditenmethode SUCCESSIVE HALVING (SHA) vor, die es ermöglicht, Beobachtungen aus vorherigen Läufen mit kleineren Budgets wiederzuverwenden. Dadurch werden nur minimale neue Stichproben benötigt. Wir haben gezeigt, dass dieselben theoretischen Garantien für den zurückgegebenen Arm gelten, als würde der Algorithmus von Grund auf mit vollständig neuen Stichproben ausgeführt. Außerdem haben wir diese Garantien mit denen verglichen, die wir für asynchrone Erweiterungen von SHA hergeleitet haben, die bisher nicht analysiert wurden. Schließlich haben wir unseren Algorithmus als Subroutine innerhalb des Hyperparameteroptimierungsgerüst HYPERBAND integriert, eine theoretische Analyse durchgeführt und seine verbesserte Effizienz durch experimentelle Studien validiert.

Acknowledgement

In the following, I would like to express my heartfelt thanks to several people who supported me in various ways throughout this work.

First of all, of course, I would like to thank my doctoral supervisor, Prof. Dr. Eyke Hüllermeier, for his reliable supervision, his guidance, insightful ideas and support during difficult phases. My gratitude also extends to the rest of the examination committee, particularly Prof. Dr. Kevin Tierney for his expert advice during our collaborations and for the productive and enjoyable visits to his group at Bielefeld University.

I am deeply thankful for the DataNinja graduate school, which not only funded my position but also connected me with many other DataNinjas. Through the program, I had the privilege of engaging in stimulating discussions, gaining insights into a broad range of topics, and meeting many renowned scientists.

Of course, I do not want to leave my coworkers unmentioned here. Special thanks go to Dr. Viktor Bengs for his support and to Dr. Björn Haddenhorst for his collaboration and mathematical expertise. Although it would be impossible to name everyone individually, I am immensely grateful to have been part of such an incredible team. I look back to interesting technical discussions, but also to funny coffee breaks and head-clearing joint after-work runs or bouldering sessions.

I am also very grateful to Prof. Dr. Andreas Krause and his team for their warm welcome during my time at ETH Zurich. This experience not only provided me with inspiring professional discussions and valuable connections but also exposed me to entirely new perspectives on topics I thought I already understood well.

Finally, I would like to thank my family and friends for always being there for me. Who knows if I would have even discovered my passion for mathematics without the early childhood influence of the geometric patterns on my mother's handbag or the daily questions from my grandparents about multiplication tables? A very special thank you goes to my boyfriend, Toni, (including his enthusiasm for academia) whose companionship during bike rides and runs helped me regain focus and balance throughout this journey.

Contents

1 Introduction

1.1 Motivation

Due to the increasing automation of many processes, a lot of different algorithms for various tasks exist, often with internal parameters that must be defined by the user in advance. Additionally, the growing availability of computational resources enables the development of larger and more complex models often characterized by a greater number of parameters that influence the behavior of the considered target algorithm. These parameters significantly impact the performance of the target algorithm affecting factors such as runtime or the quality of the solution it returns. Therefore, finding an optimal configuration for these internal parameters is crucial to maximize the performance of the target algorithms. However, manually tuning the parameters is a daunting, and sometimes even infeasible, task due to the often infinitely large parameter domains and the interactions between different parameters.

An example of such a target algorithm that should be tuned is GLUCOSE, which solves the Boolean Satisfiability problem, or SAT problem for short. Version 4 of GLUCOSE has a total of 41 parameters, of which 13 are binary and 28 are continuous. For instance, the parameter *Random Frequency* (RF) is a float in the interval $[0, 1]$ that defines the probability of making a random variable assignment instead of using the learned heuristic. In other words, RF$= 1$ corresponds to a random search while RF$= 0$ allows GLUCOSE to assign values solely based on the learned heuristic, thereby impeding exploration. Since there are infinitely many possibilities for the continuous RF parameter, it would be a daunting task to fine-tune it manually.

To address this issue, the field of automated Algorithm Configuration, which aims to automatically identify well-performing parameter configurations for a given target algorithm, has evolved and gained significant interest in recent years. For a more detailed overview see [Sch+22]. However, the subfield of automatic algorithm configurators that offer theoretical guarantees for the returned parameter configuration has received relatively little attention. Moreover, these theoretically grounded approaches often lag behind the performance of many heuristic algorithm configurators. At the same time, theoretical guarantees are crucial for users to trust these methods and to ensure that the configurators perform well not only on a few tested target algorithms and datasets but also in worst-case scenarios. Such assurances can only be provided by a theoretical worst-case analysis. It is important to note that we cannot make theoretical claims about the target algorithms themselves. However, we can prove the ability to find a parameter assignment that (almost) maximizes the

performance of the target algorithm. Thus, we can at least guarantee that the results of the target algorithm are as good as possible with the parameter configuration determined by the theoretically grounded configurator.

Most automatic algorithm configurators have internal parameters that must be specified before execution. For example, one such parameter might be the maximum resource budget, such as the runtime allocated for testing a single configuration during the configurator's execution. To avoid an endless recursion of algorithm configurators for algorithm configurators - resulting in significant runtime and computational resource overhead - it can sometimes be a better strategy to try one specific value for the parameter, and if the results are unsatisfactory, simply rerun the configurator with a different initialization of the parameter. However, a significant amount of previously observed information is lost when the algorithm being rerun cannot handle or reuse data gathered in earlier runs. To improve efficiency, there is a need for algorithmic designs that can incorporate and compare results with previously observed information. While the idea may sound straightforward, maintaining theoretical guarantees for the output of such a strategy is far from trivial. If we consider the special case of an Algorithm Configuration problem with only a finite number of configurations, it can be reduced to a best-arm identification problem in Multi-Armed Bandits [ABM10]. In this context, we are given a finite set of possible options, called arms, from which the learner can choose. After selecting an arm, immediate feedback, such as a numerical reward, is observed. This potentially noisy feedback is typically sampled from an underlying probability distribution associated with each arm and is used to guide the selection of the next arm. The objective is to identify the arm with the highest expected reward as quickly as possible.

If we consider a parameter configuration as an arm and the observed runtime or solution quality with the chosen parameter configuration as feedback, then each Algorithm Configuration problem with a finite configuration space can be regarded as a best arm identification problem in Multi-Armed Bandits. A variety of slight modifications of Multi-Armed Bandits have already been proposed, such as Dueling Bandits or Combinatorial Bandits. In these variants, two arms or a subset with at least two arms are selected simultaneously in each time step, and, for example, only winner information is observed. In the context of Algorithm Configuration, such settings can represent a "race," where different parameter configurations are executed in parallel until the first one finishes. Regardless of the specific Bandit variant, one of the primary challenges in solving best arm identification problems is balancing exploration and exploitation. Exploration involves selecting a wide range of arms to ensure that a good option is not overlooked. Exploitation focuses on repeatedly choosing promising arms to obtain a more reliable estimation of their true quality. A well-designed algorithm for identifying the best arm in a Multi-Armed Bandit setting strikes a good balance between exploration and exploitation. One significant advantage of this simple problem design is that it is often possible to derive theoretical guarantees for methods addressing the best arm identification problem. For instance, guarantees might specify the sufficient number of trials needed to identify the best arm with high probability, which increases user trust in the methods' outputs. Furthermore, theoretical guarantees that the required budget

of an algorithm aligns with or approximates the lower bound of the necessary budget for solving such problems ensure that the algorithms are also resource-efficient.

1.2 Thesis Structure

This thesis begins with two foundational chapters on Algorithm Configuration and Multi-Armed Bandits. The first chapter delves into the existing theoretical works in the field of Algorithm Configuration, with a particular focus on its subproblem, Hyperparameter Optimization. The second chapter describes variants of the Multi-Armed Bandit setting relevant to my work, such as Dueling Bandits, Combinatorial Bandits, and the non-stochastic setting. Following these foundational chapters is a concise overview of the state-of-the-art in my research area and my contributions to these fields. The main parts of my work then follow, consisting of the published papers stemming from my research. The first paper addresses the problem of finding optimal arms in non-stochastic Combinatorial Bandits. The second extends the Combinatorial Bandit approach to handle an infinite number of options, enabling its application to Algorithm Configuration. The third paper enhances an existing method for Hyperparameter Optimization by efficiently reusing information from previous runs while maintaining the same theoretical guarantees as if the method were run from scratch without reusing any information. Finally, I conclude by summarizing my results and providing a brief outlook on future research directions.

2 Algorithm Configuration

Many algorithms have numerous internal parameters that must be set before execution, and these parameters have a significant impact on the algorithm's performance. Therefore, to achieve optimal performance — whether in terms of runtime or solution quality — it is essential to search for the best combination of all internal parameters. Manually tuning these parameters is a daunting or even infeasible task, especially given the increasing complexity and time requirements of modern algorithms. Furthermore, dependencies between certain parameters may exist, meaning that tuning each parameter individually could result in suboptimal outcomes. To address this challenge, the field of automatic Algorithm Configuration (AC) has emerged.

In this chapter, the problems of AC and its subproblem, Hyperparameter Optimization, are formally introduced. Given the theoretical contributions of this thesis, the focus is particularly on the existing theoretical frameworks for these topics.

2.1 Foundations

Motivation. Let us consider a Boolean Satisfiability (SAT) problem, where the goal is to determine an assignment of variables that makes a given Boolean expression evaluate to true. Such an expression can look like the following.

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1.$$

By trial and error, a human can find an assignment like $x_1 =$ FALSE, $x_2 =$ FALSE and $x_3 =$ TRUE of the variables such that the above formula is satisfied. However, with more variables involved and clauses contained in the boolean expression, it becomes increasingly difficult or even unfeasible for humans to find a satisfying assignment for all variables, assuming one exists. In fact, it has been proven that solving such a boolean satisfiability expression is NP-hard. Even the existing algorithms that solve the boolean satisfiability problem like MINISAT [ES04] more or less work by a trial and backtracking approach extended with some sophisticated mechanism for example for the selection of the next variable on which a new assignment is tested. However, MINISAT has some parameters that have a huge influence on the behavior of the solving process and thus also on its required solving time. Examples of these parameters are presented in the following.

- *restart_first* is an integer that defines the interval of a complete restart of all variable assignments to prevent the algorithm from spending excessive time exploring unpromising regions of the search space.

- *restart_inc* gives the real-valued factor of increase of the restart interval.

- *random_var_frequency* defines the probability (expressed as a percentage) with which the decision heuristic used to find the next variable for an assignment chooses a random variable. Otherwise, the heuristic chooses a variable that is contained most frequently in all remaining clauses or was included most frequently in recent conflicts.

- *var_decay* is the real-valued factor in $[0, 1]$ that is multiplied with the increment that is added to the number of recent conflicts that is stored for each variable. If this value is large for a variable, the heuristic is more likely to select it for the next assignment.

It is needless to say that good instantiations of these parameters can highly boost the performance of MINISAT. However, even only for the above-named parameters, there are infinitely many possible configurations of parameters, which makes the task of finding a well-performing configuration per hand daunting or even infeasible. This challenge has led to the emergence of automated Algorithm Configuration (AC), which has garnered significant interest in recent years. The following sections provide a detailed explanation of Algorithm Configuration and its applications.

Problem formulation. Let $\mathcal{A}$ be a parameterized target algorithm and $\mathcal{I}$ the space of problem instances that $\mathcal{A}$ can solve. Let us assume the existence of an unknown probability distribution $\mathcal{P}$ defined over the instance space $\mathcal{I}$. We denote by $\Theta = \Theta_1 \times \cdots \times \Theta_n$ the *configuration space* that consists of all feasible parameter configurations for the target algorithm $\mathcal{A}$ that has n internal parameters from the domains $\Theta_1, \ldots, \Theta_n$. These parameters may be real-valued, integer-valued, binary, or categorical, depending on the design of the target algorithm $\mathcal{A}$. To guide the selection of the parameter configurations, we need a way to measure the quality of a specific configuration $\theta \in \Theta$, which is usually done with a *cost function* $c : \mathcal{I} \times \Theta \to \mathbb{R}$ that represents the potentially noisy cost $c(i, \theta)$ of running the target algorithm $\mathcal{A}$ with the parameter configuration $\theta \in \Theta$ on problem instance $i \in \mathcal{I}$. Common examples of this cost are the used runtime of $\mathcal{A}$ to finish or a numerical value representing the quality of the returned solution of $\mathcal{A}$.

The **goal** is to find a parameter configuration $\theta^* \in \Theta$ that performs best on average over the probability distribution over the instance space $\mathcal{P}(\mathcal{I})$:

$$\theta^* \in \text{argmin}_{\theta \in \Theta} \int_{\mathcal{I}} c(i, \theta) d\mathcal{P}(i).$$

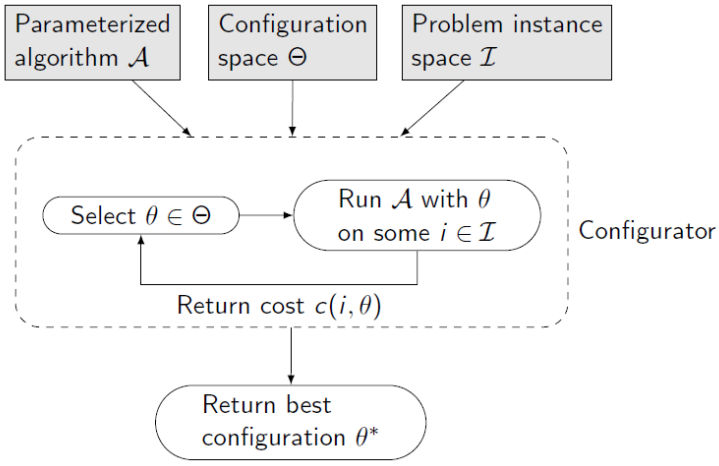


Fig. 2.1.: Demonstration of the task of an algorithm configurator.

Considering the integral over the probability distribution $\mathcal{P}(\mathcal{I})$ emphasizes the importance of problem instances $i \in \mathcal{I}$ that are more likely to be sampled. In other words, it is more important for the best parameter configuration θ^* to perform well on problem instances that are sampled more often than on problem instances that are rarely sampled.

Figure 2.1 provides a graphical representation of the AC problem. The grey boxes indicate the given inputs, while the dashed box represents the automatic solver, also referred to as the *configurator*. Usually, it works by iteratively selecting one or more parameter configuration(s) from the space Θ, and subsequently runs the target algorithm $\mathcal{A}$ with the selected parameter configuration(s) on sampled problem instance(s) from $\mathcal{I}$ and using the observed cost for this run to guide the selection of the next parameter configuration(s) in the following iterations. The objective of identifying the optimal parameter configuration is highlighted at the bottom of the figure.

However, in practice, the probability distribution $\mathcal{P}$ over the instance space $\mathcal{I}$ is usually unknown. Furthermore, $\mathcal{I}$ is often an infinitely large space, making it impossible to evaluate the performance of a parameter configuration on every problem instance. To address this issue, we can solve a proxy problem. Here, a finite training set $\mathcal{I}_{\text{train}} \subseteq \mathcal{I}$ is provided, and the goal is to identify the configuration $\hat{\theta} \in \Theta$ that performs best on average over this training set:

$$\hat{\theta} \in \text{argmin}_{\theta \in \Theta} \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{i \in \mathcal{I}_{\text{train}}} c(i, \theta).$$

It is important to note that, if the training set is sufficiently large and randomly sampled from the probability distribution $\mathcal{P}$, the performance of $\hat{\theta}$ converges to that of θ^* due to the law of large numbers.

If the configuration space Θ is also infinitely large, such as in the case of continuous parameter spaces, finding the optimal parameter configuration becomes analogous to searching for a needle in a haystack. Hence, it is reasonable to relax the objective to finding a near-optimal configuration instead. For instance, an *ϵ-optimal configuration* $\tilde{\theta} \in \Theta$ can be defined as one whose expected cost exceeds the optimal expected cost by at most a factor of $\epsilon > 0$

$$\mathbb{E}_{i\sim\mathcal{P}}[c(i,\tilde{\theta})] \leq (1+\epsilon)\text{OPT},$$

where $\text{OPT} = \inf_{\theta\in\Theta} \mathbb{E}_{i\sim\mathcal{P}}[c(i,\theta)]$ is the minimal expected cost in the whole configuration space on average over the probability distribution of the instance space.

But even such a near-optimal configuration can be hard to find if the runtime distributions of the configurations are heavy-tailed. In this case, running a configuration might take an impractically long time or even fail to terminate, preventing the automatic algorithm configuration from converging or returning a solution. To address this issue, a *capping timeout* $\kappa > 0$ can be introduced, representing the maximum time allowed for running a configuration before it is forcibly terminated if no solution is found. The κ-capped observation for a configuration $\theta \in \Theta$ on an instance $i \in \mathcal{I}$ is defined as $\min(c(i,\theta),\kappa)$, where the observed cost is either the actual cost $c(i,\theta)$ or the timeout κ, whichever is smaller. To choose an appropriate value for κ is far from trivial and multiple solutions for the search for κ were proposed in the literature. This includes some expensive approaches in time and resources like making an optimistic guess for κ and then doubling it until a specific fraction of the instances finishes within the timeout as described in [WGS19].

Given these capped observations, it becomes necessary to redefine the notion of optimal configurations to account for the introduced timeout. Otherwise, an algorithm configurator could simply choose κ as small as possible such that no parameter configuration finishes within the timeout and can return any of them as an "optimal" one according to our previous definition of (near-) optimal configurations. Thus, it is essential to ensure that the returned configuration completes successfully on at least a $(1-\delta)$-fraction of the evaluated instances, where $\delta > 0$ represents the allowed fraction of instances where the configuration may fail. In other words, we exclude a δ-fraction of instances on which the target algorithm fails to finish within the timeout κ with the regarded configuration. We call $\tilde{\theta} \in \Theta$ an (ϵ,δ)-*optimal configuration* if and only if

$$\exists\kappa > 0 \;:\; \mathbb{E}_{i\sim\mathcal{P}}[\min(c(i,\tilde{\theta}),\kappa)] \leq (1+\epsilon)\text{OPT} \quad\wedge\quad \mathbb{P}_{i\sim\mathcal{P}}(c(i,\tilde{\theta}) > \kappa) \leq \delta.$$

But still, we have the problem that we need to consider all of the configurations to decide on the (ϵ,δ)-optimality because otherwise, it is not possible to get a reliable estimation of the optimal cost OPT. To avoid this, we can relax the notion of optimal cost to the optimal cost after excluding the γ-fraction of best configurations for a $\gamma \in (0,1)$:

$$\text{OPT}_\gamma = \inf_{x\in\mathbb{R}}\{x \mid \mathbb{P}_{\theta\sim\text{Unif}(\Theta)}(\mathbb{E}_{i\sim\mathcal{P}}[c(i,\theta)] \leq x) \geq \gamma\}.$$

According to this, we can now define an $(\epsilon, \delta, \gamma)$-*optimal configuration* $\tilde{\theta} \in \Theta$ as one that has an expected capped cost at most a factor of $1 + \epsilon$ worse than the optimal cost excluding the best γ-fraction and which is able to return a solution within a capping time κ on a $(1 - \delta)$-fraction of instances

$$\exists \kappa > 0 \ : \ \mathbb{E}_{i \sim \mathcal{P}}[\min(c(i, \tilde{\theta}), \kappa)] \leq (1 + \epsilon)\text{OPT}_\gamma \quad \wedge \quad \mathbb{P}_{i \sim \mathcal{P}}(c(i, \tilde{\theta}) > \kappa) \leq \delta.$$

A variant of the AC problem is the *per-instance* AC problem, which considers an online setting. The main difference is that now the instances $i_t \in \mathcal{I}$ arrive sequentially in each time step $t \in \mathbb{N}$ and for each seen instance, we want to find the individual best parameter configuration θ_t^* that leads to the best performance only on instance i_t:

$$\theta_t^* \in \text{argmin}_{\theta \in \Theta} c(i_t, \theta) \quad \forall t \in \mathbb{N}.$$

Note that the optimal per-instance configuration for a specific problem instance may not lead to small costs on average on $\mathcal{P}(\mathcal{I})$. This setting becomes particularly relevant when the initial number of problem instances is limited, sampling new instances is costly, or the environment changes over time, for example, due to shifts in the probability distribution $\mathcal{P}$ over the instance space $\mathcal{I}$. The relaxations introduced for near-optimal configurations in general AC can be straightforwardly adapted to define near-optimal parameter configurations in the per-instance AC setting.

Example. Even if the formal problem definition is restricted to target algorithms, the problem can be generalized to find good configurations for a completely unknown black-box system. For example, consider the task of optimizing a race car. Here, the race car represents the black-box system to be optimized, while its components, such as the engine, tires, spoiler, etc., are the internal parameters for which we aim to find the optimal combination. The goal is to identify the components such that the car races as fast as possible or in other words the cost function is given by the race time of the car in this example.

In this scenario, the problem instances correspond to different race tracks, each with unique characteristics such as varying surfaces and weather conditions. If changing the car's components incurs high costs, it is reasonable to focus on identifying a configuration that performs well on average across all race tracks, which aligns with the general AC problem.

In the per-instance AC scenario, however, we assume the ability to observe specific track and weather conditions before selecting the car's components. This allows us to determine the optimal configuration tailored to the particular environment of each race course.

Existing Methods. Algorithm configurators that do not assume an underlying model to map configurations to specific performance outcomes are referred to as model-free. A prominent model-free method that solves the algorithm configuration (AC) problem using simple local neighborhood search is ParamILS [Hut+09]. To avoid getting trapped in local optima, the search frequently restarts at random, and configurations are iteratively compared to determine which parameter should be perturbed next. However, a significant drawback of this method is its inability to handle continuous parameter spaces.

Some configurators address this limitation by supporting continuous parameter spaces through simple racing strategies. In these methods, all remaining configurations are evaluated on a single problem instance, and the worst-performing configurations are subsequently eliminated. One of the most sophisticated algorithms following this strategy is irace [Lóp+11]. Beyond eliminating poor configurations, irace generates new configurations from a bivariate normal distribution over the parameter space, centered around the previous winning configuration. To prevent stagnation in local optima, the algorithm performs a soft restart if the current population of configurations becomes too similar.

Another well-known model-free method is GGA [AST09]. This algorithm divides configurations into competitive and non-competitive subsets. In each iteration, the competitive configurations are raced against one another. Subsequently, one-third of the configurations are modified using genetic operations such as recombination and mutation, involving a winning configuration and a configuration from the non-competitive subset. The process terminates once the new competitive population ceases to outperform the previous one.

In contrast, model-based AC methods employ predictive models, such as random forests or Bayesian networks, to identify promising configurations. For example, GGA++ [Ans+15] extends GGA by incorporating a surrogate model to predict the performance of newly generated configurations. After creating new configurations using genetic algorithms like crossover, the surrogate model predicts the rank of each configuration in a simulated tournament, and only the top-performing configurations are added to the competitive population. In this approach, a random forest is used as the surrogate model, with increased resolution in areas of the parameter space containing high-performing configurations. This enhancement significantly improved performance compared to GGA for tuning SAT solvers.

Another classical example of a model-based method is SMAC [HHL11]. This algorithm also builds a random forest surrogate model over the combined problem instance and parameter configuration space. Configurations are evaluated on various combinations of seeds and problem instances sampled at random. The model then predicts the performance of the target algorithm $\mathcal{A}$ with a specific configuration on a given problem instance, guiding the search for optimal configurations.

2.2 Theory

What all the above-presented methods have in common is that none of them provides a theoretical guarantee regarding the performance of the returned configuration. This lack of guarantees is primarily due to the absence of information about the structure of the configuration and instance spaces. Consequently, theoretical assumptions about these spaces, such as continuity, are difficult to evaluate and, in practice, often invalid. Additionally, conducting a theoretical worst-case analysis, which tends to be highly conservative, requires extensive exploration of the infinitely large configuration space. This leads to a waste of resources on non-promising parameter configurations. Due to these challenges, most existing configurators rely on heuristic methods and lack any rigorous theoretical analysis.

Nevertheless, there are exceptions in the field of AC that provide theoretical guarantees. One notable example is the derivation of a bound for the *estimation error*:

$$\left| \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{i \in \mathcal{I}_{\text{train}}} c(i, \theta) - \int_{\mathcal{I}} c(i, \theta) d\mathcal{P}(i) \right|. \tag{2.1}$$

As clearly visible, this bound is highly dependent on the corresponding cost function c and on the distribution $\mathcal{P}$ over the instance space $\mathcal{I}$. If any knowledge about the distribution $\mathcal{P}$ is available, we call this estimation error *data-dependent*, otherwise, it is called *distribution-free* and just a worst-case bound over all possible distributions. By fixing a desired estimation error $err > 0$, we can set a derived upper bound that will still depend on $|\mathcal{I}_{\text{train}}|$ for the term in 2.1 equal to err, solve the equation for $|\mathcal{I}_{\text{train}}|$ and get the necessary number of training samples we need to achieve the given error err.

For instance, [KLL17] demonstrated that for specific choices of γ and δ, the worst-case expected runtime of any algorithm configurator to return an $(\epsilon, \delta, \gamma)$-optimal configuration is in $\Omega\left(\frac{1}{\epsilon^2} C(\delta, \gamma) \text{OPT}_\gamma\right)$, where $C(\delta, \gamma)$ is a parameter determined by the choices of δ and γ. The goal of any algorithm configurator that is guaranteed to find such a near-optimal configuration is to match this lower bound as well as possible.

Existing Theoretical Methods. Even though most algorithm configurators are heuristic approaches, a few methods with theoretical guarantees exist. Below, some of the basic approaches are presented.

As the name suggests, the main idea in the STRUCTURED PROCRASTINATION algorithm in [KLL17] is to procrastinate the runs for hard instances. This is achieved by maintaining a queue of $(instance, timeout)$ pairs for each configuration. Iteratively, the configuration with the lowest runtime estimation so far is selected, along with the first item from its queue. If the instance cannot be solved within the

timeout, the timeout is added to the configuration's runtime estimation, and the pair $(instance, 2 \cdot timeout)$ is placed back into the queue. If the instance is solved within the timeout, the observed runtime is added to the runtime estimation of the configuration, and a new unseen instance with the initial timeout is added to the queue. The authors proved that the algorithm identifies an (ϵ, δ)-optimal configuration with high probability within a runtime that is optimal up to a logarithmic factor. Specifically, they showed that the sufficient budget for their algorithm is in $O(\ln(\frac{n}{\delta\epsilon^2})\frac{n}{\delta\epsilon^2}\text{OPT})$, while the lower bound for the necessary budget to solve this problem is in $\Omega(\frac{n}{\delta\epsilon^2}\text{OPT})$. Thus, their proposed algorithm is optimal up to the logarithmic factor. More recent AC methods including an extension of the above-described STRUCTURED PROCRASTINATION algorithm are explained in section 4.2.

The doubling trick is also applied in LEAPSANDBOUNDS [WGS18]. In this method, an optimistic initial timeout is chosen, and all configurations are run on all instances in the training set. If none of the configurations finishes before the timeout, the timeout is doubled, and the procedure is repeated. If at least one configuration finishes within the timeout, the fastest configuration is returned. The authors provided a worst-case upper bound on the total runtime, which is in $O(\ln(\frac{n \ln \text{OPT}}{\zeta})\frac{n}{\delta\epsilon^2}\text{OPT})$, to find an (ϵ, δ)-optimal configuration with probability at least $1 - \zeta$.

Utility-based algorithm configuration was proposed in [GLR23], where the existence of a decreasing runtime utility function is assumed. This utility represents the expected well-being of a configuration and the goal is to identify the configuration that maximizes the expected utility over time. The authors presented a naive approach that can find an ϵ-optimal configuration with probability at least $1 - \delta$ for a specified accuracy parameter ϵ and a capping time κ. As a more sophisticated approach, they proposed an anytime procedure, called UTILITARIAN PROCRASTINATION. This method gradually increases the capping time and shrinks the accuracy parameter ϵ during execution. The authors proved that UTILITARIAN PROCRASTINATION identifies an ϵ-optimal configuration with probability at least $1 - \delta$ if the time horizon is sufficiently large.

Despite these initial advances in theoretically grounded algorithm configurators, a significant performance gap remains in practice compared to heuristic methods. Since there is no theoretical knowledge about the configuration space, sufficient exploration of Θ must be ensured. This lack of information prevents guarantees about the quality of local neighborhood searches or genetic approaches and leads to conservative estimates of the required number of samples to identify a near-optimal configuration.

2.3 Hyperparameter Optimization (HPO)

Motivation. Not only classical algorithms but also machine learning algorithms heavily depend on the quality of their preset parameters. In this specific context,

machine learning algorithms typically have numerous internal parameters that are learned by the algorithm itself and cannot be initialized by the user. To avoid confusion, we will refer to these parameters learned by the algorithm simply as parameters. Meanwhile, the parameters that must be defined by the user before running the training process of the algorithm will be called *hyperparameters*.

The automatic search for an optimal hyperparameter configuration of a machine learning algorithm for a given dataset of training instances is known as *Hyperparameter Optimization* (HPO). This process is described in more detail in the following section.

Problem formulation. Since HPO is a subproblem of AC, we again assume that we are given a target machine learning algorithm $\mathcal{A}$ with n different hyperparameters, each defined over its domains $\Theta_1, \ldots, \Theta_n$. The overall hyperparameter configuration space is denoted by $\Theta = \Theta_1 \times \cdots \times \Theta_n$. Instead of multiple problem instances, we are provided with a training dataset $\mathcal{I}_{\text{train}}$, which can be viewed as a single instance in the context of AC. For this dataset, the goal is to find the optimal *hyperparameter configuration* (HPC) $\theta^* \in \Theta$. While in AC most algorithms aim to find a configuration that minimizes the runtime of the target algorithm, in HPO, the cost function is typically the validation error of $\mathcal{A}$. Nonetheless, the goal in HPO remains the same: to find the HPC that leads to the smallest cost - in this case to the validation error - on the given training dataset $\mathcal{I}_{\text{train}}$, thereby optimizing the performance of the learning algorithm $\mathcal{A}$.

An illustration of the HPO problem and the task of a hyperparameter configurator is shown in Figure 2.2. The components provided are indicated in the grey boxes, while the dashed box represents the hyperparameter configurator, also referred to as the *hyperparameter optimizer*. Similar to AC, the goal is to identify the HPC that results in the best performance of the target algorithm according to a loss function $l : \Theta \to \mathbb{R}$.

Example. Classical examples of important use cases for HPO include deep neural networks. These networks often have numerous hyperparameters from various domains, such as a real-valued learning rate, an integer-valued number of hidden layers, a binary variable indicating whether to use early stopping, and a categorical choice of optimizer. Searching manually for the optimal combination of these hyperparameters is highly time-consuming, as training a deep neural network typically requires significant computational resources and time.

Using an HPO method not only reduces the manual effort required by researchers and practitioners but also improves the performance of the neural network in terms of validation error. Moreover, it enhances the reproducibility of scientific work and promotes fairness when comparing different neural network architectures.

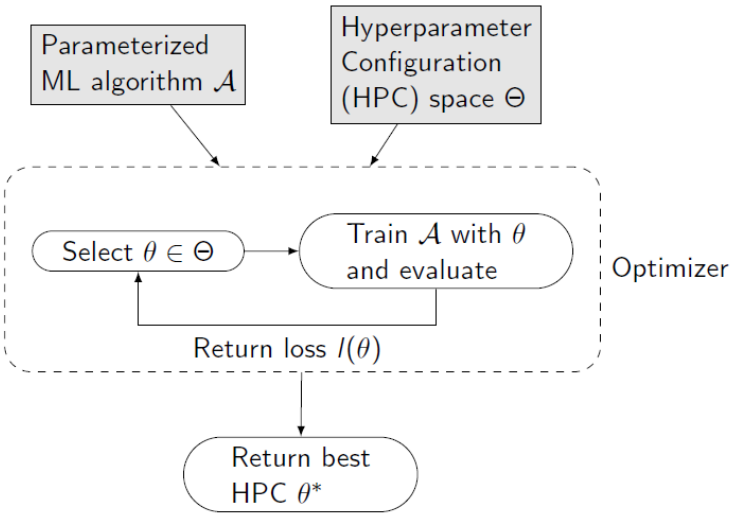


Fig. 2.2.: Demonstration of the task of an hyperparameter optimizer.

Existing Methods. A simple yet effective method, particularly for discrete parameter domains with a small number of hyperparameters is GRID SEARCH, as used in [Lar+07]. The core idea is to create a grid in the space of HPCs by selecting a fixed, discrete set of values for each hyperparameter and considering all possible combinations. However, since the number of HPCs that must be tested grows exponentially with the number of hyperparameters, this approach can become inefficient when the dimenionality is high.

An alternative approach is RANDOM SEARCH [BB12], where a finite number of HPCs is sampled randomly. Compared to grid search, where the same values for individual hyperparameters are tested repeatedly, random sampling better covers the range of each hyperparameter. However, this method still has limitations: many resources may be spent evaluating poorly performing configurations, and promising regions in the HPC space may not be explored thoroughly.

To focus the search on promising areas, BAYESIAN OPTIMIZATION [Wu+19] can be employed. This method iteratively selects HPCs to evaluate based on a probabilistic surrogate model of the cost function, such as validation error. After each evaluation, the surrogate model is updated to better guide the search toward well-performing configurations.

An HPO method with theoretical guarantees is HYPERBAND [Li+17]. It builds on the SUCCESSIVE HALVING algorithm, which is used for best arm identification in Multi-Armed Bandit problems. HYPERBAND iteratively calls SUCCESSIVE HALVING with varying sets of HPCs and budgets per configuration. This approach addresses the trade-off between the number of HPCs to evaluate and the computational budget allocated to each configuration. If many HPCs are evaluated, the configuration space is well explored, but the performance estimates for individual configurations may be unreliable due to limited budget allocation. Conversely, allocating more budget to

fewer configurations reduces exploration, but leads to more reliable performance estimated of each considered configuration. HYPERBAND balances this trade-off effectively and is proven to yield a near-optimal HPC with high probability after a given computational budget. A more detailed overview of the functionality of HYPERBAND and its subroutine SUCCESSIVE HALVING is given in Section 4.3.1.

3 Multi-Armed Bandits

As mentioned in Section 2.3, some Algorithm Configuration (AC) and Hyperparameter Optimization (HPO) methods are inspired by approaches from the Multi-Armed Bandit (MAB) community. In the following chapter, the settings of stochastic Multi-Armed Bandits, Dueling Bandits, Combinatorial Bandits, and non-stochastic feedback are formally introduced. Alongside a description of these settings, examples of problems addressed within each framework are provided, and the most important methods for solving these problems are discussed. Furthermore, the connection between Multi-Armed Bandit approaches and Algorithm Configuration is elaborated upon, highlighting the advantages of using MAB methods in the context of AC.

3.1 Stochastic setting

Motivation. In sequential decision-making, it is crucial to strike a balance between gaining new knowledge (referred to as "exploration") and optimizing outcomes by leveraging existing knowledge to select the best options (referred to as "exploitation"). For instance, in clinical trials, on the one hand, you want to evaluate the effectiveness of various medical treatments (as long as sufficient prior investigation ensures that it is ethically acceptable to administer them to patients). On the other hand, you aim to minimize patient discomfort by providing treatments already known to be effective. Since the impact of medical treatments can vary from person to person, multiple trials are necessary to develop a reliable understanding of their effects.

The Multi-Armed Bandit framework addresses this challenge by focusing on finding an optimal tradeoff between exploration and exploitation in environments with noisy feedback. The goal is to identify the best option as quickly as possible or to select it as frequently as possible within a given time horizon.

Problem formulation. In the *stochastic Multi-Armed Bandit* setting, there are n different options, also referred to as *arms*, which are denoted by their indices $\{1, \dots, n\} =: [n]$. The action set $\mathfrak{A}$, representing the set of choices available to the learner at each time step, coincides with the set of arms, i.e. $\mathfrak{A} = [n]$. At each time step $t \in \{1, \dots, T\}$ within a potentially infinite time horizon $T \in \mathbb{N} \cup \{\infty\}$ the learner must select an action $i_t \in \mathfrak{A}$. Upon making this choice, the learner directly observes feedback, typically in the form of a numerical payoff $r_{i_t,t} \in \mathbb{R}$, also referred

to as *reward*. In the *stochastic* setting, the reward $r_{i_t,t} \sim \mathcal{D}_{i_t}$ is usually drawn from an underlying but unknown probability distribution $\mathcal{D}_{i_t}$ specific to each arm, each distribution having a mean value of μ_{i_t}.

The setting was first introduced by [Tho33] and [Rob52] and the name 'Multi-Armed Bandit' originates from the prominent example of multiple slot machines, also known as one-armed bandits, arranged in a row. At each time step, the user selects one of the gamblers or in other words *pulls* an *arm* and immediately receives a payoff. In this context, the arms represent the available choices in the general problem formulation. Returning to the earlier example of clinical trials, an arm would correspond to a specific medical treatment, and the observed reward would be the improvement in a patient's health.

We can distinguish between two different problem scenarios in MABs. In the *best arm identification* problem, the learner's **goal** is to identify an arm $i^* \in \mathfrak{A}$ that yields the highest average reward as quickly or as reliably as possible. Formally, we are looking for

$$i^* \in \text{argmax}_{i \in \mathfrak{A}} \mu_i$$

and define the maximal mean value as $\mu^* = \max_{i \in \mathfrak{A}} \mu_i$. Within the best arm identification problem, we can further distinguish between two different directions. The first is the *fixed confidence* setting in which we are given a maximum allowable failure probability $\delta \in [0, 1]$ and the objective is to identify the best arm with a confidence level of at least $1 - \delta$ as quickly as possible. The second problem variant considers a *fixed budget* of allowed arm pulls. The objective in this setting is to maximize the probability that the returned best arm is indeed the true best arm, without exceeding the given budget. Note that to solve the best arm identification problem, thorough exploration is essential, while exploitation becomes unnecessary once we are sufficiently confident about the quality of an arm. For this reason, this problem scenario is also referred to as the *pure exploration* problem in some literature, such as in [BMS09].

The second potential problem scenario is the *regret minimization* problem. Here, the goal is to select and pull the arms in such an order and with such a frequency, that the expected regret incurred from these choices is minimized. To quantify this, we define the *cumulative regret* as a measure of the cumulative suboptimality of the chosen arms up to a given time $T \in \mathbb{N}$, as follows.

$$R_T = \max_{i \in \mathfrak{A}} \sum_{t=1}^{T} r_{i,t} - \sum_{t=1}^{T} r_{i_t,t}.$$

Choosing the best arm at every time step minimizes the cumulative regret. However, since both the rewards and the arm choices at each time step are stochastic in the considered setting, it is more meaningful to evaluate the cumulative regret in

expectation. This measure is referred to as the *expected regret* and is calculated as follows

$$\mathbb{E}[R_T] = T \cdot \mu^* - \sum_{t=1}^{T} \mu_{i_t}.$$

However, the learner cannot directly observe the expected regret and must estimate the mean values of each arm's reward distribution. While the time horizon T can, in principle, be infinite, exploiting high-performing arms is crucial to keep the expected regret as small as possible. However, exploration is equally important, as the learner needs to accurately estimate the mean rewards to reliably identify good arms.

In principle, a regret minimization algorithm can be adapted to solve the best arm identification problem by returning the arm with the highest estimated mean reward after meeting a predefined stopping criterion, such as a specific number of time steps. However, because regret minimization algorithms are designed to minimize expected regret as their primary objective, they are typically suboptimal for best arm identification. In other words, if a fixed budget of arm pulls is available, regret minimization algorithms are unlikely to minimize the failure probability effectively. Similarly, if a desired failure probability is specified, these methods are unlikely to identify the best arm as quickly as algorithms specifically designed for best arm identification. This discrepancy arises from the differing objectives of the two types of algorithms.

Example. A classic example is choosing a restaurant for dinner, as illustrated in Figure 3.1. Each day, you can select one of the nearby restaurants. You face the dilemma of deciding whether to revisit your favorite restaurant—where you've dined many times and know the food is delicious—or to try a restaurant you've visited less frequently. Always choosing your favorite restaurant might cause you to miss out on discovering an even better one. For instance, during your last visit to another restaurant, the chef may have had an off day, even though they usually prepare food that surpasses your favorite spot. On the other hand, trying a new restaurant involves the risk of receiving food that is not as good as what your favorite restaurant offers.

Theoretical Guarantees. One of the great advantages of MABs is the possibility to derive some theoretical guarantees. For the best arm identification problem with fixed confidence, there are for example some bounds on the necessary budget required to identify the best arm with the desired confidence. This is usually done using the law of large numbers which guarantees that the estimated mean rewards will converge to their ground truth and some concentration inequalities like Hoeffding's or Bernstein's inequality. Thus, if the time horizon is large enough, we can distinguish which arm has a higher mean reward with high probability. In

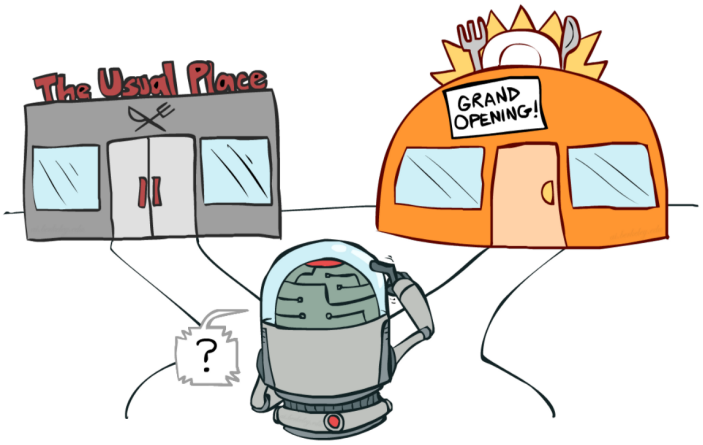


Fig. 3.1.: The difficult choice between your favorite restaurant and a new opened one from [@].

contrast to this, for the fixed budget setting the failure probability of selecting an incorrect arm can be bounded.

In the scenario of regret minimization, the goal is to bound the expected regret that is suffered until a specific time step $T \in \mathbb{N}$ as the name of the problem already suggests. Note that if you simply randomly select an arm in each time step, you will suffer a cumulative regret that grows linearly over time. Thus the goal for the theoretical analysis of an algorithm is to derive an upper bound on the suffered expected regret that is sub-linear. On the other hand, it is proven e.g. in [Bub12] that any algorithm suffers at least a logarithmic regret for Bernoulli distributed rewards. The problem-independent lower regret bound for any distribution is of order $\sqrt{nK}$, so the expected regret cannot get smaller than this to guarantee enough exploration. The approach to derive this regret bound is to estimate an upper bound on the number of times suboptimal arms are pulled.

Existing Methods. A method tackling the best arm identification problem with fixed confidence is called TRACK-AND-STOP and was proposed in [GK16]. The main idea is to ensure that the number of pulls for each arm remains proportional to the theoretically optimal allocation and to stop when a statistical test confirms that one arm has a larger estimated mean reward than all others, with an error probability of at most δ. The sample complexity of their algorithm to find an optimal arm with probability at least $1 - \delta$ asymptotically matches the lower bound they derived for the sample complexity of this problem.

An example of a method solving the fixed budget best arm identification problem is the SUCCESSIVE HALVING algorithm in [JT16]. The strategy involves dividing the available budget uniformly across all iterations. After each iteration, the worst-performing half of the arms is discarded, such that we have double of the budget available per arm in the next iteration. The authors derived a sufficient budget to

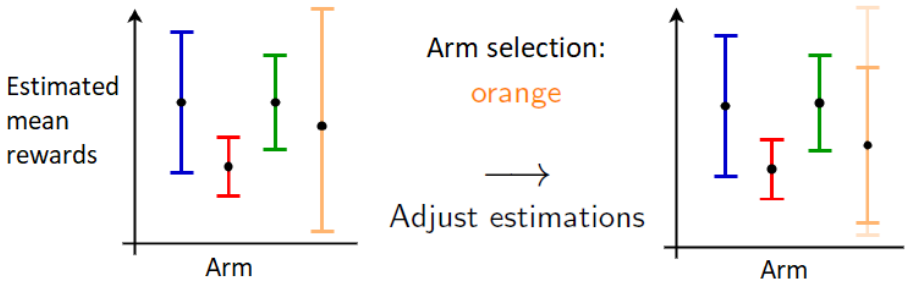


Fig. 3.2.: Example of one time step with the Upper Confidence Bound approach.

guarantee that SUCCESSIVE HALVING returns the best arm, which is dependent on the number of rounds that are needed to discard all except one arm $\log_2(n)$, on the number of overall arms n, the speed the rewards converge to their mean values and the difficulty of the problem, characterized by the suboptimality gap $\mu^* - \mu_i$ for each $i \in [n] \backslash \{i^*\}$.

Several methods address the regret minimization scenario. A famous example is an UPPER CONFIDENCE BOUND (UCB) approach (see e.g. [Bub12]), where the strategy is to construct intervals around each estimated mean reward in which the real mean reward lies with high confidence. These confidence bounds shrink as the arm is pulled more frequently, increasing certainty about the estimated mean. In each time step the arm is selected that has the highest upper confidence bound which can be interpreted as an optimistic estimate of the expected reward for each arm.

An example is shown in Figure 3.2, where the confidence intervals of the arms are represented by the colored bars around the estimated mean rewards that are shown by the black dots for each of the considered four arms. The arm with the highest upper confidence interval bound is the orange arm, which is then selected by the learner. After that, the confidence bounds are adjusted according to the observed reward and get smaller since we have a higher confidence about the estimated mean reward now. In the next time step, the learner would pull the blue arm which is the one with the highest upper confidence bound value now.
For Bernoulli distributed rewards, suboptimality gaps $\Delta_i := \mu^* - \mu_i$ and an exploration factor $\alpha > 1$, the expected regret of UCB can be bounded by $\mathbb{E}[R_T] \leq \sum_{i \in [n] \backslash \{i^*\}} \left(\frac{\alpha+1}{\alpha-1} \Delta_i + \frac{2\alpha \log T}{\Delta_i} \right)$. The lower bound for Bernoulli distributed rewards has the form $\liminf_{n \to \infty} \mathbb{E}[R_T] \geq \sum_{i \in [n] \backslash \{i^*\}} \frac{\log T \Delta_i}{\mathrm{kl}(\mu_i, \mu^*)}$ as shown in [Bub12], where $kl(x, y)$ denotes the Kullback-Leibler divergence, which measures the dissimilarity between two probability distributions.

An alternative approach to addressing the regret minimization problem is the *probability matching* strategy. Here, the goal is to pull each arm exactly with the same probability that matches the Bayesian posterior probability that the arm is the best option. The most famous algorithm that follows this approach is the THOMPSON

SAMPLING algorithm [KKM12]. It works by maintaining a probability distribution for each arm's reward (e.g., Gaussian for unbounded rewards or Beta for bounded rewards) that approximates the posterior distribution given the observed data. In each time step, a reward sample is drawn from the posterior distribution for each arm, and the arm with the highest sampled reward is selected. In a theoretical analysis, the authors have shown that for every $\epsilon > 0$, there exists a problem-dependent constant $C(\epsilon, \mu_1, \ldots, \mu_n)$, which accounts for the variability in problem complexity, such that the expected regret suffered by THOMPSON SAMPLING can be bounded by $\mathbb{E}[R_T] \leq (1+\epsilon) \sum_{i \in [n] \setminus \{i^*\}} \frac{\Delta_i (\log T + \log\log T)}{\mathrm{kl}(\mu_i, \mu^*)} + C(\epsilon, \mu_1, \ldots, \mu_n)$.

Connection to Algorithm Configuration In principle, by interpreting the parameter configurations as arms and the negative of the cost function as rewards, we can interpret each AC problem with a finite configuration space Θ as a MAB problem. While the classical AC problem in which we want to find one configuration that performs well on average coincides with the best arm identification problem in MABs, the per-instance AC problem can be seen as a regret minimization problem, where the regret is defined as the difference between the observed cost for the chosen configuration and the cost for the optimal configuration on the corresponding problem instance.

The main difference in practice is that MABs usually deal with a finite number of arms while the parameter configuration spaces in AC are often very large and may even be infinite. However, with an appropriate discretization of the configuration space, it is also possible to transfer AC problems with an infinite configuration space Θ to a MAB problem. The simplest possibility would be to sample a finite number of configurations uniformly at random from Θ and to assume simultaneously that the proportion of (near-)optimal configurations in the space is large enough to have at least one of the (near-)optimal configurations contained in the sample set. Note that one sampled configuration belongs to the γ proportion of best configurations with probability γ, thus if we sample n many configurations $\theta_1, \ldots, \theta_n \in \Theta$ uniformly at random, the probability that at least one of the n sampled configurations belongs to the top γ proportion of configurations is $1 - (1-\gamma)^n$. By defining an acceptable failure probability $\zeta \in (0, 1)$ for the algorithm configurator and setting this $\geq$ to the probability $(1-\gamma)^n$ that we have no configuration sampled from the best γ proportion, we can solve the equation for the number of samples n and can derive that we need at least $n \geq \lceil \log(\zeta) / \log(1-\gamma) \rceil$ samples to ensure that at least one configuration from the top γ proportion is included in the sample set with probability at least $1 - \zeta$.

The huge advantage of using MAB methods for AC problems is that these usually come with theoretical guarantees. By simple adjustments like discretizing the parameter space and sampling the problem instances uniformly at random from the instance space, it is possible to transform these guarantees to the AC setting which leads to more trust in the returned configurations. Moreover, resource efficiency can be guaranteed if it can be shown that the sufficient budget for the algorithm

to return a near-optimal configuration with high probability (almost) matches the theoretically derived necessary budget.

3.2 Dueling Bandits

Motivation. In some cases, it could be hard or even impossible to observe numerical feedback for selecting an arm and it might be easier to observe relative preferences, i.e., which arm performs better in a direct comparison. Consider, for example, a soccer match, where we cannot directly observe a numerical value that accurately represents the quality of a soccer team. Instead, it is often easier to observe which team wins against another in a match. To address this scenario, the concept of dueling bandits was introduced [YJ09; Yue+09; Ben+21] and will be described in more detail in the subsequent section.

Problem formulation. In the *Dueling Bandit* or also called *Preference-based Bandit* setting, we are given a set of $n \in \mathbb{N}$ different options, denoted by $[n]$. In contrast to the stochastic MAB problem, the learner can now choose in each time step $t \in \{1, \dots, T\} \subseteq \mathbb{N}$ two arms $i_t, j_t \in [n]$ instead of only one. Thus the action set is given by $\mathfrak{A} = \{(i, j) \mid i, j \in [n]\}$. The observed feedback is binary $r_{(i_t,j_t),t} \in \{0, 1\}$ and can be interpreted as information on whether arm i_t beats arm j_t in a direct comparison which we denote in the following as $i_t \succ j_t$. These direct comparisons are also called duels since you can regard each comparison as a competition of the two chosen arms with only one winner. Note that ties are usually not allowed with only a few exceptions, for instance in [BHH24]. If we allow to choose the same arm $i \in [n]$ twice in one duel, the observed feedback is defined as $r_{(i,i),t} = 1$ with probability $\frac{1}{2}$ and otherwise $r_{(i,i),t} = 0$ which, in expectation, corresponds to deciding the winner by tossing a coin. In Dueling Bandits, we estimate not the rewards of an arm but rather the probabilities that an arm $i \in [n]$ wins against its competitor $j \neq i, j \in [n]$. We denote the winning probabilities as $p_{i,j} \in [0, 1]$. Note that when $p_{i,j} = 0.5$, it is impossible to determine a better arm which motivates the introduction of the *calibrated pairwise probabilities*

$$\Delta_{i,j} := p_{i,j} - \frac{1}{2}.$$

These values define the hardness of the considered problem. If the absolute value of $\Delta_{i,j}$ is large for all $i, j \in [n]$, the problem is easy to solve, while it gets harder as $\Delta_{i,j}$ approaches 0 for some $i, j \in [n]$ because it becomes more challenging for the learner to determine whether arm i or arm j is more likely to win the majority of duels in expectation.

In this setting, defining a best arm is more complex compared to the stochastic MAB problem, which has led to the development of various winner concepts.

- Obviously, a best arm would be the one that wins against every other in each duel in expectation. We call this arm the *Condorcet Winner* (CW) and define it formally as

$$i^* \in \{i \in [n] \mid p_{i,j} \geq 0.5 \; \forall j \in [n]\}.$$

  Note that in practice, it can happen that we have cycles in the winning probabilities and thus a Condorcet Winner does not exist. An example of such a cycle can occur in a soccer tournament, where team A beats team B, team B beats team C and team C beats team A. In this case, a Condorcet Winner does not exist. This is why several other winner concepts were introduced.

- If a Condorcet Winner does not exist or in other words no arm wins against all other arms in expectation, we can weaken the notion of the best arm to the one that wins most frequently the duels against other arms in expectation. This arm is called the *Copeland Winner* and is formally defined as

$$i^*_{Cope} \in \{i \in [n] \mid d_i \geq d_j \; \forall j \in [n]\}, \text{ where } d_i = |\{j \in [n] \mid p_{i,j} > 0.5\}|.$$

- The weakest of the commonly used definitions for a best arm is the so-called *Borda Winner* which is the arm that maximizes the average winning probability across all other arms. Formally, we have

$$i^*_{Borda} \in \text{argmax}_{i \in [n]} \frac{1}{n-1} \sum_{j \in [n], j \neq i} p_{i,j}.$$

A common **goal** in the Dueling Bandit setting is to identify and return one of the best arms defined above with minimal sample complexity or maximal confidence. Similar to the stochastic MAB setting, the problem of identifying the best arm is also called the *best arm identification* problem or *pure exploration* problem in Dueling Bandits.

Analogously to the stochastic MAB setting, another goal in Dueling Bandits could be to suffer minimal expected regret during a specific or even infinite amount of duels. For this, we first have to clarify how to define the expected regret in the Dueling Bandit setting. Similar to the different notions for a best arm, we also can define various types of the expected regret for Dueling Bandits. In fact, three different kinds of regret measures are established in the Dueling Bandit literature that are presented in the following.

- The *strong regret* is the sum of the calibrated pairwise probabilities of the overall best arm over the worst of the two chosen arms at each time step. In

other words, it is only 0 in each time step if both of the chosen arms in that time step coincide with the best arm. We write

$$R_T^{strong} = \sum_{t=1}^{T} \max\{\Delta_{i^*,i_t}, \Delta_{i^*,j_t}\}.$$

From a learning perspective, choosing the same arm twice for a duel does not make sense, since we gain no new information about the winning probability of one arm over another. Each time the learner wants to explore some new information, he automatically suffers some strong regret.

- The *average regret* sums up the average of the calibrated pairwise probabilities of the overall best arm over both chosen arms in each time step. Formally,

$$R_T^{avg} = \sum_{t=1}^{T} \frac{1}{2} \left(\Delta_{i^*,i_t} + \Delta_{i^*,j_t}\right).$$

While trying to minimize the average cumulative regret, it still makes sense for the learner to try to choose two arms that are as good as possible, since both are contributing equally to the suffered regret in that time step.

- The *weak regret* only sums up the calibrated pairwise probability of the overall best arm over the better one of the chosen arms in each time step. We have

$$R_T^{weak} = \sum_{t=1}^{T} \min\{\Delta_{i^*,i_t}, \Delta_{i^*,j_t}\}.$$

When the learner aims to minimize the weak cumulative regret, it is enough to choose one good arm in each duel. The second arm can be chosen arbitrarily in that time step and does not play any role for the suffered weak regret.

Note that we get an induced hierarchy of the regrets of $r_T^{weak} \leq r_T^{avg} \leq r_T^{strong}$.

Example. Returning to the example from the stochastic MAB section about choosing the best restaurants for dining, it might be challenging for the user to rank the quality of a restaurant on a numerical scale. Instead, the user can more easily express a preference between two restaurants by comparing their food directly. To make such a comparison, the user must visit two restaurants each day and can then decide whether the first restaurant was better or worse than the second one.

In this scenario, it is typically assumed that the quality of the restaurants is transitive. This means that if restaurant A is better than restaurant B, and restaurant B is better than restaurant C, then the user should also prefer restaurant A over restaurant C. A possible goal in this example could be to identify the Condorcet Winner — in other words, the restaurant that is better than all others in direct comparisons.

Theoretical Guarantees. Similar to stochastic MABs, also in Dueling Bandits it is common to prove a bound for one or multiple of the different types of the expected regret. In addition to this and again similarly to stochastic MABs, there exist some bounds of the expected regret that only hold with high probability in the form that $\mathbb{P}(R_T < z_{Q,n,T}) \geq 1 - \delta$ where $\delta \in (0, 1)$ represents the allowed failure probability. The bound $z_{Q,n,T}$ usually somehow depends on the $n \times n$ pairwise winning probability matrix $Q = (p_{i,j})_{(i,j)\in\mathfrak{A}}$, the number of arms n and the time horizon T.

For the best arm identification problem a common aim is to derive a bound on the sample complexity, where, in our context, sample complexity means the number of duels required to identify the winner. Again this might not be guaranteed with certainty but only with a high probability for an allowed error probability.

In the *probably approximately correct* (PAC) learning scenario the goal is to identify an arm that is near-optimal in the sense that its winning probability is at most an ϵ worse than the winning probability of the overall best arm i^* or formally we call $\hat{i}$ an *ϵ-optimal arm* if

$$\forall j \in [n]: \quad p_{i^*,j} - p_{\hat{i},j} \leq \epsilon.$$

This ϵ-optimal arm should then be identified with a probability of at least $1 - \delta$ by an algorithm to classify it as (ϵ, δ)-PAC-optimal. Note that several notions of an ϵ-optimal arm exist and the above definition is only one possible out of multiple ones. Another possible definition for an ϵ-optimal arm is that its winning probabilities are only a factor $1 + \epsilon$ worse than the winning probabilities of the overall best arm i^* like in [Mas+20]. However, we will stick to the more commonly used first and additive definition of ϵ-optimality in the following.

Existing Methods. First of all, it is worth mentioning that any stochastic MAB problem can easily be transformed into a Dueling Bandit problem and solved by a Dueling Bandit algorithm in the following way. If we can observe two numerical rewards $r_{i,t}, r_{j,t} \in \mathbb{R}$ in the stochastic MAB setting we can simply define each possible pair of arms (i, j) for $i, j \in [n]$ as the action set $\mathfrak{A}$ and after pulling two arms $(i, j) \in \mathfrak{A}$ we can define the binary winner feedback as $r_{(i,j),t} := 1_{\{r_{i,t} \geq r_{j,t}\}}$ from the observed numerical rewards which can then be used as the observation in the Dueling Bandit setting.

Due to the similarity of the stochastic MAB and the Dueling Bandit setting, some algorithms presented earlier in the stochastic MAB section 3.1 can be adapted to the Dueling Bandit setting. For example, for the UPPER CONFIDENCE BOUND algorithm, there exists a preference-based version, called RELATIVE UCB [Zog+14], to solve the best arm identification in Dueling Bandits. It works by choosing in each time step the first arm uniformly at random from all arms that can still be the Condorcet Winner which are the ones that have estimated winning probabilities $\geq \frac{1}{2}$ against any other arm. The second arm is then chosen as the best competitor in the face of uncertainty or in other words, the arm with the highest upper confidence bound on

the probability that it beats the first chosen arm.
As an example assume that arm $i \in [n]$ was selected as the first arm with $p_{i,j} \geq \frac{1}{2} \ \forall j \in [n]$ and in Figure 3.2 are the pairwise winning probabilities $p_{i,j}$ shown of arm i against the other arms inclusive the colored confidence bounds of the pairwise winning probabilities. The most promising competitor for arm i would then be the orange arm, thus the algorithm executes a duel between arm i and the orange arm. Depending on the observed winner, the estimated winning probabilities are updated afterwards. The authors provide a cumulative regret bound for their algorithm of the order $O(n \log T)$ at any time step $T \in \mathbb{N}$.

A common assumption for a best arm identification algorithm in Dueling Bandits is that a Condorcet Winner exists which is, as described above, not always the case and in addition far from trivial to check this assumption beforehand. The same problem occurs for the assumption of transitivity in the sense that if arm i beats arm j and arm j beats arm k, then also arm i beats arm k.
To overcome this problem [Had+21] introduced the so-called TESTIFICATION of the Condorcet Winner in which the algorithm identifies a Condorcet Winner while simultaneously testing whether one exists. Otherwise, it will simply return the statement that no Condorcet Winner exists.

A commonly used algorithm for weak regret minimization in the Dueling Bandits is the so-called WINNER STAYS algorithm [CF17]. It chooses one arm uniformly at random from the arms that have a high winning probability and duels it against each other arm. If some arm can beat it with high probability, then this arm will be taken over to the next duel as the arm that has to be beaten now by any other. With this strategy, you always have the arm that is currently assumed to be the best one contained in each duel and thus you suffer 0 weak regret in each time step at least if the estimated best arm is correct. To be more precise, the authors derived a weak regret bound of $O(n \log n)$ that is independent of the runtime, if a total order of the arms exists.

3.3 Combinatorial Bandits

Motivation. Due to the growing amount of computational resources, in recent times, it has become relatively easy to simulate multiple arm pulls in parallel during each time step in empirical studies. In fact, to generalize the stochastic Multi-Armed Bandits and Dueling Bandits further, we can consider scenarios where a set of $K \in \mathbb{N}$, $K \geq 2$ arms are played together in each time step. Coming back to the example of restaurant preferences, we can now assume that we can visit multiple restaurants per day, for example, one for breakfast, one for lunch, and one for dinner. At the end of each day, we decide which of the visited restaurants had the best food. The example of sports tournaments can also be extended to this case if we consider now sports with not only two teams or players competing against each other but multiple ones. For instance, in a bicycle race in which multiple riders race against each other

and try to reach the finish line as fast as possible, and, most importantly, faster than all current competitors. Usually, the best riders get assigned points for the world ranking afterwards depending on their positions, where the winner of the race receives the highest number of points, and the worse the position the fewer points are given to the athlete. In this example, the goal is to order all cyclists according to their racing abilities as reliably as possible for the world ranking. However, there are various goals possible in this setting. For example, a sponsor is not interested in a ranking of all cyclists but is only interested in knowing who is the best and wins the most races on average. If the sponsor places his advertisements on the jersey of the best athlete, he will get the most attention. We will call this setting the Combinatorial Bandits and explain it in the following in detail.

Problem formulation. In the *Combinatorial Bandit* setting we are again given a set of $n \in \mathbb{N}$ different alternatives which we denote by their indices $[n] = \{1, \dots, n\}$. At each time step a subset of arms, referred to as a *superarm* $S_t = \{i_{t,1}, \dots, i_{t,K}\} \subseteq [n]$ is chosen which contains exactly $K \in \mathbb{N}$ arms. In other words, the action set can be defined as $\mathfrak{A} = \{(i_1, \dots, i_K) \mid i_1, \dots, i_K \in [n]\} \subseteq 2^{[n]}$, where $2^{[n]}$ is the set of all possible subsets of $[n]$.

With a larger number of arms pulled in each time step, we also have more options for the received feedback afterwards. The commonly used definitions as described in [CWY13] or [ABL11] are as follows.

- In the *semi-bandit feedback* a numerical reward is observed for each arm contained in the superarm. If we have $S_t = \{i_{t,1}, \dots i_{t,K}\} \in \mathfrak{A}$ in time step t, then we can observe a vector of reward values $r_t = (r_{i_{t,1},S_t,t}, \dots, r_{i_{t,K},S_t,t}) \in \mathbb{R}^K$, where the rewards $r_{i,S_t,t}$ for each arm $i \in S_t$ are sampled (not necessarily independently) from an underlying probability distribution for each arm that may be dependent on the superarm S_t.

- In the *full bandit feedback*, the learner observes a numerical reward not only for each arm in the selected superarm but also for all other arms $i \in [n]$. Thus, we can observe exactly n different numerical values $r_{i_j,S_t,t} \in \mathbb{R}$ at time step t after pulling superarm $S_t \in \mathfrak{A}$ for $j \in \{1, \dots, n\}$. These rewards are sampled from an underlying probability distribution for each arm that again can in principle depend on the selected superarm S_t.

- In the *bandit feedback* it is only possible to observe the sum of the numerical rewards for every chosen arm. In other words, let $r_{i,S_t,t} \in \mathbb{R}$ the (not necessarily independently) sampled numerical reward for each arm $i \in S_t$ contained in the superarm S_t. Then we can only observe an overall reward of $r_t = \sum_{i \in S_t} r_{i,S_t,t}$ after playing the superarm S_t.

Note that for the special case of a subset size of $K = 1$ and the semi-bandit feedback, we are in the same setting as in the stochastic Multi-Armed Bandits again, so

the Combinatorial Bandit setting can be viewed as a **generalization of the MAB** setting.

On the other hand, assume we have only binary feedback in the semi-bandit feedback scenario, thus, for each arm $i \in S_t$ contained in the superarm S_t, we receive a reward $r_{i,S_t,t} \in \{0,1\}$. In addition, a reward of 1 is assigned exactly to one arm in S_t; in other words $\sum_{i \in S_t} r_{i,S_t,t} = 1$ in each time step $t \in \mathbb{N}$. Each play of a superarm can then be interpreted as a multi-duel among the K selected competitors. The preference-based feedback indicates the winner $i' \in S_t$ of the multi-duel that has a reward of $r_{i',S_t,t} = 1$ while for every other arm $i \in S_t$ with $i \neq i'$, we observe the reward of $r_{i,S_t,t} = 0$. In the special case, that we have a subset size of $K = 2$, we have exactly the same setting as in the Dueling Bandits. For a general subset size of $K \in \mathbb{N}$ with $K \geq 2$, we can interpret the Combinatorial Bandits as a **generalization of the Dueling Bandits**. Note that the terminology for this setting varied in the related literature and is referred to as, for example, multi-dueling [Sui+17], battling [SG19], choice [AJA20] or preselection bandits [BH20]. We will stick to the term Combinatorial Bandits with preference-based semi-bandit feedback.

Depending on the feedback scenario we consider, various types of regret measures can be defined. These are typically defined by adapting and transferring the weak, average, or strong regret measures introduced in the Dueling Bandit section to the Combinatorial Bandit setting. To describe this in detail would be out of the scope of this thesis.

In alignment with the various types of feedback and regret measures, also different types of winner concepts exist.

- The first option is the aim to find a *single best arm*. To define this, the winner concepts from Dueling Bandits can be extended to the Combinatorial Bandit setting. For example, we call the arm that wins all multi-duels in which it participates, the *Generalized Condorcet Winner*, but we can also consider a *Generalized Copeland Winner* that wins most of the multi-duels or the *Generalized Borda Winner* that has the highest winning probability in a multi-duel on average. Note that in particular in the scenario where we observe some numerical reward per chosen arm or per chosen superarm, it is possible to define many more sensible winner concepts, e.g. the arm with the highest average reward in the semi-bandit or full-bandit feedback scenario or the arm that contributes the most to the reward in the bandit feedback scenario.

- The second option is the aim for learning a *ranking of the arms*. This setting typically assumes an underlying probabilistic model that is for example represented by some underlying utility for each arm that somehow values the quality of the associated arm. Using these utility values, which are learned by the method, one can compute the position of the corresponding arms in a ranking e.g. by using the Plackett-Luce [Pla75; CL59] or more generally, the Bradley-Terry model [BT52].

- The third option is to find the *best superarm* that contains the K arms yielding optimal feedback. In the bandit feedback, this is for example easily identifiable as the one with the highest reward. For the semi-bandit feedback with binary rewards, it suffices to include a Generalized Condorcet Winner in the superarm, and the rest of the arms can in principle be chosen arbitrarily. For semi-bandit feedback with numerical rewards, the task of finding the best superarm is related to the task of finding a ranking of the arms since naturally, the set of the best K arms builds the best superarm if such a ranking exists.

Example. Assume you are an online shop provider with limited space to display articles to consumers at each time step. In this scenario, the articles in the online shop correspond to the arms, and the subset of articles displayed to the consumer represents the superarm. Feedback is given as follows: a value of 1 is assigned to the item chosen and purchased by the consumer, while all other displayed items receive a feedback value of 0, indicating they were not purchased. Thus, we are in the semi-bandit feedback scenario with binary rewards.

This problem can be interpreted as a weak regret minimization problem. Since the online shop operates over a long time horizon, the objective for each visitor is to display a set of items that ensures the consumer purchases at least one, thereby generating profit for the shop. The weak cumulative regret for a given time step is 0 if the consumer decides to buy at least one article, which aligns with the shop owner's goal.

Theoretical Guarantees. In the Combinatorial Bandit setting, similar theoretical guarantees to those in the Dueling Bandit setting exist. For example in the regret minimization problem, some regret bounds were derived, e.g. for the UCB generalization algorithm, the THRESHOLDING RANDOM CONFIDENCE BOUND which has an expected utility regret bound of $O(\sqrt{KT\log(T)})$ [BH20].

For the (near-) optimal arm identification problem, there exist some proofs for the necessary budget of arm pulls in the (ϵ, δ)-PAC-learning scenario. For example, the SUCCESSIVE HALVING generalization HALVING BATTLE [SG18] needs a number of pulls in $O(\frac{K}{\epsilon^2}\log(\frac{1}{\delta}))$ to identify an ϵ-optimal arm with probability at least $1-\delta$ under specific assumptions, such as the existence of an underlying Plackett-Luce model for preference probabilities. Another Successive Halving generalization called UNIFORM ALLOCATION [SG18] provides an error probability bound for identifying the best arm within a predefined budget, assuming that rewards adhere to a specific probabilistic structure.

In scenarios where the goal is to find a top-k ranking with high probability, the ALGMULTI-WISE algorithm [CLM18] provides sample complexity bounds for the necessary pairwise comparisons.

Existing Methods. Depending on the kind of feedback you can observe, the existing algorithms for Dueling Bandits can be adapted for Combinatorial Bandits in scenarios with preference-based feedback, while methods from stochastic Multi-Armed Bandits are suitable for adaptation to numerical reward feedback. This is possible for both problems, for the best arm identification as well as for the regret minimization, as far as the used concepts for best arms and regret coincide. Thus, there are several existing algorithms for the Combinatorial Bandit setting, addressing a wide range of feedback types, regret measures, and winner concepts.

Famous examples of the existing algorithms for regret minimization in Combinatorial Bandits with preference-based feedback is INDEPENDENTSELFSPARRING [Sui+17] which is inspired by the THOMPSON SAMPLING algorithm and maintains a Beta distribution for the winning probability of each arm. Based on these distributions, the algorithm samples arms for selection in each time step. It is proven that the arm estimated as the best by the algorithm converges against the true best arm as the time horizon approaches infinity $T \to \infty$.

For numerical feedback, there exists an extension of the UCB framework called MULTI-DUELING [Bro+16], where confidence bounds on the estimated mean reward of each arm are constructed based on the number of pulls. In each time step, the algorithm selects the K arms with the highest upper confidence bounds for evaluation.

For best arm identification in the preference-based feedback case, [SG18] introduced TRACE-THE-BEST which selects the currently most preferred arm together with $K-1$ randomly chosen arms. This approach exploits the best arm while maintaining exploration of other arms.

3.4 Non-stochastic setting

Motivation. To further generalize the stochastic MAB, Dueling Bandit, and Combinatorial Bandit settings, it is useful to relax the common assumption of an underlying stochastic process that generates the observed feedback. This generalization addresses more practical scenarios where such stochastic distributions might not exist or adequately represent the feedback generation process.

In many real-world applications, the observations might be generated by adversarial processes, or the feedback might only satisfy a weaker assumption of convergence to a limiting value for each arm over time. This relaxation broadens the applicability of bandit frameworks to non-stationary or adversarial environments, where traditional stochastic models fail to capture the complexity of the observed data. In this section, we will focus on the case and examine settings where the observations for each arm converge to a limit over time, rather than being generated by fixed stochastic distributions.

Formulation. We extend the MAB, Dueling, and Combinatorial Bandit frameworks by considering a non-stochastic feedback setting, where the observed feedback is generated differently from the stochastic case. Instead of rewards sampled from a stochastic distribution, we allow rewards $r_{i,t}$ for a (super-)arm $i \in \mathfrak{A}$ after $t \in \mathbb{N}$ many pulls to be generated without assuming an underlying stochastic distribution in the *non-stochastic feedback* setting. The only assumption is that the time-averaged rewards for each arm converge to a limit value:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{s=1}^{t} r_{i,s} \to \nu_i \in \mathbb{R} \quad \forall i \in \mathfrak{A}.$$

Note that the setting of an underlying stochastic reward distribution is a special case of the non-stochastic setting since the arithmetic mean of the observed rewards also converges against the expected value of the reward distribution for each arm according to the law of large numbers. Thus we can regard the non-stochastic setting as a generalization of the stochastic setting.

The assumption that such a limit exists automatically implies the existence of a convergence speed. Thus we have a non-increasing function $\gamma : \mathbb{N} \to \mathbb{R}$ with

$$\left| \frac{1}{t} \sum_{s=1}^{t} r_{i,s} - \nu_i \right| \leq \gamma(t) \quad \forall i \in \mathfrak{A}.$$

This can be seen as a generalization of the law of large numbers in the stochastic setting and is an important ingredient for the theoretical analysis of algorithms that can solve the non-stochastic best arm identification problem. We can interpret the convergence speed as an indicator of the hardness of the best arm identification problem. The steeper the function γ decreases, the faster the rewards converge against its limit values, and the fewer samples of the rewards are needed to get a reliable estimation of the limit value, thus the easier the problem is to solve. However, the inverse of γ can be used to determine the number of arm pulls that are necessary to distinguish the quality of the arms with high probability in a theoretical analysis. It is worth mentioning that in the case of Combinatorial Bandits, the rewards of a single arm in the semi-bandit or full-bandit feedback scenario can depend on the chosen superarm $S_t \in \mathfrak{A}$ at this time. For simplicity, we omitted the notation of these dependencies.

Example. In some cases, problems that appear to be stochastic best arm identification problems can be more effectively addressed by relaxing the i.i.d. assumption for the observations. A notable example is hyperparameter optimization, where the goal is to identify the best internal parameters for a machine learning algorithm. In such scenarios, it can be advantageous to go beyond simple random restarts by selecting the initial conditions strategically or dynamically allocating resources based on the current performance of the evaluated configurations. Following this approach, the performance metric of the target algorithm under a given configuration is no longer an i.i.d. sample of its true quality.

Existing Research. The pure exploration problem in non-stochastic MABs with fixed budget has been studied in [JT16]. The authors derived a necessary budget to ensure the identification of the best arm and extended their results to a PAC-learning setting. In this setting, they guarantee the identification a near-optimal arm with a high probability, provided the budget is sufficiently large.

In the case of feedback generated by an adversary, there exists an extensive body of literature. For example, [BS12] and [Abb+18a] proposed a framework designed to perform well in both scenarios: for stochastic feedback and for adversarial feedback.

4 State-Of-The-Art and Contributions

In this chapter, we briefly discuss the relevant related work and the state-of-the-art in the field related to our contribution. In addition, we outline the specific contributions of this thesis, highlighting how our methods differ from and improve upon existing approaches.

4.1 Finding Optimal Arms in Non-stochastic Combinatorial Bandits

Combinatorial Bandits with preference-based feedback. As described in Section 3.3, there exists already a huge amount of literature tackling the best arm identification problem in Combinatorial Bandits with preference-based feedback. Famous examples are TRACE-THE-BEST [SG19], in which the currently estimated best arm is dueled against a random subset for a specific amount of time steps and afterwards the worst arms are eliminated from the possible winner candidates iteratively. To return an ϵ-optimal arm with probability at least $1-\delta$ for given ϵ and δ, TRACE-THE-BEST needs a sample complexity of order $O\left(\frac{n}{\epsilon^2}\log\frac{n}{\delta}\right)$.

Another algorithm solving the best arm identification problem in Combinatorial Bandits with fixed confidence is DIVIDE-AND-BATTLE [SG19]. It randomly divides all arms into subsets of size K, plays each subset a specific amount of time, and eliminates all arms from the subset except the one that won most frequently. This improves the logarithmic term in the sample complexity to an overall number of sufficient samples of order $O\left(\frac{n}{\epsilon^2}\log\frac{K}{\delta}\right)$.

The above-mentioned extensions of SUCCESSIVE HALVING called HALVING BATTLE [SG19] and UNIFORM ALLOCATION [SG19] both randomly select a subset of K arms, duel them for a specific amount of time and discard the worst half afterwards. While in the HALVING BATTLE the number of pulls per subset is determined based on a given fixed confidence δ and error bias ϵ, UNIFORM ALLOCATION chooses the number of time steps one subset is dueled based on a fixed budget of time steps that is distributed uniformly over all subsets. The authors derived a sample complexity of $O\left(\frac{n}{\epsilon^2}\log\frac{1}{\delta}\right)$ for the HALVING BATTLE which improves upon DIVIDE-AND-BATTLE further. What all of these methods have in common is that they assume a stochastic

feedback mechanism and to our knowledge no algorithm exists that identifies the best arm in Combinatorial Bandits in a non-stochastic preference-based feedback setting.

Combinatorial Bandits with numerical feedback. Best arm identification with fixed confidence in Combinatorial Bandits with numerical semi-bandit feedback was studied by [Jou+21]. In their approach, the action set is derived from a zero-sum two-player game where both aim to minimize the regret based on estimated weights for each arm. However, their work assumes an underlying stochastic distribution for the reward generation process.

The same holds for the literature regarding the bandit feedback setting, such as [DKC20] and [Kur+20]. Moreover, to the best of our knowledge, there is no existing work for Combinatorial Bandits that can cover both, preference-based as well as numerical feedback.

Non-stochastic feedback. A MAB algorithm that finds the best arm in a non-stochastic reward scenario is the well-known SUCCESSIVE HALVING algorithm [JT16] that is described in detail in Section 4.3.1. In [BS12], the authors propose an algorithm designed to cover both settings, the stochastic and non-stochastic feedback scenario. In the context of Dueling Bandits, some works address the non-stochastic feedback scenario like [GUC15] and [SKM20], but both aim for regret minimization and not for best arm identification. However, all of these methods are designed only for Dueling Bandits and none of them can handle multi-duels nor is extended to the Combinatorial Bandit setting yet. There are some first steps to have only one algorithm that can be used to handle Combinatorial Bandits as well as Dueling Bandits done in [SJR16]. However, these algorithms were only developed for preference-based feedback and not for numerical rewards.

Our Contribution. Our work presents a general framework capable of identifying the best arm in Combinatorial Bandits with a variable set size $K \in \mathbb{N}$, also covering the case of Dueling Bandits where only two arms are compared. The framework supports a non-stochastic feedback mechanism, which generalizes the stochastic feedback setting, and it accommodates both preference-based and numerical semi-bandit feedback. Due to our derived lower bound for the problem and our theoretical analysis of our algorithm COMBINATORIAL SUCCESSIVE ELIMINATION (CSE), we can guarantee that the necessary budget for CSE is near-optimal. Our experimental results validate these theoretical guarantees.

[SG19] Aadirupa Saha and Aditya Gopalan. “PAC Battling Bandits in the Plackett-Luce Model”. In: *Algorithmic Learning Theory ALT*. vol. 98. Proceedings of Machine Learning Research. PMLR, 2019, pp. 700–737

[Jou+21] Marc Jourdan et al. “Efficient Pure Exploration for Combinatorial Bandits with Semi-Bandit Feedback”. In: *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*. Ed. by Vitaly Feldman et al. Vol. 132. Proceedings of Machine Learning Research. PMLR, 2021, pp. 805–849

[DKC20] Yihan Du et al. “Combinatorial Pure Exploration with Full-Bandit or Partial Linear Feedback”. In: *AAAI Conference on Artificial Intelligence*. 2020

[Kur+20] Yuko Kuroki et al. “Polynomial-Time Algorithms for Multiple-Arm Identification with Full-Bandit Feedback”. In: *Neural Computation* 32.9 (2020), pp. 1733–1773

[JT16] Kevin Jamieson and Ameet Talwalkar. “Non-stochastic Best Arm Identification and Hyperparameter Optimization”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. vol. 51. Proceedings of Machine Learning Research. PMLR, 2016, pp. 240–248

[BS12] Sébastien Bubeck and Aleksandrs Slivkins. “The Best of Both Worlds: Stochastic and Adversarial Bandits”. In: *Proceedings of the 25th Annual Conference on Learning Theory*. Vol. 23. Proceedings of Machine Learning Research. PMLR, 2012, pp. 42.1–42.23

[GUC15] Pratik Gajane et al. “A Relative Exponential Weighing Algorithm for Adversarial Utility-based Dueling Bandits”. In: *International Conference on Machine Learning (ICML)*. 2015

[SKM20] Aadirupa Saha et al. “Adversarial Dueling Bandits”. In: *International Conference on Machine Learning (ICML)*. 2020

[SJR16] Max Simchowitz et al. “Best-of-K-bandits”. In: *29th Annual Conference on Learning Theory*. Vol. 49. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1440–1489

4.2 AC-Band

Theoretically grounded Algorithm Configuration. As already mentioned in Section 2.2, most methods addressing the AC problem are heuristic with only a few theoretically grounded approaches. As an extension of the first published theoretical method STRUCTURED PROCRASTINATION, [Kle+19] proposed STRUCTURED PROCRASTINATION WITH CONFIDENCE where the idea is still to delay the solve of hard problem instances until later. But instead of selecting the configuration with the smallest mean runtime for the next step, they consider a lower confidence bound on the estimated mean runtime. The authors proved that an (ϵ, δ)-optimal configuration will be identified with high probability if the budget is large enough, and were

able to derive a smaller necessary budget than in STRUCTURED PROCRASTINATION, which itself already needs only a budget that is optimal up to a logarithmic factor in comparison to the lower bound on the budget that is of order $\Omega\left(\text{OPT}\frac{n}{\epsilon^2\delta}\right)$.

In 2019, [WGS19] proposed the theoretical method CAPSANDRUNS which begins by estimating a runtime timeout for each configuration to ensure that a specific fraction of instances can complete within the allotted time. Subsequently, a Bernstein race is conducted over the configurations, where random instances are solved using each configuration. Finally, configurations with excessively high average runtime estimates are eliminated from the set of potential winners. An improved version of this method, called IMPATIENTCAPSANDRUNS [Wei+20], was later published. This version incorporates a "precheck" mechanism during the runtime timeout estimation to discard poorly performing configurations more efficiently. The authors also derived the necessary budget for identifying an $(\epsilon, \delta, \gamma)$-optimal configuration with high probability for specific ranges of ϵ, δ and γ.

Bandits for Algorithm Configuration. Each algorithm solving the best arm identification problem in Multi-Armed Bandits can be adapted to find the best configuration in an Algorithm Configuration setting by considering each configuration as an arm if the number of configurations is finite or if the space of configurations is discretized in some way. The parallel executions of the configurations when racing them against each other can be modeled by a Dueling or Combinatorial Bandit setting. Motivated by this, [Li+17] use the well-known Multi-Armed Bandit algorithm SUCCESSIVE HALVING as a subroutine in their HYPERBAND algorithm for Hyperparameter Optimization. However, HYPERBAND is only applicable to the special case of HPO and not to the general AC setting in which we have to deal with different problem instances. In the general AC setting, not only is a sampling strategy for selecting problem instances necessary, but also the goal of the returned configuration changes to perform well on average over all instances and not only on one specific training dataset. In addition, a general framework that solves the AC problem should be able to deal with different cost functions as a performance measure and not only with the validation loss of the target algorithm.

Our Contribution. Inspired by HYPERBAND, we utilized our previously introduced Combinatorial Bandit algorithm CSE, which generalizes SUCCESSIVE HALVING, to develop an AC framework called AC-BAND. This framework samples a varying number of configurations and calls CSE on these configurations with a specific budget of resources. We successfully transferred the theoretical guarantees of CSE to the PAC scenario and derived the necessary budget for AC-BAND to identify an ϵ-optimal configuration with high probability. In our experiments, AC-BAND demonstrated significantly faster performance compared to other theoretically grounded AC methods, while still returning high-performing configurations.

[Kle+19] Robert D. Kleinberg et al. “Procrastinating with Confidence: Near-Optimal, Anytime, Adaptive Algorithm Configuration”. In: *Neural Information Processing Systems (NeurIPS)*. 2019

[WGS19] Gellert Weisz et al. “CapsAndRuns: An Improved Method for Approximately Optimal Algorithm Configuration”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6707–6715

[Wei+20] Gellert Weisz et al. “ImpatientCapsAndRuns: Approximately Optimal Algorithm Configuration from an Infinite Pool”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 17478–17488

[Li+17] Lisha Li et al. “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816

4.3 Incremental Successive Halving and Incremental Hyperband

Our proposed approach to tackling the HPO problem is based on the HYPERBAND algorithm and its subroutine SUCCESSIVE HALVING. Although both methods are described at a high level in the above chapters, we will go into more detail in the following to clarify how these methods operate and establish the foundational concepts for the proposed work.

4.3.1 Hyperband

A famous method that solves the HPO problem and has its origins in the MAB world is HYPERBAND (HB) [Li+17]. In HPO, one of the major challenges is to find a good tradeoff between the number of HPCs to consider and the corresponding amount of resources like the computation time per HPC if we only have a fixed budget of overall available resources. Sampling too many HPCs and distributing the budget across all of them may result in insufficient resources per HPC and thus in unreliable estimations of the costs of each HPC. On the other hand, sampling too few HPCs may risk too little exploration in the space of HPCs and missing out some (near-) optimal one. HYPERBAND tackles this problem by iterating over a different number of sampled HPCs while keeping the overall budget of resources fixed in each iteration. This leads to a different amount of resources that are assigned to each HPC in the different iterations of HYPERBAND. While in the first iterations, a huge set of HPCs is sampled and only tried out on a few resources, in the last iterations the number of

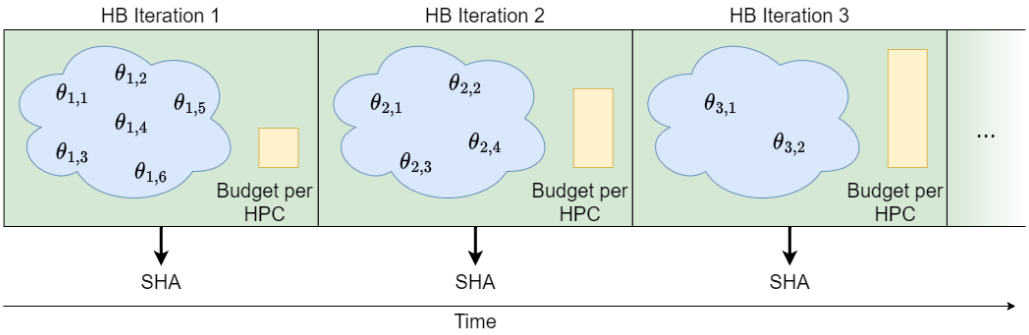


Fig. 4.1.: Illustration of the first iterations of HYPERBAND.

sampled HPCs gets smaller, and thus the allocated resources per HPC get more. An illustration at least of the first iterations of a run of HYPERBAND is shown in Figure 4.1. The sampled sets of HPCs with varying sizes are represented by the blue clouds, while the amount of allocated resources per HPC is depicted by the yellow bars. If we assume that the resources are referring to the runtime in this example, then each iteration of HYPERBAND takes exactly the same time. In each iteration, the Combinatorial Bandit algorithm SUCCESSIVE HALVING (SHA) [JT16] is called as a subroutine with the set of sampled HPCs serving as the arms to be compared, and the available budget of resources determines the number of allowed pulls. As described above in Section 3.1, SHA evaluates all HPCs that are still potential winners in parallel using a predefined budget of resources and eliminates the worst-performing half after each round. This procedure is repeated until there is only one HPC left which is then returned as the winner of the regarded sample set. An example of a run of SHA is shown in Figure 4.2. Each line represents a different arm or in our case a different HPC. The blue boxes are the assigned resources e.g. the runtime for each HPC. While in the first iteration of SHA each HPC gets assigned a small amount of runtime, only the better half of all HPCs is left in the second iteration. These remaining HPCs can then be tested on double of the amount of resources as in the first iteration. This procedure continues until we can decide after the third iteration which one of the two remaining HPCs performed best in the last iteration.

4.3.2 Related Work

Hyperparameter Optimization. In addition to the described approaches in Section 2.3 like grid search [Lar+07], random search [BB12], or Bayesian Optimization [Wu+19], the field of *multi-fidelity optimization* has emerged to tackle the HPO problem. The idea is to allocate more resources to promising HPCs while evaluations of poor-performing HPCs are stopped early. This results in a low-fidelity estimation of the quality of the poor-performing HPCs, because they are only tested on such a small amount of resources that they cannot represent the real-world system of interest. On the other hand, we get high fidelity for well-performing HPCs that are evaluated on a

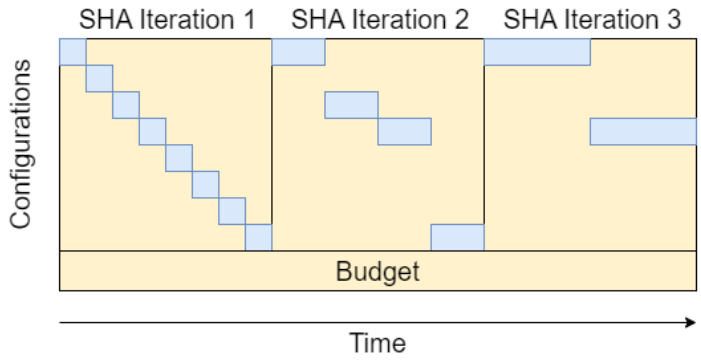


Fig. 4.2.: Illustration of a run of SUCCESSIVE HALVING.

large amount of resources. Note that SUCCESSIVE HALVING and thus also HYPERBAND that are explained in detail in Section 4.3.1 are multi-fidelity optimizers due to their elimination of the worst half of HPCs after each iteration. Thus only the better-performing HPCs are kept into the next iteration and get assigned more resources. There exist some extensions of HYPERBAND such as improving its sampling efficiency in [FKH18] by using Bayesian Optimization to guide the sampling of new HPCs or in [AMH21] with a differential evolution technique. [Men+23] accelerates the evaluation process of the HPCs by canceling the recent Hyperband iteration and jumping to the next iteration if the risk of missing out important information is low. However, all of these extensions still have in common, that the user must predefine the overall budget of resources. If this budget is chosen too small, only too few resources get allocated to the HPCs leading to unreliable cost estimates for each HPC. In the worst case, we observe in hindsight that the budget was not large enough and have to run the whole algorithm with a larger budget again from scratch without reusing the already gathered information.

Successive Halving Extensions. There exists a large body of literature for the best arm identification problem in Multi-Armed Bandits with fixed budget and stochastic feedback, e.g. [CV15; Abb+18b; She19; AKG21; Kat+22]. In comparison to this, the non-stochastic feedback setting has not been extensively explored to date. To our knowledge, the only work that investigated this kind of feedback are [JT16], [Li+17] and our work presented in Section 5. However, the SUCCESSIVE HALVING algorithm has some extensions like ASYNCHRONOUS SUCCESSIVE HALVING (ASHA) [Li+20] that only promotes HPCs asynchronously to the next rung in which it gets assigned more resources. This allows for a better parallelization of the evaluation of HPCs. Another advantage of this method is that the maximum assigned budget of resources per HPC can be adapted during a run of ASHA. However, the algorithmic design of sampling new HPCs step after step and evaluating a specific amount of HPCs before promoting one to the next rung can lead to a huge amount of unnecessary runs and promotions to ensure that a good HPCs gets into a high rung if it is sampled late. A further extension of ASHA is PROGRESSIVE ASHA (PASHA) [Boh+23] that

only introduces higher rungs and dynamically allocates the maximum amount of resources depending on the demand. Both methods, ASHA and PASHA are studied empirically but are lacking theoretical analysis.

Our Contribution. In our work proposed in Section 7, we developed a method called INCREMENTAL SUCCESSIVE HALVING (ISHA), that allows reusing already gathered information from a previous run of the algorithm on a smaller maximal budget. If we detect in hindsight that the chosen budget was too small and decide to run the algorithm again with a larger budget, ISHA does not require evaluating the same number of instances as a new run of SHA. It can compare to the previously evaluated HPCs and fills the newly available slots for samples with additional HPCs. In contrast to ASHA, ISHA uses the approach of synchronous promotion of all HPCs to the next higher rung. By this, we avoid the overload of samples that are necessary to evaluate if a good HPC is sampled late in an asynchronous algorithm and should still get promoted to the highest rung. While the approach seems to be quite simple, it is far from trivial to derive theoretical guarantees that ISHA finds a near-optimal HPC. However, we have included a detailed theoretical analysis of ISHA in our work that states exactly this and in addition, we have proven some theoretical guarantees for ASHA to compare both. Moreover, we have derived the factor of improvement in the sample efficiency of ISHA in comparison to SHA itself. More theoretical guarantees are given for including ISHA as a subroutine in HYPERBAND when enlarging the maximal budget in hindsight which is also tested empirically and demonstrates significant runtime improvements.

[Li+17] Lisha Li et al. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816

[JT16] Kevin Jamieson and Ameet Talwalkar. "Non-stochastic Best Arm Identification and Hyperparameter Optimization". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. vol. 51. Proceedings of Machine Learning Research. PMLR, 2016, pp. 240–248

[Lar+07] Hugo Larochelle et al. "An Eempirical Evaluation of Deep Architectures on Problems with Many Factors of Variation". In: *Proceedings of the 24th International Conference on Machine Learning (ICML)*. Association for Computing Machinery, 2007, pp. 473–480

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305

[Wu+19] Jia Wu et al. "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization". In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40

[FKH18] Stefan Falkner et al. “BOHB: Robust and Efficient Hyperparameter Optimization at Scale”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1437–1446

[AMH21] Noor Awad et al. “DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 2147–2153

[Men+23] Pedro Mendes et al. “HyperJump: Accelerating HyperBand via Risk Modelling”. In: *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI’23/IAAI’23/EAAI’23. AAAI Press, 2023

[CV15] Alexandra Carpentier and Michal Valko. “Simple regret for infinitely many armed bandits”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 1133–1141

[Abb+18b] Yasin Abbasi-Yadkori et al. “Best of both worlds: Stochastic & adversarial best-arm identification”. In: *Proceedings of the 31st Conference On Learning Theory (COLT)*. vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 918–949

[She19] Cong Shen. “Universal Best Arm Identification”. In: *IEEE Transactions on Signal Processing* 67.17 (2019), pp. 4464–4478

[AKG21] Javad Azizi et al. “Fixed-Budget Best-Arm Identification in Structured Bandits”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2021

[Kat+22] Masahiro Kato et al. *Optimal Best Arm Identification in Two-Armed Bandits with a Fixed Budget under a Small Gap*. 2022. arXiv: 2201.04469

[Li+20] Liam Li et al. “A System for Massively Parallel Hyperparameter Tuning”. In: *Proceedings of Machine Learning and Systems*. Vol. 2. 2020, pp. 230–246

[Boh+23] Ondrej Bohdal et al. “PASHA: Efficient HPO and NAS with Progressive Resource Allocation”. In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2023

5 Finding Optimal Arms in Non-stochastic Combinatorial Bandits with Semi-bandit Feedback and Finite Budget

Author Contribution Statement

The idea of this work was developed in a common brainstorming with Viktor Bengs, Björn Haddenhorst and the author. The theoretical works mainly come from the author with support and revisions by Viktor Bengs and Björn Haddenhorst. The lower bound was proven by Björn Haddenhorst and revised by the author and Viktor Bengs. The implementation of the experiments was done by the author and the paper was written mainly by the author with subsequently revisions by all authors.

Supplementary Material

An appendix to the paper is provided in Appendix A. The code of the official implementation is provided at https://github.com/BrandtJasmin/CSE.

Finding Optimal Arms in Non-stochastic Combinatorial Bandits with Semi-bandit Feedback and Finite Budget

Jasmin Brandt[a], **Viktor Bengs**[b], **Björn Haddenhorst**[a], **Eyke Hüllermeier**[b,c]
[a]Department of Computer Science, Paderborn University, Germany
[b]Institute of Informatics, University of Munich (LMU), Germany
[c]Munich Center for Machine Learning, Germany
jasmin.brandt@upb.de, viktor.bengs@lmu.de, bjoernha@mail.upb.de, eyke@lmu.de

Abstract

We consider the combinatorial bandits problem with semi-bandit feedback under finite sampling budget constraints, in which the learner can carry out its action only for a limited number of times specified by an overall budget. The action is to choose a set of arms, whereupon feedback for each arm in the chosen set is received. Unlike existing works, we study this problem in a non-stochastic setting with subset-dependent feedback, i.e., the semi-bandit feedback received could be generated by an oblivious adversary and also might depend on the chosen set of arms. In addition, we consider a general feedback scenario covering both the numerical-based as well as preference-based case and introduce a sound theoretical framework for this setting guaranteeing sensible notions of optimal arms, which a learner seeks to find. We suggest a generic algorithm suitable to cover the full spectrum of conceivable arm elimination strategies from aggressive to conservative. Theoretical questions about the sufficient and necessary budget of the algorithm to find the best arm are answered and complemented by deriving lower bounds for any learning algorithm for this problem scenario.

1 Introduction

The multi-armed bandits (MAB) problem is an intensively studied problem class in the realm of machine learning, in which a learner is facing a sequential decision making problem under uncertainty [32]. A decision (action) corresponds to making a choice between a finite set of specific choice alternatives (objects, items, etc.), also called *arms* in reference to the metaphor of gambling machines in casinos. After each decision to choose a particular arm, the learner receives some form of *feedback* – typically a numerical reward – determined by a *feedback mechanism* of the chosen arm. The learner is not aware of the arms' feedback mechanisms and consequently tries to learn these in the course of time by performing actions according to its learning strategy. The concrete design of its learning strategy depends essentially on two main components of the learning setting: the assumptions on the feedback mechanisms and the learning task.

Traditionally, and even up to now the most prevalent assumption is that the feedback received by choosing one arm is generated by means of a probability distribution of the chosen arm [41, 31]. In this way, any useful learning strategy revolves around learning specific probabilistic features of the arms' distributions such as the means. These features, in turn, quite naturally provide a way to define a notion of *(sub-)optimality of an arm* as well as a *best arm*. A relaxation of this *stochastic setting* is the *non-stochastic setting*, in which no assumption is made in the form of probabilistic laws of the feedback mechanisms. Instead, either no assumptions are made on the feedback mechanisms, so that

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

these can also be generated by an adversary [7], or that the sequence of feedback observations (or a transformation thereof) of an arm converges asymptotically to a fixed point [25]. In the latter case, the notion of an arm's (sub-)optimality is again straightforward, given that the limit points can be ordered, while in the former usually the best action in hindsight plays the role of the best arm.

Regarding the learning task, the most prominent one is that of *regret minimization*, where in each learning round the learner suffers regret unless the optimal decision is made (determined by the feedback mechanisms). The main challenge for the learner is to manage the trade-off between exploration and exploitation, i.e., constantly balancing (i) the degree of new information acquisition about some arms' feedback mechanism in order to appropriately expand the current knowledge base (exploration), and (ii) the degree of choosing arms considered to be optimal given the current knowledge base in order to keep the overall regret low (exploitation). In many practical applications, however, the learning task is of a quite different kind, as the focus is rather on finding the (approximately) correct answer to a problem specific question, e.g., which arm is the (approximately) optimal one, within a reasonable time (number of learning rounds). This *pure exploration* learning task can be considered in two variants, namely the fixed confidence and the fixed budget setting. In the former the learner tries to find the answer within as few as possible learning rounds, while guaranteeing a given pre-determined confidence for the correctness of its returned answer. In the latter, it is the other way around, as the learner is provided with a limit on the possible number of learning rounds (budget) and the confidence for the returned answer should be as high as possible. In both variants the main challenge for designing a suitable learner is to specify a clever exploration strategy for finding the correct answer.

In order to model more complex learning settings in practice, the basic setup of MAB problems has been generalized in various ways, such as incorporating additional side information [3, 6, 17, 2] or infinite number of arms [11, 38], just to name a few. Of special practical interest is the generalization of the basic setup, where the learner is allowed to choose specific sets of arms as its action. Consider as an example the online algorithm selection problem with parallel runs, where for sequentially arriving problem instances one selects a subset of available algorithms (solvers) to be run in parallel in order to solve the current problem instance.

If the feedback received is of a numerical nature, this variant has manifested itself under the term *combinatorial bandits* [14] while for feedback of a qualitative nature this variant can be referred to as *preference-based bandits* as put forward by [8]. Combinatorial bandits are further distinguished with respect to the type of feedback between semi-bandit feedback, where feedback of each single arm in the selected subset is observed, and full bandit feedback, where only some aggregated value of the individual numerical feedback of the arms in the selected subset is observed. Although both combinatorial and preference-based bandits consider a similar action set and for both the learner needs to deal with the possibly exponential size of the action set, the process of learning is quite different due to the nature of the observed feedback. The main reason for this is that in preference-based feedback the mutual correlations that may exist between the arms in the chosen subset play a major role in both the assumption about the feedback mechanisms and the learning task from the outset. In contrast to this, the standard setting in combinatorial bandits with semi-bandit feedback is that the individual reward generation mechanisms are independent of the chosen subsets. However, this modeling assumption is questionable in a couple of practical applications, especially when humans provide the feedback. For example, in opinion polls or rating systems where humans rate a subset of objects (political parties/candidates, products, etc.), it is well known that the ratings of the objects may be affected by context effects, i.e., preferences in favor of an object may depend on what other objects are available. In the fields of economics and psychology, context effects are among others divided into compromise [47], attention [22] and similarity [48] effects.

In this paper, we take a step towards unifying these two variants for the best arm identification (BAI) problem in a pure exploration learning setting with fixed budget and non-stochastic feedback mechanisms. The main motivation for this unification is to derive a general purpose learner, which can tackle the BAI problem in both feedback variants. In this way, for example, one can transform a learning problem with numerical signals into a preference-based learning problem and thus conveniently apply such a general purpose learner. Recent works have demonstrated in two different learning scenarios with numerical feedback that such a transformation has great potential [37, 27].

Needless to say, the main challenge is to unify both feedback variants through suitable abstractions allowing them to be treated as instantiations of the same problem class. This bridge is built by dropping

the common independence assumption (of the chosen arm set) for the numerical combinatorial bandits and abstracting the nature of the observations. Additionally, we simply assume that the learner is provided with an appropriate statistic customized to the explicit nature of the feedback. By appropriate choice of the statistic one obtains the respective setting, e.g., the empirical mean for the case of numerical feedback and relative frequencies for the case of preference feedback.

Our contribution. Under mild assumptions on the asymptotic behavior of these statistics, we derive a proper definition of a best arm a learner seeks to find (Section 2) as well as lower bounds on the necessary budget for this task (Section 3). To the best of our knowledge, such lower bounds are novel for non-stochastic settings and the derivation is rather non-standard due to the combinatorial setup of the problem. We suggest a general algorithmic framework suitable to cover the full spectrum of conceivable arm elimination strategies from aggressive to conservative, which we analyze theoretically regarding the algorithms' sufficient and necessary budget to find the best arm (Section 4). As a consequence, we obtain to the best of our knowledge the first algorithm(s) for non-stochastic preference-based bandits as well as for combinatorial bandits under semi-bandit feedback, in which the individual (numerical) feedback received for an arm depends on the chosen subset due to possibly existing mutual correlations between the arms in the chosen subset. The mild assumptions on the asymptotics of the statistics allow to transfer our theoretical results to the stochastic counterparts of the semi-bandit combinatorial and preference-based bandits (Section 5). We demonstrate the usefulness of the generality of our setting in an experimental study for an algorithm selection problem with parallel runs (Section 6), where once again the transformation of numerical feedback to preference feedback plays a key role. Additional experiments are given in the supplementary material, where also all proofs of the theoretical results are collected.

Related Work. A large body of literature considers the combinatorial bandit problem under preference-based feedback, see [8] for an overview. Although *dueling bandits* [52] has established as an overall agreed term for the scenario with actions of size two, the terminology for action sets of larger sizes is still discordant, e.g., multi-dueling [10], battling [42], choice [4] or preselection bandits [9], mainly due to subtle nuances of the motivating practical applications. While pure exploration settings with a stochastic preference-based feedback haven been considered by a series of works [36, 39, 15, 40, 43, 21], a pure exploration setting under a non-stochastic feedback mechanism as in our case has yet to be studied.

Pure exploration has been intensively studied in the basic multi-armed bandits (MAB) setting with stochastic feedback mechanisms as well, see Section 33.5 in [32] for a detailed overview. The non-stochastic variant of the fixed budget MAB setting is considered in [25], which is the backbone for the well-known Hyperband algorithm [34] and additionally inspired in some part the assumptions we make for our work. Initiated by the work of [12] to design learners for regret minimization frameworks which can perform well in both stochastic and non-stochastic settings, the fixed budget framework has been the subject of research by [1] and [45].

Combinatorial bandits with numerical feedback have been introduced by [14] and [16] in a regret minimization framework. The fixed confidence setting for stochastic combinatorial bandits with semi-bandit feedback is studied in [26], and full bandit feedback in [18, 30]. Finally, the best-of-k bandits game introduced in [46], which in some way unifies the combinatorial bandits with binary set-dependent feedback and preference-based bandits in one joint framework similarly as we do in this work. However, they consider a fixed confidence setting with stochastic feedback mechanisms and do not provide a learner for the dependent arm case, although they derive lower bounds on the worst case sample complexity for this case. The only work assuming a set-dependent feedback mechanism in combinatorial bandits with semi-bandit feedback is in [50], where, however, the regret minimization task is studied under stochastic feedback mechanisms. In summary, there seems to be no existing work which considers a pure exploration setting for combinatorial bandits with non-stochastic or even stochastic semi-bandit feedback, where the (mean) rewards of the arms in the chosen subset of arms depends on the subset. Accordingly, our results provide new theoretical contributions to this field.

2 Problem Formulation

In our setup, we assume a set $\mathcal{A}$ of n arms, which we simply identify by their indices, i.e., $\mathcal{A} = [n] = \{1, \dots, n\}$. For some fixed $k < n$ we denote the set of all possible subsets of arms with size of at least 2 and at most k by $\mathcal{Q}_{\leq k} = \{Q \subseteq \mathcal{A} \mid 2 \leq |Q| \leq k\}$. Further, we assume that for any

$Q \in \mathcal{Q}_{\leq k}$ we can query some feedback, in the form of a feedback vector $\mathbf{o}_Q = (o_{i|Q})_{i \in Q} \in D^{|Q|}$ which in turn can be of numerical or qualitative nature specified by the domain D. If we query a subset of arms Q for t many times, then $\mathbf{o}_Q(t)$ is the corresponding feedback vector. We suppose that we are given a suitable statistic s for the type of observation vectors, which maps a multiset of observations to some specific value relevant for the decision making. With this, $\mathbf{s}_Q(t) = (s_{i|Q}(t))_{i \in Q}$ is the statistic vector derived by the sequence of feedback $(\mathbf{o}_Q(t))_t$ of the query set $Q \in \mathcal{Q}_{\leq k}$, and $s_{i|Q}(t) = s(\{o_{i|Q}(1), \ldots, o_{i|Q}(t)\})$ is the relevant statistic for decision making about arm i in the "context" Q after querying Q for t many times.

Examples. For combinatorial bandits with semi-bandit feedback, the observation $\mathbf{o}_Q(t) = (o_{i|Q}(t))_{i \in Q}$ corresponds to the reward one obtains for each arm $i \in Q$ by using Q for the t-th time, so that in particular $D = \mathbb{R}$. The most natural statistic in this case is the empirical mean given for a multiset O of observations by $s(O) = \frac{1}{|O|} \sum_{x \in O} x$, such that $s_{i|Q}(t) = \frac{1}{t} \sum_{t'=1}^{t} o_{i|Q}(t')$, which is also the arguably most prevalent statistic used in the realm of bandit problems for guiding the decision making process. However, other statistics s such as quantiles or the expected shortfall are of interest as well [13].

In the preference-based bandit setting with winner feedback we observe after the t-th usage of the query set Q only a binary (winner) information, i.e., $o_{i|Q}(t) = 1$ if arm i is preferred over the other arms in Q at "pull" t and $o_{i|Q}(t) = 0$ otherwise, so that $D = \{0, 1\}$. Once again the empirical mean of these binary observations is a quite intuitive choice for the statistic s, as in a stochastic feedback setting the corresponding statistic vector $s_{i|Q}(t) = \frac{1}{t} \sum_{t'=1}^{t} o_{i|Q}(t)$ would converge to the probability vector determining how likely an arm will be preferred over all the other arms in the query set Q. For preference-based bandits with full ranking feedback we observe after the t-th usage of the query set Q an entire ranking of the arms in Q, i.e., $o_{i|Q}(t)$ is arm i's rank among the arms in Q at "pull" t, so that $D = \{1, \ldots, k\}$. In such a case the statistic s might be a positional scoring rule [28].

Goal. The goal of the learner in our setting is to find a or the best arm (specified below) within a fixed budget of at most B samples (numbers of queries). For any Q, write $n_Q(t)$ for the number of times Q has been queried until (including) time t. An algorithm, which tackles the problem, chooses at time t a set $Q_t \in \mathcal{Q}_{\leq k}$ and observes as feedback $\mathbf{o}_{Q_t}(n_{Q_t}(t))$ leading to an update of the relevant statistic vector $\mathbf{s}_{Q_t}(n_{Q_t}(t)) = (s_{i|Q_t}(n_{Q_t}(t)))_{i \in Q_t}$.

Best arm. Inspired by the theoretical groundings of Hyperband [25, 34] for best arm identification problems in numerical bandit problems with non-stochastic rewards, we make the following assumption regarding the limit behavior of the statistics

$$(A1) : \forall Q \in \mathcal{Q}_{\leq k} \; \forall i \in Q \; : \; S_{i|Q} \coloneqq \lim_{t \to \infty} s_{i|Q}(t) \text{ exists.}$$

This assumption is in general slightly looser than assuming (stationary) stochastic feedback mechanisms, as (A1) is fulfilled for many prevalent statistics by means of a limits theorem such as the strong law of large numbers. Conceptionally, our Assumption (A1) is similar to the assumption on the sequence of losses in [25], as both have in common that the statistics (losses in [25]) converge to some fixed point, respectively. However, due to the difference of the action spaces (single arms vs. set of arms) and the nature of the feedback (scalar vs. vector observation), our assumption can be seen as a combinatorial extension of the one in [25].

Given assumption (A1) a straightforward notion of a best arm is obtained by leveraging the idea of a Borda winner from dueling bandits with pairs of arms as the possible subsets to more general subsets of arms. A *generalized Borda winner* (GBW) is then an arm which has on average the largest asympotical statistic, i.e.,

$$i^*_{\mathcal{B}} \in \arg\max_{i \in \mathcal{A}} S^{\mathcal{B}}_i = \arg\max_{i \in \mathcal{A}} \frac{\sum_{Q \in \mathcal{Q}_{=k}(i)} S_{i|Q}}{|\mathcal{Q}_{=k}(i)|},$$

where $\mathcal{Q}_{=k}(i) = \{Q \in \mathcal{Q}_{=k} \mid i \in Q\}$ and $S^{\mathcal{B}}_i$ are the asymptotic Borda scores, i.e., the limits according to (A1) of $s^{\mathcal{B}}_i(t) \coloneqq \frac{\sum_{Q \in \mathcal{Q}_{=k}(i)} s_{i|Q}(t)}{|\mathcal{Q}_{=k}(i)|}$. Similarly, a *generalized Copeland winner* (GCopeW) is an arm i, which wins w.r.t. the asymptotic statistics on average on the most query sets, i.e.,

$$i^*_{\mathcal{C}} \in \arg\max_{i \in \mathcal{A}} S^{\mathcal{C}}_i = \arg\max_{i \in \mathcal{A}} \frac{\sum_{Q \in \mathcal{Q}_{=k}(i)} \mathbb{1}\{S_{i|Q} = S_{(1)|Q}\}}{|\mathcal{Q}_{=k}(i)|}.$$

However, both these notions of best arm have two major drawbacks, as there might be multiple GBWs and GCopeWs and due to averaging over all subsets in their definition, there is no way to identify a GBW or a GCopeW within a sampling budget of $o(\binom{n-1}{k-1})$ in the worst case (see Theorem 3.1).

In light of these drawbacks, we specify another reasonable notion of a best arm, for which we leverage the concept of the *generalized Condorcet winner* [21, 4] from the preference-based bandits literature. For this purpose, we introduce the following assumption

$$(A2): \exists i^* \in \mathcal{A} \text{ such that } \forall Q \in \mathcal{Q}_{\leq k}(i^*)\ \forall j \in Q \setminus \{i^*\} \text{ it holds that } S_{i^*|Q} > S_{j|Q},$$

where $\mathcal{Q}_{\leq k}(i) = \{Q \in \mathcal{Q}_{\leq k} \mid i \in Q\}$ for $i \in [n]$. We call i^* the generalized Condorcet winner (GCW), which is the arm dominating all the other arms in each possible query set containing it. It is worth noting that such an arm may not exist, but if it exists, then it is arguably the most natural way to define the optimal arm, even though it may differ from the GBW. Nevertheless, the existence of the generalized Condorcet winner (or simply the Condorcet winner for the case $k = 2$) is a common assumption in the preference-based bandits literature [4, 21, 8]. Additionally, we will show below that identifying a GCW is possible for a sampling budget of size $\Omega(n/k)$ even in worst case scenarios.

Problem characteristics. In light of (A1) and (A2), there are two key characteristics which will determine the appeal of any learner in our setting. The first one is the speed of convergence of the statistics $s_{i|Q}$ to their limit values $S_{i|Q}$. More precisely, the function $\gamma_{i|Q} : \mathbb{N} \to \mathbb{R}$, which is the point-wise smallest non-increasing function fulfilling $|s_{i|Q}(t) - S_{i|Q}| \leq \gamma_{i|Q}(t)$ for any $t \in \mathbb{N}$, plays a major role in characterizing the difficulty of the learning problem. Moreover, the worst speed of convergence function of a query set $Q \in \mathcal{Q}_{\leq k}$ given by $\overline{\gamma}_Q(t) \coloneqq \max_{i \in Q} \gamma_{i|Q}(t)$ and the overall worst speed of convergence function $\overline{\gamma}(t) \coloneqq \max_{Q \in \mathcal{Q}_{\leq k}} \overline{\gamma}_Q(t)$ will be of relevance as well. Assuming a stochastic setting, the role of $\gamma_{i|Q}$ is played by the minimax rate of convergence of the statistic to its population counterpart, e.g., $1/\sqrt{t}$ for the empirical mean and the expected value. Usually the speed of convergence functions will appear implicitly by means of their (quasi-) inverses given by $\gamma_{i|Q}^{-1}(\alpha) \coloneqq \min\{t \in \mathbb{N} \mid \gamma_{i|Q}(t) \leq \alpha\}$, $\bar{\gamma}_Q^{-1}(t) \coloneqq \min_{i \in Q} \gamma_{i|Q}^{-1}(t)$ and $\bar{\gamma}^{-1}(t) \coloneqq \min_{Q \in \mathcal{Q}_{\leq k}} \bar{\gamma}_Q^{-1}(t)$.

The other relevant problem characteristic are the gaps of the limits statistics, i.e., $\Delta_{i|Q} \coloneqq S_{i^*|Q} - S_{i|Q}$ for $i \in [n]$, $Q \in \mathcal{Q}_{\leq k}(i) \cap \mathcal{Q}_{\leq k}(i^*)$. Such gaps are prevalent in the realm of bandit problems, as they can be used to define a measure of (sub-)optimality of an arm in the stochastic feedback case. Note that in our setting this is not straightforward, as the gaps are depending on the query set and more importantly the speed of convergence has a decisive impact on the severeness of these gaps.

3 Lower Bounds

Let us abbreviate $\mathbf{S} \coloneqq (S_{i|Q})_{Q \in \mathcal{Q}_{\leq k}, i \in Q}$ and $\boldsymbol{\gamma} \coloneqq (\gamma_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$. Given $\mathbf{S}$ and $\boldsymbol{\gamma}$, write $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$ for the set of all $\mathbf{s} = (s_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$ that fulfill

(i) $\forall Q \in \mathcal{Q}_{\leq k}, i \in Q$: $S'_{i|Q} = \lim_{t \to \infty} s_{i|Q}(t)$ exists,
(ii) $\forall Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}$: $|s_{i|Q}(t) - S'_{i|Q}| \leq \gamma_{i|Q}(t)$,
(iii) $\forall Q \in \mathcal{Q}_{\leq k}$: $\exists \pi_Q : Q \to Q$ bijective such that $S'_{i|Q} = S_{\pi(i)|Q}$ for all $i \in Q$.

For Q and $l \in \{1, \ldots, |Q|\}$ write $S_{(l)|Q}$ for the l-th order statistic of $\{S_{i|Q}\}_{i \in Q}$, i.e., $\{S_{i|Q}\}_{i \in Q} = \{S_{(l)|Q}\}_{l \in Q}$ and $S_{(1)|Q} \geq \cdots \geq S_{(|Q|)|Q}$. If Alg is a (possibly probabilistic) sequential algorithm, we denote by $B(\mathrm{Alg}, \mathbf{s})$ the number of queries made by Alg before termination when started on $\mathbf{s}$. In the following, we provide lower bounds on $\mathbb{E}[B(\mathrm{Alg}, \mathbf{s})]$ for algorithms Alg, which identify, for any instance $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$, almost surely one GCW resp. GBW resp. GCopeW of $\mathbf{s}$.

Theorem 3.1. *Let* $\mathbf{S}$ *be such that* $S_{(1)|Q} > S_{(2)|Q}$ *for all* $Q \in \mathcal{Q}_{\leq k}$.
(i) There exists $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$ *such that if* Alg *correctly identifies a GCW for any instance in* $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$*, then*

$$\mathbb{E}\left[B(\mathrm{Alg}, \mathbf{s})\right] \geq \left\lceil \frac{n}{k} \right\rceil \min_{Q \in \mathcal{Q}_{\leq k}, j \in Q} \gamma_{j|Q}^{-1}\left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}\right).$$

(ii) Assume $(S_{(1)|Q}, \dots, S_{(|Q|)|Q})$ does not depend on Q for any $Q \in \mathcal{Q}_{=k}$. If Alg *correctly identifies a GBW for any instance in $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$, then*

$$\sup_{\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})} \mathbb{E}\left[B(\mathrm{Alg}, \mathbf{s})\right] = \Omega\left(\binom{n-1}{k-1}\right).$$

(iii) If Alg *correctly identifies a GCopeW for any instance in $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$, then it fulfills*

$$\sup_{\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})} \mathbb{E}\left[B(\mathrm{Alg}, \mathbf{s})\right] = \Omega\left(\binom{n-1}{k-1}\right).$$

The theorem is proven in Section B in the supplement, where we provide in fact slightly stronger versions of these bounds.

4 Algorithms

In this section, we present a class of algorithms along with three possible instantiations solving the corresponding learning task for the case of the generalized Condorcet winner being the best arm. In particular, we analyze all algorithms theoretically in terms of their sufficient and necessary budget to find the respective best arm. In light of the results in Theorem 3.1 for GBW and GCopeW identification, it is straightforward that a simple algorithm, which enumerates all possible subsets, pulls each of them equally often in a round-robin fashion, and returns the empirical GBW (or GCopeW) is already optimal. For sake of completeness, we show this result for the case of GBW identification in Section C and also include this simple algorithm in the experimental study below (called RoundRobin).

4.1 Generalized Condorcet Winner Identification

In the following we introduce a general class of algorithms in Algorithm 1 which is instantiable with various different elimination strategies of the arms. Below, we present some instantiations which build on commonly used elimination strategies in the standard multi-armed bandit setting. The idea of Algorithm 1 is simply to maintain a set of active arms, which is successively reduced by following a specific arm elimination strategy (Algorithm 2) referred to as elimination rounds. In each elimination round $r \in \{1, 2, \dots, R\}$ the set of active arms $\mathbb{A}_r$ is partitioned into P_r many sets of size k (up to a possible remainder set) denoted by $(\mathbb{A}_{r,j})_{j \in P_r}$ for which the elimination strategy is applied with a roundwise-dependent budget b_r. The budget allocated to a partition $\mathbb{A}_{r,j}$ in round r is of the form $b_r = \lceil B/(R \cdot P_r) \rceil$ following the idea to split up the available budget equally first for each round and second for all partitions in each round. The explicit arm elimination strategy used in Algorithm 1 is specified by Algorithm 2 and corresponds to pulling the chosen query set Q for a fixed number of times and afterwards keeping only the best $f(|Q|)$ arms of Q. Here, $f : [k] \to [k]$ is an arbitrary function with $f(x) \leq x - 1$ for all x, which essentially determines the aggressiveness or conservativeness of an arm elimination strategy as we will see below.

Algorithm 1 Combinatorial Successive Elimination

Input: set of arms $[n]$, subset size $k \leq n$, sampling budget B, a function $f : [k] \to [k]$, sequence $\{P_r\}_r$ (number of partitions at round r), R (number of rounds in total)
Initialization: $\mathbb{A}_1 \leftarrow [n]$, $r \leftarrow 1$

while $|\mathbb{A}_r| \geq k$ **do**
  $b_r \leftarrow \lfloor B/(P_r R) \rfloor$, $J \leftarrow P_r$
  $\mathbb{A}_{r,1}, \mathbb{A}_{r,2}, \dots, \mathbb{A}_{r,J} \leftarrow \mathrm{Partition}(\mathbb{A}_r, k)$
  if $|\mathbb{A}_{r,J}| < k$ **then**
    $\mathcal{R} \leftarrow \mathbb{A}_{r,J}$, $J \leftarrow J - 1$
  else
    $\mathcal{R} \leftarrow \emptyset$
  end if
  $\mathbb{A}_{r+1} \leftarrow \mathcal{R}$
  for $j \in [J]$ **do**
    $\mathcal{R} \leftarrow \mathrm{ArmElimination}(\mathbb{A}_{r,j}, b_r, f(|\mathbb{A}_{r,j}|))$
    $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \mathcal{R}$
  end for
  $r \leftarrow r + 1$
end while
$\mathbb{A}_{r+1} \leftarrow \emptyset$
while $|\mathbb{A}_r| > 1$ **do**
  $\mathbb{A}_{r+1} \leftarrow \mathrm{ArmElimination}(\mathbb{A}_{r+1}, b_r, f(|\mathbb{A}_{r+1}|))$, $r \leftarrow r + 1$
end while

Output: The remaining item in $\mathbb{A}_r$

In the following we provide three possible instantiations of Algorithm 1 inspired by commonly used elimination strategies in the standard multi-armed bandit setting for pure exploration tasks.

Combinatorial Successive Winner Stays. The most aggressive elimination strategy is to keep only the arm with the best statistic (the winner) of each partition in each round and discard all others from the set of active arms for the next round. Concretely, we use $f^{\mathrm{CSWS}}(s) = 1$ for f in Algorithm 1 in this case.

Algorithm 2 ArmElimination$(\mathbb{A}', b, l)$

1: Use $\mathbb{A}'$ for b times
2: For all $i \in \mathbb{A}'$, update $s_{i|\mathbb{A}'}(b)$
3: Choose an ordering $i_1, \ldots, i_{|\mathbb{A}'|}$ of $(s_{i|\mathbb{A}'}(b))_{i\in\mathbb{A}'}$
4: **return** $\{i_1, \ldots, i_l\}$

The resulting instantiation of Algorithm 1 is called *Combinatorial Successive Winner Stays* (CSWS), which has at most $R^{CSWS} = \lceil \log_k(n) \rceil + 1$ many rounds in total (at most $\lceil \log_k(n) \rceil$ rounds in the first while-loop and at most 1 in the second). The total number of partitions in round r is at most $P_r^{CSWS} = \lceil n/k^r \rceil$.

Combinatorial Successive Reject. On the other extreme regarding the aggressiveness of the arm elimination strategy is to dismiss only the worst arm of each partition in each round and keep all others in the set of active arms for the next round. More specifically, we use $f^{\mathrm{CSR}}(s) = s - 1$ for this variant, which can be seen as a variant of the Successive Reject algorithm [5] for best arm identification adopted to the combinatorial bandit problem. Consequently, we call the resulting instantiation of Algorithm 1 the *Combinatorial Successive Reject* (CSR) algorithm, whose number of rounds in the first while-loop is at most $\lceil \log_{k-1/k}(1/n) \rceil$ and in the second at most $k-1$. Overall, we have a maximal number of rounds $R^{CSR} = \lceil \log_{k-1/k}(1/n) \rceil + k - 1$ and a maximal number of partitions per round $P_r^{CSR} = \lceil n(1-\frac{1}{k})^{(r-1)}/k \rceil$.

Combinatorial Successive Halving. As a compromise between the aggressive elimination strategy of CSWS and the conservative elimination strategy of CSR one could discard in every elimination round the worse half of all arms in the partition, i.e., using $f^{\mathrm{CSH}}(s) = \lceil s/2 \rceil$ for f in Algorithm 1. This can be seen as a generalization of the successive halving algorithm [25] adopted to the combinatorial bandit problem we are considering. Thus, the instantiation of Algorithm 1 in this spirit will be called the *Combinatorial Successive Halving* (CSH) algorithm. Note that we have at most $\lceil \log_2(n) \rceil$ rounds in the first while-loop and additional $\lceil \log_2(k) \rceil$ in the second while-loop resulting in at most $R^{CSH} = \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil$ many rounds throughout a run of CSH. Furthermore, we have at most $P_r^{CSH} = \lceil \frac{n}{2^{r-1}k} \rceil$ partitions in round r.

4.2 Theoretical Guarantees

In the following, we derive the sufficient budget for Algorithm 1 to return under assumptions (A1) and (A2) the best arm i^*, i.e., the generalized Condorcet winner. For this purpose, we write $\mathbb{A}_r(i^*)$ for the unique set $\mathbb{A}_{r,j} \in \{\mathbb{A}_{r,1}, \ldots, \mathbb{A}_{r,P_r}\}$ with $i^* \in \mathbb{A}_{r,j}$ and define $\Delta_{(l)|Q} = S_{i^*|Q} - S_{(l)|Q}$ for any $Q \subseteq [n]$ with $i^* \in Q$.

Theorem 4.1. *Assume P_r, R are such that Algorithm 1 called with B does not exceed B as total budget. Under Assumptions (A1) and (A2) Algorithm 1 returns i^* if B is larger than*

$$z\left(f, R, \{P_r\}_{1\le r\le R}\right) \coloneqq R \max_{r\in[R]} P_r \cdot \left\lceil \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\Delta_{(f(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}/2\right)\right\rceil$$

The following theorem indicates optimality of z in the theorem above (cf. Sec. D.2 for the proofs).

Theorem 4.2. *For any distinct asymptotic values $\mathbf{S}$, there exists a family of statistics $\{s_{i|Q}(t)\}_{t\in\mathbb{N},Q\in\mathcal{Q}_{\le k},i\in Q}$ with $s_{i|Q}(t) \to S_{i|Q}$ for all $i \in [n], Q \in \mathcal{Q}_{\le k}$ such that if Algorithm 1 is used with a budget $B < z\left(f, R, \{P_r\}_{1\le r\le R}\right)$ then it does **not** return i^*.*

By means of Theorem 4.1, we can infer the following result regarding the sufficient sampling budget B for the three instantiations to output i^* (cf. Sec. D.3 of the appendix for the proof).

Corollary 4.3. *Under Assumptions (A1) and (A2),* $\mathrm{CSX} \in \{\mathrm{CSWS}, \mathrm{CSR}, \mathrm{CSH}\}$ *returns i^* if it is executed with a budget $B \ge z_{\mathrm{CSX}}$, where $z_{\mathrm{CSX}} \coloneqq z\left(f^{\mathrm{CSX}}, R^{\mathrm{CSX}}, \{P_r^{\mathrm{CSX}}\}_{1\le r\le R^{\mathrm{CSX}}}\right)$.*

By substituting the concrete values for P_r, R and f of the corresponding instantiation into Corollary 4.3 and using a rough estimate for the inverse function of the speed of convergence, we see that all of

the resulting sufficient budgets are essentially $\tilde{\mathcal{O}}(n/k)$ (see Table 1) almost[1] matching the dependency on n and k in Theorem 3.1. If we would allow the special case of singleton sets of arms as query sets, i.e., $k = 1$, the sufficient budget for CSH matches the one derived in [25] for its non-combinatorial counterpart in the special case of numerical feedback.

Table 1: Sufficient budget for CSWS, CSR and CSH. Here, π is as in (iii) in Section 3.

z_{CSWS}	$\lceil \frac{n}{k} \rceil \left(\lceil \log_k(n) \rceil + 1\right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \max_{i \in Q \setminus \{i^*\}} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\mid Q} - S_{i\mid Q}}{2} \right) \right\rceil$
z_{CSR}	$\lceil \frac{n}{k} \rceil \left(\left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 \right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \min_{i \in Q \setminus \{i^*\}} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\mid Q} - S_{i\mid Q}}{2} \right) \right\rceil$
z_{CSH}	$\lceil \frac{n}{k} \rceil \left(\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil\right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\mid Q} - S_{\pi(Q)\mid Q}}{2} \right) \right\rceil$

Regarding n and k both lower and upper bounds coincide, but the gap-term in the lower bounds include a min-term over $\mathcal{Q}_{\leq k}$, while the gap-term in the upper bound are coming with a max-term over $\mathcal{Q}_{\leq k}$. The difference between these terms depends on the underlying hardness of the bandit problem in terms of $\bar{\gamma}^{-1}$, i.e., how fast the considered statistics converge to their limit values. Due to the generality of our setting it is difficult to specify this difference more explicitly and it would be worth considering this for special cases, i.e., the numerical bandits or preference-based bandits separately.

Finally, it is worth mentioning that all of the three instantiations of Algorithm 1 have only been studied for the case of single arm pulls, but not for pulls of subsets of arms, where additionally a dependency on the set might be present. Thus, the theoretical guarantees are novel in this regard.

5 Applications to Stochastic Settings

Numerical feedback. In stochastic combinatorial bandits [16], each arm-query set pair (i, Q) is associated with a probability distribution $\nu_{i|Q}$ and querying Q for the t-th time results in the feedback $o_{i|Q}(t) \sim \nu_{i|Q}$, usually referred to as a reward (i.e., $D = \mathbb{R}$). The sequence of rewards $\{o_{i|Q}(t)\}_t$ is supposed to be independent and the statistic s is the empirical mean such that (A1) holds by the law of large numbers with $S_{i|Q} = \mathbb{E}_{X \sim \nu_{i|Q}}[X]$. If the $\nu_{i|Q}$ are sub-Gaussian, an anytime confidence bound by [24] based on the law of iterated logarithm ensures $|s_{i|Q}(t) - S_{i|Q}| \leq c_\delta(t)$ for all $t \in \mathbb{N}$ with probability at least $1 - \delta$ for some appropriate function $c_\delta(t) \in \mathcal{O}(\sqrt{t \ln(\ln(t)/\delta)})$. This implies the following result, the proof of which is deferred to Section E.

Corollary 5.1. *Let f, R and $\{P_r\}_{r \in [R]}$ be as in Theorem 4.1 and suppose the reward distributions $\nu_{i|Q}$ to be σ-sub-Gaussian and such that their means $S_{i|Q}$ satisfy (A2). There is a function $C(\delta, \varepsilon, k, R, \sigma)$ in $\mathcal{O}\left(\sigma^2 \varepsilon^{-2} \ln\left(kR/\delta \ln\left(kR\sigma/\varepsilon\delta\right)\right)\right)$ such that if i^* is the optimal arm for $(S_{i|Q})_{Q \in \mathcal{Q}_{\leq k}, i \in Q}$ and $\sup_{Q \in \mathcal{Q}_{\leq k}(i^*)} \Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$, then Algorithm 1 used with a budget B larger than $C(\delta, \varepsilon, k, R, \sigma) \cdot R \max_{r \in [R]} P_r$ returns i^* with probability at least $1 - \delta$.*

Other statistics for numerical feedback. A rich class of statistics can be obtained by applying a linear functional $U(F) = \int r(x) \mathrm{d}F(x)$, where F is a cumulative distribution function (CDF) and $r : \mathbb{R} \to \mathbb{R}$ some measurable function [49], on the empirical CDF, i.e., $\tilde{s}(O, x) = |O|^{-1} \sum_{o \in O} \mathbb{1}\{x \leq o\}$, for any $x \in \mathbb{R}$ and any multiset of (reward) observations O. This leads to the statistics

$$s_{i|Q}(t) = U(\tilde{s}(o_{i|Q}(1), \ldots, o_{i|Q}(t), \cdot)) = \sum_{s=1}^{t} \frac{r(o_{i|Q}(s))}{t},$$

which converge to $S_{i|Q} = \mathbb{E}_{X \sim \nu_{i|Q}}[r(X)]$ by the law of large numbers, provided these expected values exist. For this class of statistics we can show a quite similar result to Corollary 5.1 by generalizing the findings in [24] (see Section E). However, the result is in fact more general than Corollary 5.1 as for r being the identity function we can recover the empirical mean.

[1] Here, $\tilde{\mathcal{O}}$ hides logarithmic factors.

Preference feedback. In the preference-based bandits, we observe when querying Q for the t-th time a categorical random variable with values in Q, i.e., $o_{i|Q}(t) \sim \mathrm{Cat}_Q(\mathbf{p}_Q)$ for some underlying unknown parameter $\mathbf{p}_Q = (p_{i|Q})_{i \in Q}$. Let $w_{i|Q}(t) \coloneqq \sum_{s \leq t} \mathbf{1}\{o_{i|Q}(t) = i\}$ be the number of times arm i has won in the query set Q until the t-th pull of Q. We consider as the relevant statistics $s_{i|Q}(t) = \frac{w_{i|Q}(t)}{t}$, which converge to $p_{i|Q} =: S_{i|Q}$ by the law of large numbers. The Dvoretzky-Kiefer-Wolfowitz inequality [19] ensures a concentration inequality on $\sup_{i \in Q} |s_{i|Q}(t) - S_{i|Q}|$, which can be used to deduce the following result (cf. Sec. E for the proof).

Corollary 5.2. *Let f,R and $\{P_r\}_{r \in [R]}$ be as in Theorem 4.1 and suppose preference-based winner feedback with parameter $(p_{i|Q})_{Q \in \mathcal{Q}_{\leq k}, i \in Q}$, which satisfies (A2). There is a function $C(\delta, \varepsilon, k, R) \in \mathcal{O}\left(\varepsilon^{-2} \ln\left(R/\delta\varepsilon^4\right)\right)$ with the following property: If i^* is the optimal arm and $\sup_{Q \in \mathcal{Q}_{\leq k}(i^*)} \Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$, then Algorithm 1 used with a budget B larger than $C(\delta, \varepsilon, k, R) \cdot R \max_{r \in [R]} P_r$ returns i^* with probability at least $1 - \delta$.*

By substituting the concrete values for P_r, R and f of the corresponding instantiation of Algorithm 1 into the bound on the budget in Corollary 5.2 (compare to Table 1), we see that each of the three resulting bounds almost matches the optimal sample complexity bounds for identifying the (generalized) Condorcet Winner under fixed confidence in preference-based bandits [8, 21] indicating near optimality of the algorithms in stochastic settings. However, since no stochastic counterpart of our combinatorial setting for the numerical case exists, it would be interesting to investigate whether the analogous implication by means of Corollary 5.1 for the three algorithms is nearly optimal as well. We leave this to future work, as it is beyond the scope of our work.

6 Experimental Section

In this section we present an experimental study for our proposed algorithms on an algorithm selection problem. Further experiments, also on synthetic data and with other statistics are provided in the supplementary material in Section G.

Setting. In the following, we consider an algorithm selection problem, where the goal is to select the most efficient algorithm for solving an instance of a satisfiability (SAT) problem. For this, we randomly chose $n = 20$ parameterizations of the SAPS solver [23] which represent our candidate algorithms and correspond to the arms in our terminology. Our possible problem instances are sampled from the first 5000 problem instances from the sat_SWGCP folder of the AClib[2]. We compare CSWS, CSR, CSH and RoundRobin on this problem with the Successive Halving (SH) algorithm [25]. To the best of our knowledge, there are no algorithms available as baselines, which are designed for the pure exploration problem with finite budget and subsets of arms as the actions, e.g., [4] investigates a regret minimization problem, while [21] is dealing with a stochastic pure exploration setting with fixed-confidence. However, Successive Halving serves as a baseline, which we included as a representative for the algorithms dealing with a pure exploration problem with finite budget and single arms as the actions. In each learning round, we randomly draw a problem instance from the 5000 problem instances without replacement and then start a parallel solution process with the SAPS parameterizations chosen by the corresponding learning algorithm (only one parameterization for the case of SH), where the process is stopped as soon as the first algorithm has solved the current instance. In particular, one obtains only for the "finisher" SAPS parameterization an explicit numerical value (its runtime) among the chosen set of SAPS parameterizations, as the others are right-censored. Since our proposed algorithms are designed for the case, in which feedback for all arms in the pulled query set is observed, while SH is designed for the case in which only a single arm is queried resulting in a single feedback, we enlarge the available budget for SH to $k \cdot B$ for a fairer comparison.

Instantiation of CSE. Although we could consider the negative runtimes of the parameterizations as rewards (i.e., runtimes correspond to losses) and use a statistic suitable for numerical feedback for the combinatorial successive elimination (CSE) approaches, there might be a major disadvantage due to the censored runtimes. Indeed, in order to apply a statistic suitable for numerical feedback, some sensible imputation technique is required to deal with the censored observations, which in turn could introduce a severe bias. However, thanks to the generality of our framework, we can simply

[2] http://www.aclib.net

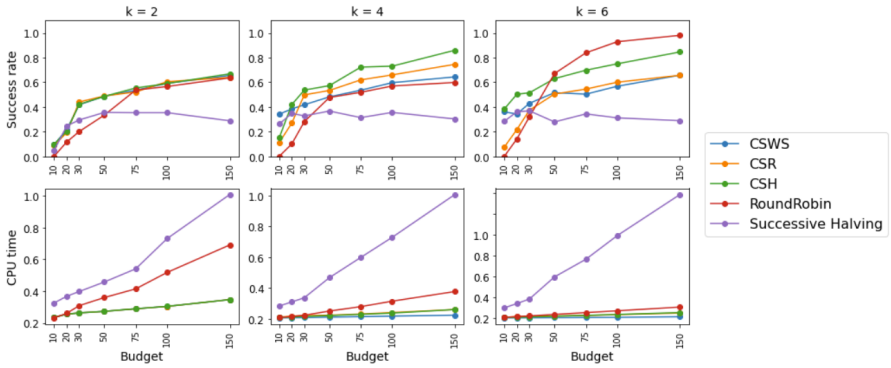


Figure 1: Success rates and runtimes for different subset sizes k and budgets B.

interpret the observed feedback as a preference in the sense that the "finisher" SAPS parameterization is preferred over the others in the chosen set of parameterizations. In this way, using a statistic based on preference-based feedback defuses the bias issue. Quite naturally, we use the relative frequency statistic for preference feedback as specified in Section 5.

Analysis. As the best arm (SAPS parameterization) we use the one having the smallest empirical runtime over all problem instances such that RoundRobin and SH will tendentially return this arm if the budget is sufficiently large. The resulting success rates for our proposed algorithms and SH of identifying the best arm are illustrated in the top panel of Figure 1. One can see, that the algorithms which follow our CSE strategy significantly outperform SH if the budget is sufficiently large. In addition, CSWS, CSR and CSH identify the best arm more often than RoundRobin if the subset sizes k are small, which is a realistic situation in practice. Moreover, the bottom panel in Figure 1 shows the overall runtimes of the algorithms revealing that SH takes much longer than CSWS, CSR and CSH and as expected the difference in the runtimes gets larger with the subset sizes k. Quite interestingly, even RoundRobin needs a longer runtime than the CSE approaches, although it queries the same number of subsets and also stops the respective run as soon as the "finisher" SAPS parameterization is clear. Thus, the differences in the runtimes of RoundRobin and the CSE approaches are only due to the fact that the latter discard the slowest SAPS parameterizations quickly and do not run them again, while RoundRobin uses throughout all subsets the same amount of time, even if they contain only bad performing parameterizations. In other words, the differences are due to the sophisticated strategies of the CSE approaches.

7 Future Work

For future work, it would be interesting to investigate whether switching the elimination strategy during the learning process leads to any performance improvements both theoretically and empirically. A similar question could be asked regarding the considered statistic for numerical feedback variants. Further, the goal of identifying the best set of arms in our scenario would also be interesting. However, in the case where the observations depend on the chosen set of arms, it is far from obvious how to define a suitable optimality term (cf. Sec. 6.3.2 in [8]). Finally, a more extensive experimental study would definitely be a relevant future direction of research, especially for hyperparameter optimization problems with possible parallelization options such as in [33] or for more general algorithm configuration problems [44].

Acknowledgments and Disclosure of Funding

This work was partially supported by the German Research Foundation (DFG) within the project "Online Preference Learning with Bandit Algorithms" (project no. 317046553) and by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

References

[1] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, Alan Malek, and Michal Valko. Best of both worlds: Stochastic & adversarial best-arm identification. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 918–949, 2018.

[2] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved Algorithms for Linear Stochastic Bandits. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 2312–2320, 2011.

[3] Naoki Abe and Philip M Long. Associative reinforcement learning using linear probabilistic concepts. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3–11, 1999.

[4] Arpit Agarwal, Nicholas Johnson, and Shivani Agarwal. Choice bandits. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Jean Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 41–53, 2010.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[7] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331. IEEE, 1995.

[8] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22:1–108, 2021.

[9] Viktor Bengs and Eyke Hüllermeier. Preselection bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 778–787, 2020.

[10] Brian Brost, Yevgeny Seldin, Ingemar J. Cox, and Christina Lioma. Multi-dueling bandits and their application to online ranker evaluation. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2161–2166, 2016.

[11] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(5), 2011.

[12] Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 42.1–42.23, 2012.

[13] Asaf Cassel, Shie Mannor, and Assaf Zeevi. A general approach to multi-armed bandits under risk criteria. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 1295–1306, 2018.

[14] Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[15] Wei Chen, Yihan Du, Longbo Huang, and Haoyu Zhao. Combinatorial pure exploration for dueling bandit. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1531–1541. PMLR, 2020.

[16] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 151–159, 2013.

[17] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 208–214, 2011.

[18] Yihan Du, Yuko Kuroki, and Wei Chen. Combinatorial pure exploration with full-bandit or partial linear feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 7262–7270, 2021.

[19] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27(3):642 – 669, 1956.

[20] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 998–1027, 2016.

[21] Björn Haddenhorst, Viktor Bengs, and Eyke Hüllermeier. Identification of the generalized Condorcet winner in multi-dueling bandits. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[22] Joel Huber and Christopher Puto. Market boundaries and product choice: Illustrating attraction and substitution effects. *Journal of Consumer Research*, 10(1):31–44, 1983.

[23] Frank Hutter, Dave AD Tompkins, and Holger H Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *International Conference on Principles and Practice of Constraint Programming*, pages 233–248. Springer, 2002.

[24] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'UCB : An optimal exploration algorithm for multi-armed bandits. In *Proceedings of Annual Conference on Learning Theory (COLT)*, volume 35, pages 423–439, 2014.

[25] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 240–248, 2016.

[26] Marc Jourdan, Mojmír Mutnỳ, Johannes Kirschner, and Andreas Krause. Efficient pure exploration for combinatorial bandits with semi-bandit feedback. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, pages 805–849, 2021.

[27] Johannes Kirschner and Andreas Krause. Bias-robust Bayesian optimization via dueling bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5595–5605, 2021.

[28] Anna Korba. *Learning from ranking data: theory and methods*. PhD thesis, Université Paris-Saclay (ComUE), 2018.

[29] Michael R. Kosorok. *Introduction to Empirical Processes and Semiparametric Inference*. Springer, New York, USA, 2008.

[30] Yuko Kuroki, Liyuan Xu, Atsushi Miyauchi, Junya Honda, and Masashi Sugiyama. Polynomial-time algorithms for multiple-arm identification with full-bandit feedback. *Neural Computation*, 32(9):1733–1773, 2020.

[31] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[32] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[33] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems (MLSys)*, 2:230–246, 2020.

[34] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[35] Pascal Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, 18(3):1269 – 1283, 1990.

[36] Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top-k rank aggregation from noisy comparisons. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2488–2497, 2017.

[37] Felix Mohr, Viktor Bengs, and Eyke Hüllermeier. Single player Monte-Carlo tree search based on the Plackett-Luce Model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 12373–12381, 2021.

[38] Rémi Munos. From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends® in Machine Learning*, 7(1):1–129, 2014.

[39] Wenbo Ren, Jia Liu, and Ness Shroff. On sample complexity upper and lower bounds for exact ranking from noisy comparisons. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 10014–10024, 2019.

[40] Wenbo Ren, Jia Liu, and Ness Shroff. The sample complexity of best-k items selection from pairwise comparisons. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8051–8072, 2020.

[41] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[42] Aadirupa Saha and Aditya Gopalan. Battle of bandits. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 805–814, 2018.

[43] Aadirupa Saha and Aditya Gopalan. From PAC to instance-optimal sample complexity in the Plackett-Luce model. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 8367–8376, 2020.

[44] 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, 2022.

[45] Cong Shen. Universal best arm identification. *IEEE Transactions on Signal Processing*, 67(17):4464–4478, 2019.

[46] Max Simchowitz, Kevin Jamieson, and Benjamin Recht. Best-of-k-bandits. In *Proceedings of Annual Conference on Learning Theory (COLT)*, pages 1440–1489, 2016.

[47] Itamar Simonson. Choice based on reasons: The case of attraction and compromise effects. *Journal of Consumer Research*, 16(2):158–174, 1989.

[48] Amos Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79(4):281, 1972.

[49] Larry Wasserman. *All of Statistics: A concise Course in Statistical Inference*. Springer Science & Business Media, 2013.

[50] Shuo Yang, Tongzheng Ren, Inderjit S Dhillon, and Sujay Sanghavi. Combinatorial bandits without total order for arms. *arXiv preprint arXiv:2103.02741*, 2021.

[51] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 222–227, 1977.

[52] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

6

AC-Band: A Combinatorial Bandit-Based Approach to Algorithm Configuration

Author Contribution Statement

The idea of this work was developed in a common brainstorming with Viktor Bengs, Kevin Tierney and the author. The theoretical works come from the author with support and revisions by Viktor Bengs and Björn Haddenhorst. The implementation of the experiments was done by Elias Schede and the paper was written mainly by the author with subsequently revisions by all authors.

Supplementary Material

An appendix to the paper is provided in Appendix B. The code of the official implementation is provided at https://github.com/DOTBielefeld/ACBand.

AC-Band: A Combinatorial Bandit-Based Approach to Algorithm Configuration

Jasmin Brandt[1], Elias Schede[2], Björn Haddenhorst[1], Viktor Bengs[3,4], Eyke Hüllermeier[3,4], Kevin Tierney[2]

[1]Department of Computer Science, Paderborn University, Germany
[2]Decision and Operation Technologies Group, Bielefeld University, Germany
[3]Institute of Informatics, LMU Munich, Germany
[4]Munich Center for Machine Learning (MCML), Germany
{jasmin.brandt, bjoernha}@upb.de, {elias.schede, kevin.tierney}@uni-bielefeld.de, {viktor.bengs, eyke}@ifi.lmu.de

Abstract

We study the algorithm configuration (AC) problem, in which one seeks to find an optimal parameter configuration of a given target algorithm in an automated way. Recently, there has been significant progress in designing AC approaches that satisfy strong theoretical guarantees. However, a significant gap still remains between the practical performance of these approaches and state-of-the-art heuristic methods. To this end, we introduce AC-Band, a general approach for the AC problem based on multi-armed bandits that provides theoretical guarantees while exhibiting strong practical performance. We show that AC-Band requires significantly less computation time than other AC approaches providing theoretical guarantees while still yielding high-quality configurations.

Introduction

Algorithm configuration (AC) is concerned with the task of automated search for a high-quality configuration (e.g., in terms of solution quality or runtime) of a given parameterized target algorithm. Parameterized algorithms are found in many different applications, including, for example, optimization (e.g., satisfiability (SAT) (Audemard and Simon 2018) or mixed-integer programming (MIP) solvers (IBM 2020)), simulation, and machine learning. Finding good configurations is a significant challenge for algorithm designers, as well as for users of such algorithms who may want to adjust the algorithm to perform well on data specific to their use case. Finding good configurations through trial and error by hand is a daunting task, hence automated AC methods have been developed on the basis of heuristics, such as ParamILS (Hutter, Hoos, and Stützle 2007; Hutter et al. 2009), GGA (Ansótegui, Sellmann, and Tierney 2009; Ansótegui et al. 2015), irace (Birattari et al. 2002; López-Ibáñez et al. 2016) or SMAC (Hutter, Hoos, and Leyton-Brown 2013, 2011).

While heuristic configurators have had great success at finding good configurations on a wide variety of target algorithms, they do not come with theoretical guarantees. To this end, the pioneering work of Kleinberg, Leyton-Brown, and Lucier (2017) proposes the first algorithm configurator with provable, theoretical guarantees on the (near-)optimality of the configuration returned. Further improvements and adjustments to those guarantees have followed (Weisz, György, and Szepesvári 2018, 2019; Kleinberg et al. 2019; Weisz et al. 2020). All of these works essentially consider the runtime as the performance objective and provide, in particular, an upper bound on the total runtime of the respective algorithm configurator that is (nearly) optimal in a worst-case sense.

Despite their appealing theoretical properties and the steady progress on their empirical performance, these approaches still cannot compete with heuristic approaches in terms of practical performance. The main issue of these theoretical approaches is that they are conservative in the process of discarding configurations from the pool of potential candidates, as pointed out in recent work (Weisz et al. 2020). This is indeed a key characteristic difference compared to the heuristic approaches, which discard configurations quite quickly after being used only on a couple of problem instances. From a practical point of view, this makes sense, as the risk of discarding all good configurations is generally lower than wasting lots of time looking at bad configurations.

In an attempt to further bridge the gap between heuristic configurators and theoretical approaches, we propose AC-Band, a general algorithm configurator inspired by the popular Hyperband (Li et al. 2016) approach based on multi-armed bandits (Lattimore and Szepesvári 2020). Hyperband is an algorithm for the hyperparameter optimization problem (HPO) (Yang and Shami 2020; Bischl et al. 2021b), which is essentially a subproblem of the general AC problem focusing on configuring solution quality of algorithms, with a particular focus on machine learning methods. While using HPO approaches for AC looks attractive at first, it is rather uncommon in practice due to the subtle differences between the two problems. These differences include potentially using runtime as a configuration metric and the existence of multiple *problem instances*, which are different settings or scenarios that an optimization method must solve.

Our suggested approach reconciles the basic idea behind the mechanism of Hyperband with the key characteristics of the AC problem and incorporates the advantageous property of discarding configurations quickly. This is achieved by first replacing the underlying bandit algorithm of Hyperband, Successive Halving (SH) (Karnin, Koren, and Somekh 2013), by a more general variant, Combinatorial Successive Elimination (CSE) (Brandt et al. 2022), and then carefully adapting the parameters of the iterative CSE calls over time. In contrast to SH, as well as to all other multi-armed bandit algorithms

for pure exploration with finite budget, CSE allows (i) to steer the aggressiveness of discarding arms (configurations in our terminology) from the set of consideration, (ii) to pull more than one arm simultaneously (run multiple configurations in parallel), and (iii) to work with observations either of quantitative or qualitative nature. As mentioned above, the first property seems to be of major importance in AC problems, but the other two properties will turn out to be particularly helpful as well. Indeed, (ii) obviously allows parallelization of the search process, while the generality regarding the nature of the observations in (iii) transfers quite naturally to the suggested method.

The interplay of the second and third properties allows us to instantiate AC-Band to obtain appealing practical performance compared to existing theoretical approaches regarding the total runtime. On the theoretical side, we derive (under mild assumptions on the underlying AC problem) that AC-Band is guaranteed to return a nearly optimal configuration with high probability if used on sufficiently many problem instances. Our theoretical result is quite general in the sense that the notion of optimality is not restricted to the runtime of the configurations, but also encompasses other target metrics such as solution quality or memory usage. The technical appendix can be found on arXiv[1].

Related Work

Theoretical advances in AC. The field of AC has grown to include many different methods and settings; we refer to Schede et al. (2022) for a full overview, especially with regard to the heuristic methods previously mentioned. Inspired by Kleinberg, Leyton-Brown, and Lucier (2017), who introduced *Structured Procrastination* together with a non-trivial worst-case runtime bound, more and more algorithms with even better theoretical guarantees with respect to the runtime have been proposed. *LeapsAndBounds* (Weisz, György, and Szepesvári 2018) tries to guess an upper bound on the optimal runtime by doubling the last failed guess, whereas *Structured Procrastination with confidence* (Kleinberg et al. 2019) works by delaying solving hard problem instances until later. Rather, it first runs the configurations with the smallest lower confidence bound of its mean runtime on instances that are easy to solve. Another recent method, *CapsAndRuns* (Weisz, György, and Szepesvári 2019), first estimates a timeout for each configuration and afterwards performs a Bernstein race over the configurations. As a follow up method, *Impatient-CapsAndRuns* (Weisz et al. 2020) uses a more aggressive elimination strategy by filtering unlikely optimal configurations in a preprocessing. Further theoretical progress has been made regarding the analysis of the estimation error in AC settings (Balcan et al. 2019; Balcan, Sandholm, and Vitercik 2020), the distribution of the computational budget (Liu et al. 2020) and the understanding of heuristic methods (Hall, Oliveto, and Sudholt 2019, 2020).

HPO. As a subset of AC, HPO involves setting the *hyperparameters* of algorithms, in particular machine learning approaches. The term hyperparameter differentiates parameters that change the behavior of the algorithm being configured from parameters that are induced or learned from data and are thus not set by a user. In contrast, AC focuses on configuring algorithms that solve instances of a dataset independently. We refer to Bischl et al. (2021a) for a full overview of HPO.

Bandit methods for AC. Classically, methods for multi-armed bandits (MAB) (Lai and Robbins 1985; Bubeck and Cesa-Bianchi 2012; Lattimore and Szepesvári 2020) are designed to find a good balance between exploration-exploitation of specific choice alternatives (e.g., configurations or hyperparameters). The pure exploration setting, however, has attracted much research interest as well (Bubeck, Munos, and Stoltz 2009; Karnin, Koren, and Somekh 2013; Aziz et al. 2018), especially for HPO (Jamieson and Talwalkar 2015; Li et al. 2016). However, up to now bandit algorithms making single choice alternative decisions have been leveraged, although the parallel execution of configurations (or hyperparameters) in the AC (or HPO) setting seems to be predetermined for the combinatorial bandit variant (Cesa-Bianchi and Lugosi 2012; Chen, Wang, and Yuan 2013; Jourdan et al. 2021). In light of this, the recently proposed CSE algorithm (Brandt et al. 2022) seems promising as a generalization of the popular SH approach.

Problem Formulation

We adopt the formulation of the problem as by Schede et al. (2022). Let $\mathcal{I}$ be the space of problem instances and $\mathcal{P}$ an unknown probability distribution over $\mathcal{I}$. Suppose $\mathcal{A}$ is an algorithm that can be run on any problem instance $i \in \mathcal{I}$, and has different parameters p_j coming from a known domain Θ_j for each $j \in \{1, \dots, m\}$. We call $\mathcal{A}$ the *target algorithm* and the Cartesian product of its parameter domains $\Theta = \Theta_1 \times \cdots \times \Theta_m$ the *configuration* or *search space* consisting of all feasible parameter *configurations*. For a configuration $\theta \in \Theta$, we denote by $\mathcal{A}_\theta$ an instantiation of the target algorithm $\mathcal{A}$ with configuration θ. Running the target algorithm $\mathcal{A}$ with configuration θ on a specific problem instance $i \in \mathcal{I}$ results in costs specified by an unknown, and possibly stochastic, function $c : \mathcal{I} \times \Theta \to \mathbb{R}$, i.e., $c(i, \theta)$ represents the costs of using $\mathcal{A}_\theta$ for problem instance i. Here, the costs can correspond to the runtime of $\mathcal{A}_\theta$ for i, but also to other relevant target metrics such as the solution quality or the memory usage.

Algorithm Configurator. The goal in algorithm configuration is, roughly speaking, to find a configuration that is optimal, or at least nearly optimal, with respect to the costs in a certain sense, which we specify below. The search for such configurations is achieved by designing an algorithm configurator $\mathcal{AC}$ that (i) selects specific configurations in Θ and (ii) runs them on some (perhaps randomly) chosen problem instances in $\mathcal{I}$. To this end, the algorithm configurator uses a statistic $s : \bigcup_{t \in \mathbb{N}} \mathbb{R}^t \to \mathbb{R}$ that maps the observed cost of a configuration θ used for a set of problem instances $i_1, \dots, i_t$ to a representative numerical value $s(c(i_1, \theta), \dots, c(i_t, \theta))$, that guides the search behavior of $\mathcal{AC}$. For example, s could be the arithmetic mean, i.e., $s(c(i_1, \theta), \dots, c(i_t, \theta)) = t^{-1} \sum_{s=1}^{t} c(i_s, \theta)$.

In this work, we are interested in algorithm configurators that can run several different configurations, up to a certain

[1] https://arxiv.org/abs/2212.00333

size k, in parallel on a selected problem instance. The algorithm configurator is given a fixed computational budget B, which represents the maximum number of such parallel runs and is set externally. For this purpose, let $\Theta_{[2,k]} = \{\tilde{\Theta} \subset \Theta \mid 2 \leq |\tilde{\Theta}| \leq k\}$ be the set of all possible subsets of parameter configurations that have at least size 2 and at most size k. Furthermore, let $\Theta_{[2,k]}(\theta) = \{\tilde{\Theta} \in \Theta_{[2,k]} \mid \theta \in \tilde{\Theta}\}$ be the set of all possible subsets of parameter configurations containing the configuration $\theta \in \Theta$. Note, that the observed cost $c(i, \theta)$ for running θ along with other configurations on an instance $i \in \mathcal{I}$ could depend on the respectively chosen configuration set $\tilde{\Theta} \in \Theta_{[2,k]}(\theta)$. For example, the algorithm configurator could stop the parallel solution process as soon as one of the configurations provides a solution, and set a default cost (penalty) for the configurations that did not complete. Hence, we write $c_{\tilde{\Theta}}$ for the cost function in the following to take this contingency into account. Finally, we introduce $s_{\theta|\tilde{\Theta}}(t) = s(c_{\tilde{\Theta}}(i_1, \theta), \ldots, c_{\tilde{\Theta}}(i_t, \theta))$ which is the statistic of θ after running it in parallel with the configurations in $\tilde{\Theta} \backslash \{\theta\}$ on t problem instances $i_1, \ldots, i_t$.

ϵ-optimal Configurations. Since the observed costs are potentially dependent on the chosen set of configurations to evaluate, we first introduce the following assumption on the limit behavior of the statistics.

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

In words, for each possible set of configurations the (possibly dependent) statistic of each configuration involved converges to some limit value if run on infinitely many problem instances. Recalling the example of s being the arithmetic mean, this is arguably a mild assumption and implicitly assumed by most approaches for AC problems, due to the considered i.i.d. setting. Since our assumption is more general, it would also allow considering non-stationary scenarios of AC.

The natural notion of an optimal configuration θ^* is a configuration that has the largest (configuration-set dependent) limit value over all configurations. Indeed, if we would replace $\Theta_{[2,k]}$ by the singleton sets of all configurations in Θ, this would correspond to the commonly used definition of the optimal configuration (see (Schede et al. 2022)), as the limit value would be then $\mathbb{E}[c(i, \theta)]$[2]. However, in our case this notion of optimality has two decisive drawbacks, as first of all such a θ^* may not exist. Moreover, even if it exists, the search for it might be hopeless as the configuration space is infinite (or very large). The latter issue arises in the "usual" AC problem scenario as well, and is resolved by relaxing the objective to finding a "good enough" configuration. We adapt this notion of near optimality by resorting to the definition of an ϵ-best arm from the preference-based bandit literature (Bengs et al. 2021). For some fixed relaxation parameter $\varepsilon > 0$, we call a configuration θ an *ϵ-best configuration* iff

$$\forall \tilde{\Theta} \in \Theta_{[2,k]}(\theta) \ : \ S_{\theta|\tilde{\Theta}} \geq S_{(1)|\tilde{\Theta}} - \epsilon, \tag{1}$$

where $S_{(1)|\tilde{\Theta}} \geq \ldots \geq S_{(|\tilde{\Theta}|)|\tilde{\Theta}}$ is the ordering of $\{S_{\theta|\tilde{\Theta}}\}_{\theta \in \tilde{\Theta}}$.

[2]The expectation is w.r.t. $\mathcal{P}$ and the possible randomness due to $\mathcal{A}_\theta$ and/or the cost generation.

Although we have relaxed the notion of optimality, finding ϵ-best configurations is still often like searching for a needle in a haystack. Hence, we need to ensure that there is a sufficiently high probability that an ϵ-best configuration is included in a large random sample set of Θ:

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

Note this assumption is once again inspired by the bandit literature dealing with infinitely many arms (de Heide et al. 2021). By fixing the probability for the non-occurrence of an ϵ-best configuration to some $\delta \in (0, 1)$, Assumption $(A2)$ ensures that a uniformly at random sampled set of configurations with size $N_{\alpha,\delta} = \lceil \log_{1-\alpha}(\delta) \rceil$ contains at least one ϵ-best configuration with probability at least $1 - \delta$.

Of course, an efficient algorithm configurator $\mathcal{AC}$ that aims to find an ϵ-best configuration θ^* cannot verify the condition $S_{\theta^*|\tilde{\Theta}} \geq S_{(1)|\tilde{\Theta}} - \epsilon$ for every possible query set $\tilde{\Theta} \in \Theta_{[2,k]}(\theta^*)$, in particular when the number of configurations, and thus the cardinality of $\Theta_{[2,k]}(\theta^*)$, is infinite. Instead, $\mathcal{AC}$ can only guarantee the above condition for a finite number of query sets and therefore it will always find a proxy for an ϵ-best configuration that does not have to be a true ϵ-best configuration. To guarantee that $\mathcal{AC}$ finds a true ϵ-best configuration with high probability, we introduce the following assumption.

$$(A3) : \forall M \in \mathbb{N}, \forall \theta \in \Theta :$$
$$\mathbb{P}\Big((\forall i \in [M] : S_{\theta|\tilde{\Theta}_i} \geq S_{(1)|\tilde{\Theta}_i} - \epsilon)$$
$$\Rightarrow \ (\forall \tilde{\Theta} \in \Theta_{[2,k]}(\theta) : S_{\theta|\tilde{\Theta}} \geq S_{(1)|\tilde{\Theta}} - \epsilon) \Big) \geq 1 - \psi(M),$$

where $\tilde{\Theta}_1, \ldots, \tilde{\Theta}_M \sim \text{Uniform}(\Theta_{[2,k]}(\theta))$ and $\psi : \mathbb{N} \to [0, 1]$ is a strictly monotone decreasing function. In words, the probability that a configuration θ is a global ϵ-best configuration increases with the number of (randomly chosen) configuration sets on which it is a local ϵ-best configuration, i.e., the characteristic condition in (1) is fulfilled.

Note that $(A3)$ is a high-level assumption on the difficulty of the underlying AC problem. The "easier" the problem, the steeper the form of ψ and vice versa. As we do not impose further assumptions on ψ other than monotonicity, our theoretical results below are valid for a variety of AC problems.

AC-Band Algorithm

The AC-Band algorithm consists of iterative calls to CSE (Brandt et al. 2022) that allow it to successively reduce the size of a candidate set of configurations. We first describe how CSE works, then elaborate on its use in the AC-Band approach.

Combinatorial Successive Elimination. CSE (Algorithm 1) is a combinatorial bandit algorithm that, given a finite set of arms (configurations), finds an optimal arm[3] defined similarly to (1) for $\epsilon = 0$ using only a limited amount of feedback observations (budget). To this end, CSE proceeds on a round-by-round basis, in each of which (i) the non-eliminated arms

[3]Its existence is simply assumed.

Algorithm 1: Combinatorial Successive Elimination (CSE)

Input: set of configurations $\tilde{\Theta}$ with $|\tilde{\Theta}| = n$, subset size $k \leq n$, budget B, $\rho \in (0, \log_2(k)]$, problem instances I with $|I| = B$ which can be partitioned into $R^{\rho,k,n} = \min_x\{g^{\circ x}(n) \leq k\} + \min_x\{f_\rho^{\circ x}(k) \leq 1\}$ problem instances I_r with $|I_r| = P_r^{\rho,k,n} \cdot b_r$ where $\left\{P_r^{\rho,k,n}\right\}_r = \left\{\lfloor n/k \left(f_\rho(k)/k\right)^{r-1} \rfloor\right\}_r$, and $g(x) = f_\rho(k) \cdot \lfloor x/k \rfloor + x \mod k$
Initialization: $\tilde{\Theta}_1 \leftarrow \tilde{\Theta}$, $r \leftarrow 1$

1: **while** $|\tilde{\Theta}_r| \geq k$ **do**
2: $b_r \leftarrow \left\lfloor B/(P_r^{\rho,k,n} \cdot R^{\rho,k,n})\right\rfloor$, $J \leftarrow P_r^{\rho,k,n}$
3: $\tilde{\Theta}_{r,1}, \tilde{\Theta}_{r,2}, \ldots, \tilde{\Theta}_{r,J} \leftarrow \mathrm{Partition}(\tilde{\Theta}_r, k)$
4: **if** $|\tilde{\Theta}_{r,J}| < k$ **then**
5: $\mathcal{R} \leftarrow \tilde{\Theta}_{r,J}$, $J \leftarrow J - 1$
6: **else**
7: $\mathcal{R} \leftarrow \emptyset$
8: **end if**
9: $I_{r,1}, \ldots, I_{r,J} \leftarrow \mathrm{Partition}(I_r, b_r)$
10: $\tilde{\Theta}_{r+1} \leftarrow \mathcal{R}$
11: **for** $j \in [J]$ **do**
12: $\mathcal{R} \leftarrow \mathrm{ArmElimination}(\tilde{\Theta}_{r,j}, b_r, f_\rho(|\tilde{\Theta}_{r,j}|), I_{r,j})$
13: $\tilde{\Theta}_{r+1} \leftarrow \tilde{\Theta}_{r+1} \cup \mathcal{R}$
14: **end for**
15: $r \leftarrow r + 1$
16: **end while**
17: $\tilde{\Theta}_{r+1} \leftarrow \emptyset$
18: **while** $|\tilde{\Theta}_r| > 1$ **do**
19: $\tilde{\Theta}_{r+1} \leftarrow \mathrm{ArmElimination}(\tilde{\Theta}_r, b_r, f_\rho(|\tilde{\Theta}_r|), I_r)$
20: $r \leftarrow r + 1$
21: **end while**

Output: The remaining item in $\tilde{\Theta}_r$

Algorithm 2: ArmElimination$(\tilde{\Theta}', b, l, I')$

1: Use $\tilde{\Theta}'$ for b times on problem instances I'
2: For all $\theta \in \tilde{\Theta}'$, update $s_{\theta|\tilde{\Theta}'}(b)$
3: Choose an ordering $\theta_1, \ldots, \theta_{|\tilde{\Theta}'|}$ of $(s_{\theta|\tilde{\Theta}'}(b))_{\theta \in \tilde{\Theta}'}$
4: **Output:** $\{\theta_1, \ldots, \theta_l\}$

are partitioned into groups of a given size k (line 3) if possible (lines 4–8, 18–19) and (ii) feedback for each group is queried under a round-specific budget, which, once exhausted, leads to the elimination of a certain fraction of arms in each group (Algorithm 2 called in lines 12 and 19). Here, the feedback observed for an arm is mapped to a numerical value using a statistic s that indicates the (group-dependent) utility of the arm, and is used as the basis for elimination in each round. The fraction of eliminated arms is steered via a function $f_\rho : [k] \to [k]$ with $f_\rho(x) = \lfloor x/2^\rho \rfloor$, where the predetermined parameter $\rho \in (0, \log_2(k)]$ controls the aggressiveness of the elimination. A large value of ρ corresponds to a very aggressive elimination, retaining only the arm(s) with (the) highest statistics, while a small ρ eliminates only the arm(s) with the lowest statistics. The overall available budget is first split equally for each round, and then for all partitions in each round (line 2).

In light of the AC problem we are facing, *querying feedback for a group of arms* corresponds to running a subset of configurations in parallel on a problem instance, which results in observations in the form of costs. Moreover, we do not reuse a single problem instance for any parallel run so that the budget is in fact equal to the number of problem instances used. Accordingly, the overall budget of CSE corresponds to the number of problem instances used in total, which are split into disjoint problem instance sets of size b_r, i.e., the round-specific budget (line 9).

Since CSE initially assumes a finite set of arms and a fixed parameter ρ guiding the overall elimination aggressiveness, we face two trade-offs. The first is regarding the interplay between the number of initial arms and the round-wise budget:

(T1): If the initial number of configurations for CSE is small (large), the more (fewer) runs can be carried out on different instances, leading to potentially more reliable (unreliable) statistics, but only on a few (many) subsets of configurations.

The second trade-off arises through the interplay between the round-wise budget and the elimination aggressiveness:

(T2): If the elimination behavior of CSE is aggressive (conservative), then more (fewer) runs can be carried out on different instances, leading to potentially more reliable (unreliable) statistics, but only on a few (many) subsets of configurations.

The challenge now is to reconcile these two trade-offs and, above all, to take into account the specifics of AC problems.

Algorithm 3: AC-Band

Input: target algorithm $\mathcal{A}$, configuration space Θ, problem instance space $\mathcal{I}$, Budget B, subset size k, suboptimality ϵ of "good enough" configuration, proportion of ϵ-best configurations α, failure probability δ, $n_0 > \lceil \ln(\delta)/\ln(1-\alpha) \rceil \in \mathbb{N}$
Initialization: $E \leftarrow \lceil \log_2\left(n_0/n_0 - N_{\alpha,\delta}\right) \rceil$, $q \leftarrow 1 + k-1/E$
$C_1 \leftarrow \log_q(2)$, $C_2 \leftarrow 1 + \log_q\left(n_0 + \frac{4n_0}{n_0 - N_{\alpha,\delta}}\right)$,
$C_3 \leftarrow \lceil \log_q(k) \rceil$

1: sample $\theta_0 \in \Theta$
2: **for** $e \in [E]$ **do**
3: $n_e = \lceil n_0/2^e \rceil + 1$, $\rho_e = \log_2\left(e + k - 1/e\right)$,
4: sample $\theta_{e,1}, \ldots, \theta_{e,n_e-1} \in \Theta$
5: sample $I_e \subset \mathcal{I} \setminus \cup_{e'=1}^{e-1} I_{e'}$ with $|I_e| = B/c_e$, where $c_e = \frac{(C_1 E - (2^E - 1)(2C_1 - C_2 - C_3))2^e}{2^E(-eC_1 + C_2 + C_3)}$
6: $\theta_e = \mathrm{CSE}(\{\theta_{e-1}, \theta_{e,1}, \ldots, \theta_{e,n_e-1}\}, k, |I_e|, \rho_e, I_e)$
7: **end for**

Output: θ_E

AC-Band. The design of AC-Band (Algorithm 3) seeks to find a good balance for both tradeoffs (T1) and (T2) by calling CSE iteratively. Initially, CSE is invoked with larger sets of configurations, and an aggressive elimination strategy is applied. Over time, the size of the candidate sets is successively reduced, and the aggressiveness of the elimination strategy is also gradually decreased. Roughly speaking, the idea is to have high exploration in the beginning, and thus more risk that good configurations are discarded, and become more and more careful towards the end.

More specifically, AC-Band proceeds in epochs $e \in$

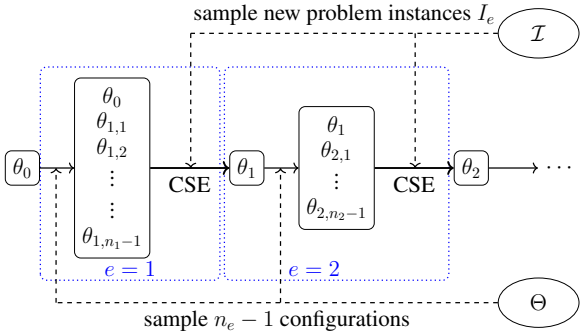


Figure 1: Illustration of AC-Band's solving process.

$\{1, \dots, E\}$, in each of which CSE is called on a specific set of problem instances using a specified degree of elimination aggressiveness ρ_e and a set of configurations of size n_e, with both ρ_e and n_e decreasing w.r.t. e (line 3). At the end of an epoch, i.e., when CSE terminates, a single high quality configuration among the considered set of configurations is returned (line 6). The set of configurations used in an epoch consists of the high quality configuration of the previous epoch, which is sampled randomly for the first epoch (line 1), and $n_e - 1$ randomly sampled configurations, which have not been considered before (line 4).

This epoch-wise procedure of AC-Band is depicted in Figure 1. Although AC-Band is similar in design to Hyperband in that it tries to find a good balance between specific trade-offs by successively invoking a bandit algorithm (CSE vs. SH), AC-Band differs in the way it defines the quality of the search objects[4]. Unlike Hyperband, we do not run configurations individually on problem instances and consider the cost of each configuration on its own, rather configurations are run in parallel and we consider potential interactions. Accordingly, we do not have one global quality value that we can compare for all configurations seen, but several at once.

The overall number of considered problem instances is B (the evaluation budget), which is a parameter that we analyze below. Besides the "usual" parameters of an algorithm configurator, i.e., the target algorithm $\mathcal{A}$, its configuration space Θ and the problem instance space $\mathcal{I}$, AC-Band requires:

- the maximum number of configurations that can be run in parallel k,
- some relevant summary statistic $s : \bigcup_{t \in \mathbb{N}} \mathbb{R}^t \to \mathbb{R}$ for the observed costs (see Problem Formulation),
- the theoretically motivated guarantee parameters $\epsilon > 0$, $\alpha \in (0, 1)$, and $\delta \in (0, 1)$ (see Problem Formulation)
- a reference size n_0 for the set of epoch-wise sampled configurations (must be $\geq \lceil \frac{\ln(\delta)}{\ln(1-\alpha)} \rceil$ for technical reasons).

With these parameters specified, AC-Band determines the overall number of epochs E and the sufficient number of problem instances B/c_e (line 5) for an epoch-wise CSE to return a high quality configuration (see Theoretical Guarantees). Moreover, the overall number of considered configurations is guaranteed to be at least $N_{\alpha,\delta} = \lceil \frac{\ln(\delta)}{\ln(1-\alpha)} \rceil$, which in light of the random sampling of the configurations ensures that at least one ϵ-best configuration is sampled with a probability of at least $1 - \delta$.

[4]Hyperband uses hyperparameters of machine learning models as search objects and AC-Band algorithm configurations.

Theoretical Guarantees

In Appendix C, we prove the following theoretical guarantee for AC-Band regarding the sufficient evaluation budget (or number of problem instances) to find an ϵ-best configuration with high probability w.r.t. $\mathcal{P}$ as well as the randomness invoked by AC-Band. For the proof, we need to extend the theoretical guarantees for CSE to the setting of finding ϵ-best configurations (see Appendix B).

Theorem 0.1. *Let R^e be the number of rounds of CSE in epoch $e \in \{1, \dots, E\}$, let C_1, C_2 and C_3 be as defined in Alg. 3 and further let $\mathbb{A}_r(\theta^*)$[5] be the partition in round r of CSE containing θ^* and*

$$\bar{\gamma}^{-1} = \max_{e \in [E], r \in [R^e]} \big(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(\theta^*)}\big(\max\big\{\frac{\epsilon}{2}, \max_{r \in [R^e]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(\theta^*)|)+1)|\mathbb{A}_r(\theta^*)}}{2}\big\}\big)\big),$$

$$\Delta_{(i)|\tilde{\Theta}} = S_{(1)|\tilde{\Theta}} - S_{(i)|\tilde{\Theta}},$$

$$\bar{\gamma}^{-1}_{\mathbb{A}_r(\theta^*)}(t) = \min_{\theta \in \mathbb{A}_r(\theta^*)} \inf_{t' \in \mathbb{N}} \{|s_{\theta|\mathbb{A}_r(\theta^*)}(t') - S_{\theta|\mathbb{A}_r(\theta^*)}| \leq t\}.$$

Under Assumptions $(A1)$–$(A3)$, Algorithm 3 used with a subset size k, an ϵ-best configuration proportion of α, and a failure probability of δ finds an ϵ-best configuration θ^* with probability at least $\min\{1 - \delta, 1 - \psi((R^E)^{-1})\}$ if

$$B \geq \bar{\gamma}^{-1} \cdot \frac{n_0}{k} \cdot \frac{C_1 E - (2^E - 1)(2C_1 - C_2 - C_3)}{2^E}.$$

Roughly speaking, AC-Band finds a near-optimal configuration with a probability depending on the allowed failure probability δ and the probability that a locally optimal configuration is also a globally optimal one (see Assumption (A3)) if the budget is large enough. The sufficient budget, in turn, is essentially driven by $\bar{\gamma}^{-1}$, which depends on the difficulty of the underlying AC problem by two characteristics: the maximal inverse convergence speed $\bar{\gamma}^{-1}_{\mathbb{A}_r(\theta^*)}$ of the used statistic s, and the maximal (halved) suboptimality gap Δ of the limits of the statistic between the best configuration and the best one that will be discarded from the query set θ^* is contained. The remaining terms of the sufficient budget can be computed explicitly once the theoretical guarantee parameters α and δ, as well as the subset size k, are fixed. The sufficient budget in dependence of the mentioned parameters is discussed and plotted in Appendix C.3.

Note that the theoretical guarantee in Theorem 0.1 is not directly comparable to the ones by the theoretical AC approaches (Kleinberg et al. 2019; Weisz, György, and Szepesvári 2018, 2019; Weisz et al. 2020). This is due to the major differences of our approach and the later ones on how we approach the AC problem. Indeed, we do not restrict ourselves to runtime as the target metric (or the costs), and

[5]$\mathbb{A}_r(\theta) = \emptyset$ if θ is not contained anymore in the set of active configurations in round r.

we also take possible dependencies in the parallel runs into account. As a consequence, the notion of near optimality of a configuration in the other works is tailored more towards runtimes in an absolute sense, i.e., without considering interaction effects, while ours is more general and in particular focusing on such interaction effects. Thus, the theoretical guarantee in Theorem 0.1 in the form of a sufficient evaluation budget to obtain a nearly optimal configuration is sensible, as we do not commit to a specific target metric.

Experiments

We examine AC-Band on several standard datasets for evaluating theoretical approaches for AC. We note that while these datasets refer exclusively to runtimes, AC-Band is applicable to other target metrics (see Section Problem Formulation). In our experiments, we address the following two research questions: **Q1**: Is AC-Band able to find high quality configurations in less CPU time than state-of-the-art AC methods with guarantees? **Q2**: How does AC-Band scale with k?

Datasets. We use one SAT and two MIP datasets that have been used before to assess theoretical works on AC (Weisz et al. 2020). Due to space constraints, we only consider one of the MIP datasets here, while the appendix also discusses the other. The SAT dataset contains precomputed runtimes of configurations of the MiniSat SAT solver (Eén and Sörensson 2003) obtained by solving instances that are generated using CNFuzzDD (Weisz, György, and Szepesvári 2018). The dataset contains runtimes for 948 configurations on 20118 instances. The MIP datasets curated by Weisz et al. (2020) are generated using an Empirical Performance Model (EPM)(Hutter et al. 2014). In particular, the EPM is trained on the CPLEX solver (IBM 2020) separately using instances from a combinatorial auction (Regions200 (Leyton-Brown, Pearson, and Shoham 2000)) and wildlife conservation (RCW (Ahmadizadeh et al. 2010)) dataset. The resulting model is used to predict runtimes for 2000 configurations and 50000 and 35640 new instances, respectively. Since all runtimes are precomputed (a timeout of 900 seconds is used), the evaluation of configuration-instance pairs only required table look-ups in our experiments.

Evaluation. To compare methods, we consider two metrics: (i) the accumulated CPU time needed by a method to find a configuration, and (ii) the percent gap to the best configuration. This second metric measures in percent how much more time the configuration returned by the method needs to solve all instances compared to the best overall configuration for the dataset. Smaller values indicate that the configuration found is closer to the best configuration in terms of runtime and the best configuration has a value of 0. This allows for comparing the quality between methods, as well as to determine how "far" a configuration is from the optimal one. In practical applications of AC, wall-clock time is often a bottleneck, and speeding up the process of finding a suitable configuration is the main focus. For these speedups, practitioners are (usually) willing to sacrifice configuration quality to a certain extent. The other theoretical works use the R^δ metric (note that δ has a different meaning in this work) to evaluate the quality of a returned configuration. This metric is a variation of the mean runtime, where the mean runtime of a configuration is only computed over the $(1-\delta)$ portion of instances with the lowest runtime. In real-world settings, we do not have the luxury of ignoring a part of the instance set, thus we do not view this metric as suitable for evaluating our approach. For the sake of completeness, we nevertheless report the R^δ values in Appendix D.

Initialization of AC-Band. Due to the generality of our approach, a summary statistic s that measures the quality of a configuration needs to be determined. In our case, the k configurations in a subset of CSE can be evaluated in parallel for an instance given that k CPUs are available. When running k configurations in parallel, time can be saved by stopping all remaining configuration runs as soon as the first configuration finishes on the given instance. Through this capping, we obtain right-censored feedback where a runtime is only observed for the "finisher". A statistic that is able to deal with this censored feedback is needed to avoid using an imputation technique that could potentially add bias to the feedback. In line with Brandt et al. (2022), we interpret the obtained feedback as a preference: the finishing configuration is preferred over the remaining, unfinished configurations. Once we have obtained these preferences for multiple instances, we can use a preference-based statistic such as the relative frequency to establish an ordering over the configurations in k. In particular, we count how many times a configuration finishes first over a set of instances[6].

Competitors. AC-Band is compared against ICAR, CAR++ (Weisz et al. 2020) and Hyperband (Li et al. 2016). At the moment, ICAR is the best performing AC method that comes with theoretical guarantees. We use the implementation provided by Weisz et al. (2020) with the same hyperparameter settings. Since AC-Band is inspired by Hyperband, we also adapt Hyperband for the AC setting for a comparison. Specifically, we set the parameter R of Hyperband such that it uses the same total budget (number of instances) as AC-Band. In addition, we use the average runtime over instances as the validation loss and consequently return the configuration with smallest average runtime. Finally, we set $s_{max} = \lceil(\log_\eta(n_{max}))\rceil$, adjust the calculation of r_i slightly to account for instances that were already seen, and try different values of η.

Choice of δ. Varying δ can lead to significantly different performance of AC-Band and other techniques. Due to space constraints, we only show results for two datasets for $\delta = 0.05$ since this setting has also been used in previous work (Weisz et al. 2020). We note, however, that other settings are just as valid, and therefore also provide results for $\delta = 0.01$ and additional datasets in Appendix D. In fact, using $\delta = 0.01$ can result in finding better configurations, albeit it is up to the user of the AC approach to decide which δ best suits their needs.

Q1 Figure 2 shows the CPU time used by each method and the percent gap to best configuration. With a small value of k, AC-Band lies on the Pareto front of percent gap versus CPU time for both datasets (for the third, Regions200, as well). This allows us to answer **Q1** in the affirmative. In particular, with $k = 2$ AC-Band is 72% percent faster than ICAR and

[6]Code: https://github.com/DOTBielefeld/ACBand

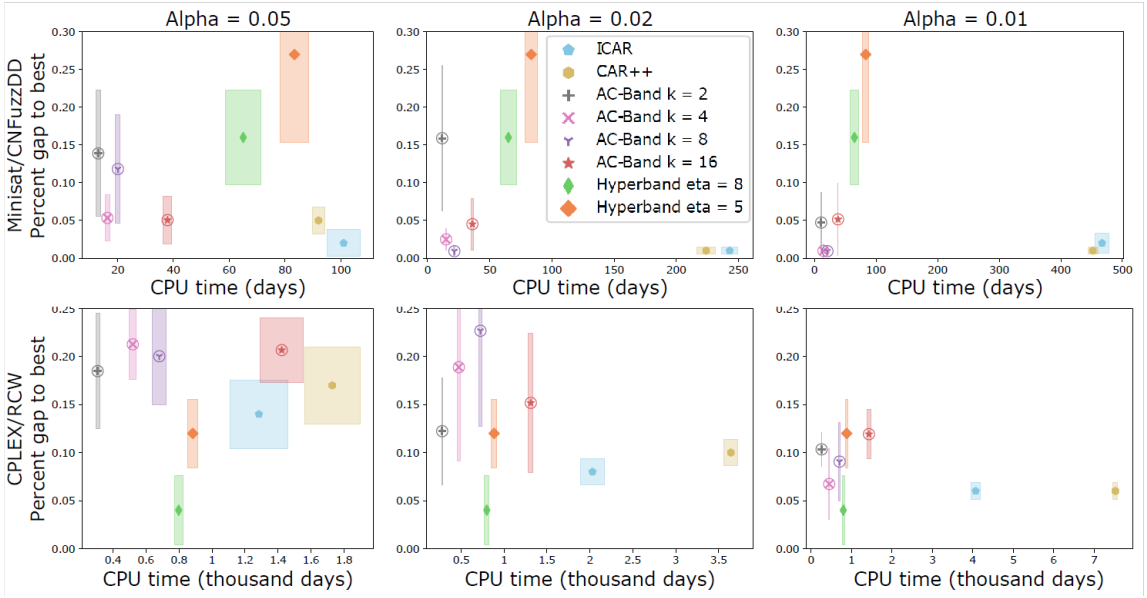


Figure 2: Mean CPU time and percent gap to best over 5 seeds for $\delta = 0.05$ and different α (columns) for AC-Band, ICAR, CAR++ and Hyperband on CNFuzzDD (top) and RCW (bottom). Circles indicate variants of AC-Band. Rectangles represent the standard error over the seeds. The number of configurations tried for CAR++: $\{97, 245, 492\}$, ICAR: $\{134, 351, 724\}$, AC-Band: $\{60, 153, 303\}$, Hyperband($\eta = 5$): $\{842\}$, Hyperband($\eta = 8$): $\{618\}$.

73% faster than Hyperband for $\delta = 0.05$ over all α and datasets, while providing configurations that are only 7% and 6% worse in terms of the gap to the best configuration. For most real-world applications, this is an acceptable trade-off in time versus quality. For all datasets, the percent gap to best decreases when AC-Band samples more configurations (smaller α). This increase in exploration does not lead to a significant increase in CPU time for a fixed k, since AC-Band still has the same budget, i.e., additional configurations are evaluated on fewer instances.

Hyperband samples more configurations in total than AC-Band, which helps it to achieve a better percent gap to best on datasets where the majority of configurations have a high mean runtime. On these datasets, only a few good configurations are present. This is the case for the RCW dataset (and the Regions200 dataset in the appendix) where only a few instances are needed to determine that a configuration should be discarded. On datasets where the runtime of configurations is more evenly distributed, such as the CNFuzzDD dataset, using too few instances may lead to discarding promising configurations early, giving AC-Band an edge by evaluating less configurations more thorough. Lastly, since Hyperband does not use capping, its CPU time deteriorates.

Q2 Our experiments clearly indicate that lower values of k are preferable. With $k = 2$, more CSE rounds are performed and thus the number of configurations decreases slower than with a higher k. With a larger k, the opposite occurs, and significant amounts of CPU time are expended with little information gain. However, note that higher k's have a lower wall-clock time, so a user would receive answers sooner.

Conclusion

In this paper we introduced AC-Band, a versatile approach for AC problems that comes with theoretical guarantees even for a range of target metrics in addition to runtime. We showed that AC-Band returns a nearly optimal configuration w.r.t. the target metric with high probability if used with a sufficient number of problem instances and the underlying AC problem satisfies some mild assumptions. In our experimental study, we considered an instantiation of AC-Band based on preference feedback, which generally leads to faster average CPU times than other theoretical approaches, while still returning a suitable configuration in the end.

Our results open up several possibilities for future work. First, it would be interesting to analyze AC-Band specifically for the case in which runtime is the relevant target metric and investigate whether a similar worst-case overall runtime guarantee can be derived as for the other theoretical approaches in this vein. Next, a theoretical as well as empirical analysis regarding the interplay between the explicit instantiation of AC-Band w.r.t. the underlying statistic s and the characteristics of the underlying AC would be desirable. In other words, on what types of AC problems does a specific instantiation of AC perform well or poorly? Also, our mild Assumption (A1) would even allow some leeway in the configuration or problem instance sampling strategy of the algorithm configurator, which is currently simply uniformly at random for AC-Band. Finally, real-world AC applications generally have only a handful of instances available, thus it would be advantageous to have strong theoretical guarantees even for scenarios without thousands of instances.

Acknowledgements

This research was supported by the research training group Dataninja (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia and by the German Research Foundation (DFG) within the project "Online Preference Learning with Bandit Algorithms" (project no. 317046553).

References

Ahmadizadeh, K.; Dilkina, B.; Gomes, C. P.; and Sabharwal, A. 2010. An empirical study of optimization for maximizing diffusion in networks. In *International Conference on Principles and Practice of Constraint Programming*, 514–521. Springer.

Ansótegui, C.; Malitsky, Y.; Samulowitz, H.; Sellmann, M.; and Tierney, K. 2015. Model-Based Genetic Algorithms for Algorithm Configuration. In *IJCAI*.

Ansótegui, C.; Sellmann, M.; and Tierney, K. 2009. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. 142–157.

Audemard, G.; and Simon, L. 2018. On the glucose SAT solver. *International Journal on Artificial Intelligence Tools*, 27(01): 1840001.

Aziz, M.; Anderton, J.; Kaufmann, E.; and Aslam, J. 2018. Pure exploration in infinitely-armed bandit models with fixed-confidence. In *Algorithmic Learning Theory*, 3–24. PMLR.

Balcan, M.; DeBlasio, D. F.; Dick, T.; Kingsford, C.; Sandholm, T.; and Vitercik, E. 2019. How much data is sufficient to learn high-performing algorithms? *CoRR*, abs/1908.02894.

Balcan, M.; Sandholm, T.; and Vitercik, E. 2020. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, 580–590. PMLR.

Bengs, V.; Busa-Fekete, R.; El Mesaoudi-Paul, A.; and Hüllermeier, E. 2021. Preference-based Online Learning with Dueling Bandits: A Survey. *Journal of Machine Learning Research*, 22: 1–108.

Birattari, M.; Stützle, T.; Paquete, L.; and Varrentrapp, K. 2002. A Racing Algorithm for Configuring Metaheuristics. In *Gecco*, 11–18.

Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L.; Deng, D.; and Lindauer, M. 2021a. Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges.

Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L.; et al. 2021b. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv preprint arXiv:2107.05847*.

Brandt, J.; Haddenhorst, B.; Bengs, V.; and Hüllermeier, E. 2022. Finding Optimal Arms in Non-stochastic Combinatorial Bandits with Semi-bandit Feedback and Finite Budget.

Bubeck, S.; and Cesa-Bianchi, N. 2012. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *CoRR*, abs/1204.5721.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, 23–37. Springer.

Cesa-Bianchi, N.; and Lugosi, G. 2012. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5): 1404–1422.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In *Proceedings of the 30th International Conference on Machine Learning*, 151–159.

de Heide, R.; Cheshire, J.; Ménard, P.; and Carpentier, A. 2021. Bandits with many optimal arms. In *Advances in Neural Information Processing Systems*, volume 34, 22457–22469. Curran Associates, Inc.

Eén, N.; and Sörensson, N. 2003. An extensible SAT-solver. In *International conference on theory and applications of satisfiability testing*, 502–518. Springer.

Hall, G. T.; Oliveto, P. S.; and Sudholt, D. 2019. On the Impact of the Cutoff Time on the Performance of Algorithm Configurators. *CoRR*, abs/1904.06230.

Hall, G. T.; Oliveto, P. S.; and Sudholt, D. 2020. Analysis of the Performance of Algorithm Configurators for Search Heuristics with Global Mutation Operators. *CoRR*, abs/2004.04519.

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2013. Bayesian Optimization With Censored Response Data. *arXiv preprint arXiv:1310.1947*.

Hutter, F.; Hoos, H.; Leyton-Brown, K.; and Stützle, T. 2009. ParamILS: An Automatic Algorithm Configuration Framework. *J. Artif. Intell. Res. (JAIR)*, 36: 267–306.

Hutter, F.; Hoos, H.; and Stützle, T. 2007. Automatic Algorithm Configuration based on Local Search. In *Proceedings of the Twenty-Second Conference on Artifical Intelligence (AAAI '07)*.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization*, 507–523. Springer Berlin Heidelberg.

Hutter, F.; Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2014. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206: 79–111.

IBM. 2020. ILOG CPLEX Optimization Studio 20.1.0: User's Manual.

Jamieson, K. G.; and Talwalkar, A. 2015. Non-stochastic Best Arm Identification and Hyperparameter Optimization. *CoRR*, abs/1502.07943.

Jourdan, M.; Mutnỳ, M.; Kirschner, J.; and Krause, A. 2021. Efficient pure exploration for combinatorial bandits with semi-bandit feedback. In *Algorithmic Learning Theory*, 805–849. PMLR.

Karnin, Z.; Koren, T.; and Somekh, O. 2013. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, 1238–1246. PMLR.

Kleinberg, R.; Leyton-Brown, K.; and Lucier, B. 2017. Efficiency Through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 2023–2031. ijcai.org.

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

Lai, T.; and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1): 4–22.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit algorithms*. Cambridge University Press.

Leyton-Brown, K.; Pearson, M.; and Shoham, Y. 2000. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, 66–76.

Li, L.; Jamieson, K. G.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2016. Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits. *CoRR*, abs/1603.06560.

Liu, S.; Tang, K.; Lei, Y.; and Yao, X. 2020. On Performance Estimation in Automatic Algorithm Configuration. In *The Thirty-Fourth Conference on Artificial Intelligence, AAAI*, 2384–2391. AAAI Press.

López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Birattari, M.; and Stützle, T. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3: 43–58.

Schede, E.; Brandt, J.; Tornede, A.; Wever, M.; Bengs, V.; Hüllermeier, E.; and Tierney, K. 2022. A Survey of Methods for Automated Algorithm Configuration. *CoRR*, abs/2202.01651.

Weisz, G.; György, A.; Lin, W.; Graham, D. R.; Leyton-Brown, K.; Szepesvári, C.; and Lucier, B. 2020. ImpatientCapsAndRuns: Approximately Optimal Algorithm Configuration from an Infinite Pool. In *Annual Conference on Neural Information Processing Systems, NeurIPS*.

Weisz, G.; György, A.; and Szepesvári, C. 2018. LEAPSANDBOUNDS: A Method for Approximately Optimal Algorithm Configuration. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, 5254–5262. PMLR.

Weisz, G.; György, A.; and Szepesvári, C. 2019. CapsAndRuns: An Improved Method for Approximately Optimal Algorithm Configuration. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, 6707–6715. PMLR.

Yang, L.; and Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415: 295–316.

7

Best Arm Identification with Retroactively Increased Sampling Budget for More Resource-Efficient HPO

Author Contribution Statement

The original idea of this works comes from Marcel Wever and was revisited by the author. The theoretical guarantees were proven by the author with support and revisions by Viktor Bengs. The implementation of the experiments was done by Marcel Wever and the paper was written mainly by the author with subsequently revisions by all authors.

Supplementary Material

An appendix to the paper is provided in Appendix C. The code of the official implementation is provided at https://github.com/mwever/incremental-successive-halving.

Best Arm Identification with Retroactively Increased Sampling Budget for More Resource-Efficient HPO

Jasmin Brandt[1] , **Marcel Wever**[2,3] , **Viktor Bengs**[2,3] and **Eyke Hüllermeier**[2,3]
[1]Paderborn University, Germany
[2]LMU Munich, Germany
[3]Munich Center for Machine Learning, Germany
jasmin.brandt@upb.de, {viktor.bengs, marcel.wever, eyke}@ifi.lmu.de

Abstract

Hyperparameter optimization (HPO) is indispensable for achieving optimal performance in machine learning tasks. A popular class of methods in this regard is based on Successive Halving (SHA), which casts HPO into a pure-exploration multi-armed bandit setting under finite sampling budget constraints. This is accomplished by considering hyperparameter configurations as arms and rewards as the negative validation losses. While enjoying theoretical guarantees as well as working well in practice, SHA has several hyperparameters itself, one of which is the maximum budget that can be allocated to evaluate a single arm (hyperparameter configuration). Although there are already solutions to this meta hyperparameter optimization problem, such as the doubling trick or asynchronous extensions of SHA, these are either practically inefficient or lack theoretical guarantees. In this paper, we propose incremental SHA (iSHA), a synchronous extension of SHA, allowing to increase the maximum budget a posteriori while still enjoying theoretical guarantees. Our empirical analysis of HPO problems corroborates our theoretical findings and shows that iSHA performs more reliably than existing SHA-based approaches.

1 Introduction

Hyperparameter optimization (HPO) is a crucial step in the process of engineering machine learning (ML) applications, as optimal performance can only be obtained if parameterized ML algorithms are tuned to the task at hand [Feurer and Hutter, 2019; Bischl *et al.*, 2023]. Such a task is specified in the form of a dataset $\mathcal{D}$ and a loss function ℓ. Typically, HPO is carried out in a trial-and-error fashion by evaluating ℓ on the given data $\mathcal{D}$ for various candidate hyperparameter configurations (HPCs).

In the early days of HPO, grid search and random search [Bergstra *et al.*, 2011] have been the main tools. However, they can be criticized for their disability in finding an optimal hyperparameter configuration as well as their computational cost. In the age of deep learning, a highly efficient HPO method is inevitable, as evaluating hundreds or even thousands of configurations is prohibitive. To address this challenge, several HPO methods have been proposed to improve sampling or evaluation efficiency. For the former, the methods mainly focus on Bayesian Optimization [Hutter *et al.*, 2011], whereas for the latter, the HPO problem is extended by a budget parameter. Using this parameter, the optimizer can specify for which budget a hyperparameter configuration should be evaluated. This area of the HPO literature is also referred to as multi-fidelity optimization.

Probably the simplest procedure in this area is the successive halving algorithm (SHA) [Karnin *et al.*, 2013], which is rooted in the multi-armed bandit (MAB) literature [Lattimore and Szepesvári, 2020]. It first evaluates a set of candidate hyperparameter configurations (arms) for a minimum starting budget R_0, discards the worse half, and continues evaluation with the better half for a doubled budget. This procedure is repeated until a maximum budget of R is reached. By concentrating the budget on more promising hyperparameter configurations, the reliability of the evaluations is gradually increased, but also their evaluation costs. In contrast, less promising solutions are discarded early on with little budget.

Following the terminology of the multi-armed bandits, SHA attempts to solve a best arm identification problem for a given fixed sampling budget. [Jamieson and Talwalkar, 2016b] as well as [Li *et al.*, 2018] have derived theoretical bounds on the necessary sampling budget to guarantee to find an optimal or near-optimal arm (hyperparameter configuration) with high probability. However, these theoretical bounds have two problems regarding practical application: Firstly, they depend on problem parameters that are unknown in practice, and secondly, they are derived for worst-case scenarios and are therefore often too conservative. Accordingly, the common approach is to set a budget in an ad-hoc manner and later check whether it was sufficiently large to support the reliability of the returned HPC. If this is not the case, the budget is increased retrospectively, which is problematic as SHA is not incremental by design, so that the entire algorithm must be re-run. This is not only costly but also comes with a loss of valuable knowledge already accumulated. Needless to say, from an ecological perspective, this is undesired either, as the computational resources, as well as the consumed energy for optimizing the hyperparameters for the lower maximum budget, is essentially wasted [Tornede *et al.*, 2023].

Although there are two variants of SHA that do not need to specify the maximum budget R in advance, these have the decisive disadvantage that they come without theoretical guarantees. Asynchronous SHA (ASHA) [Li *et al.*, 2020] is the first variant, in which decisions about candidate evaluations for larger budgets are made asynchronously, allowing for higher parallelization. This variant has recently been further developed into PASHA [Bohdal *et al.*, 2023], which progressively increases the budget if the ranking of the configurations in the top two high-fidelity sets has not stabilized. However, asynchronous decision-making comes at the risk of mistakenly promoting HPCs to the next budget level. While [Li *et al.*, 2020] invoke the law of large numbers to argue that this is not an issue, the problem remains in the case of finite budget constraints, where only a limited number of hyperparameter configurations can be considered.

Contributions. We will focus on the HPO application for the most part when presenting the necessary concepts and our results. However, our theoretical results apply to the more general bandit setting and can therefore be carried over to applications other than HPO. Following the terminology of the MAB literature, we are considering the best arm identification (BAI) problem for which the sampling budget is increased retroactively. Our contributions can be summarized as follows:

- We provide the first theoretical results for ASHA, analyzing its capabilities in setups with constraints on the overall budget. These findings are accompanied by empirical evidence for a set of HPO benchmarks.
- We propose an incremental extension of SHA (iSHA) that still allows one to increase the maximum allocatable budget R retrospectively in a synchronous manner.
- A theoretical and empirical analysis of iSHA is provided, finding iSHA to be theoretically sound relative to the original SHA, while being provably more resource-efficient.
- In an extensive empirical study, we compare iSHA to the original SHA, and PASHA embedded into the Hyperband framework. We find iSHA to give more robust results compared to PASHA, often yielding higher quality hyperparameter configurations, while being more resource-efficient than SHA.
- We show a long-missing lower bound on the necessary budget for finding a nearly-optimal arm under common assumptions of the non-stochastic BAI problem.

2 (Near-)Optimal Arm Identification

The best arm identification problem in multi-armed bandits (MABs) is a sequential decision-making problem in which an agent has to choose in each time step $t \in \{1, \dots, B\}$ within a fixed budget $B \in \mathbb{N} \cup \{\infty\}$ one out of $n \in \mathbb{N}$ possible options that we denote by their indices $[n] := \{1, \dots, n\}$ and call arms in the following. After choosing (or pulling) one arm, the agent directly observes a loss $\ell(i)$ for the chosen arm $i \in [n]$ given by a loss function $\ell : [n] \to \mathbb{R}$. The observed loss after r pulls of arm $i \in [n]$ will be denoted $\ell_r(i)$. In the non-stochastic setting, which is the setting we consider, the observed losses are not necessarily governed by an underlying stochastic distribution. Instead, a common assumption is that the sequence of losses of an arm converges asymptotically to a fixed final value [Jamieson and Talwalkar, 2016b; Li *et al.*, 2018; Brandt *et al.*, 2023].

Assumption 2.1. $\forall i \in [n]$ and for $R \in \mathbb{N} \cup \{\infty\}$ the loss function converges against a limit value $\nu_i = \lim_{r \to R} \ell_r(i)$.

Note, that the stochastic scenario in which $(\ell_k(i))_{k \geq 1}$ for each $i \in [n]$ is an i.i.d. sample from a stochastic distribution with $\mathbb{E}[\ell_k(i)] = \nu_i$ can be treated as a special case of our setting [Jamieson and Talwalkar, 2016b].

Goal. Usually, the goal in best arm identification is to find the arm with the smallest loss $i^* \in \arg\min_{i \in [n]} \nu_i$. We will relax this goal to only identify a near-optimal arm where "near-optimal" is defined in the following way.

Definition 2.2. Let $\epsilon > 0$, then we call $\hat{i} \in [n]$ an ϵ-optimal arm if $\nu_{\hat{i}} \leq \nu_{i^*} + \epsilon$.

It is straightforward to extend the scenario to the case in which we deal with a countably infinite set of stochastic bandit arms indexed by $i = 1, 2, \dots$

Hyperparameter Optimization. Hyperparameter optimization (HPO) deals with the problem of finding a suitable hyperparameter configuration $\boldsymbol{\lambda}$ of an ML algorithm $\mathcal{A}$ with a corresponding hyperparameter space Λ for a given learning task (e.g. image classification, regression analysis, etc.). For a suitable loss function ℓ and a finite set of HPC samples, say $\tilde{\Lambda}$, from the possibly uncountable infinite space Λ, we can consider each HPO problem as a MAB problem, by simply considering the HPCs as arms. An example of a loss function is the validation error of a (supervised) learning algorithm $\mathcal{A}$ with parameterization $\boldsymbol{\lambda}$ and resource allocation r, which could be for instance the wall-clock time, number of used data points, etc. Sampling or generating a finite set of HPCs $\tilde{\Lambda}$ can be done in different ways, for example, simple uniform sampling from Λ [Li *et al.*, 2018] or by leveraging a Bayesian search mechanism [Falkner *et al.*, 2018].

3 Successive Halving and Hyperband

The successive halving algorithm (SHA) by [Karnin *et al.*, 2013] is applicable for the non-stochastic best arm identification problem with a finite set of arms under a fixed budget constraint. By sampling n hyperparameter configurations (HPCs) uniformly at random, it has already been applied successfully to HPO by [Jamieson and Talwalkar, 2016a]. Starting from a minimum budget R_0 for which all the n available arms are evaluated, it iteratively discards the worse half and continues to evaluate the remaining arms with double the budget. This procedure is repeated until either only a single arm is left or a maximum allocatable budget R is reached (maximum for a single arm). Typically, the number of arms n is chosen such that at least one candidate reaches the final iteration of the algorithm. A budget level for which arms are evaluated is also referred to as *rung* in the following. Furthermore, we write that an arm is *promoted* to the next rung if it was not discarded and thus considered in the next iteration of SHA. While SHA allows allocating exponentially more budget on the more promising arms, its final performance crucially depends on its parameterization. The parameters n, R

and R_0 need to be chosen with care and depending on the task. With regard to HPO, starting with a too low initial budget R_0, we face the problem of rejecting actually promising arms (HPCs) too early, namely those that require more budget, e.g., more data or more training iterations, to perform well enough to remain in the set of promising candidates.

The Hyperband (HB) algorithm [Li *et al.*, 2018] comes with a heuristic of how to choose different values for n and R_0, and subsequently uses SHA as a subroutine. Even if it can generally be used for a bandit problem with an infinite number of arms, its design is tailored to HPO. HB allows different allocation strategies to be considered for the tradeoff between (i) considering many arms (HPCs) n starting with a rather small R_0, and (ii) giving some arms (HPCs) more budget from the beginning. The latter is motivated by the fact that in machine learning, some HPCs may require a larger amount of resources to show off their better performance. We refer to each call of SHA as a *bracket* [Li *et al.*, 2018], for which the set of arms (HPCs) is sampled uniformly at random and given to SHA as an input.

4 Related Work

The pure-exploration and best arm identification problem in MABs has been studied intensively with a stochastic feedback mechanism (see [Gong and Sellke, 2023] for a more recent overview). Especially in the case of a fixed budget, there is some work in this direction [Carpentier and Valko, 2015; Abbasi-Yadkori *et al.*, 2018; Shen, 2019; Azizi *et al.*, 2022; Kato *et al.*, 2022]. However, the non-stochastic setting, as considered in our work, has so far only been investigated in [Jamieson and Talwalkar, 2016a; Li *et al.*, 2018] and [Brandt *et al.*, 2022]. The first two with a special focus on HPO similar to our work and the latter with a focus on algorithm selection/configuration [Rice, 1976].

Considering HPO as a black-box optimization problem, various methods can be used to tackle this problem [Feurer and Hutter, 2019; Bischl *et al.*, 2023]. Grid search and random search are rather straightforward solutions. However, both are rather expensive, and thus, methods emerged to improve sample efficiency and evaluation efficiency. While the former methods are mostly centered around Bayesian optimization [Frazier, 2018; Hutter *et al.*, 2011], the latter emerged in the branch of multi-fidelity optimization.

In multi-fidelity optimization, the goal is to distribute the budget for evaluating HPCs in a way that more budget is concentrated on the more promising HPCs and less so on inferior candidates. The successive halving algorithm (SHA), initially proposed by [Karnin *et al.*, 2013] and later used by [Jamieson and Talwalkar, 2016b; Jamieson and Talwalkar, 2016a] for HPO, devises a powerful HPO method, which has been incorporated as a subroutine in the well-known HPO method Hyperband [Li *et al.*, 2018]. Hyperband has been extended in various directions such as improving its sampling efficiency [Falkner *et al.*, 2018; Awad *et al.*, 2021; Mallik *et al.*, 2023] and introducing shortcuts in the evaluation process [Mendes *et al.*, 2021].

But also SHA has been subject to improvements. [Li *et al.*, 2020] extend SHA to asynchronous SHA (ASHA), which helps to better leverage parallel computing resources by promoting candidates asynchronously to the next rung. Simultaneously, the maximum budget R can be adapted on-the-fly. Progressive ASHA (PASHA) proposed by [Bohdal *et al.*, 2023] builds on ASHA and incorporates a mechanism to only introduce higher rungs where necessary. While both ASHA and PASHA have been extensively studied empirically, a thorough (theoretical) analysis of the costs of the asynchronous promotion scheme is still lacking. Also, these empirical studies have considered comparably large setups with vast amounts of resources. In our study, we consider small-scale setups and analyze the behavior of ASHA and PASHA in that scope.

Algorithm 1 Incremental Successive-Halving Algorithm (`iSHA`)

Input: S initial set of HPCs, r, maximum resource R, reduction factor η, $(C_k)_k$ sequence of HPCs promoted in previous run, $(L_k)_k$ old sequence of losses
Initialize: $S_0 \leftarrow S$, $\tilde{n} = |C_0|$, $n = |S_0| + |C_0|$, $s = \log_\eta(R)$

for $k \in \{0, 1, \ldots, s\}$ **do**
  $n_k = \lfloor n/\eta^k \rfloor - \lfloor \tilde{n}/\eta^k \rfloor$, $r_k = r\eta^k$
  pull each arm in S_k for r_k times
  if $k \leq s - 1$ **then**
    $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor - \lfloor \tilde{n}/\eta^{k+1} \rfloor$ arms from $S_k \cup C_k \backslash C_{k+1}$
  else
    $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor$ arms from $S_k \cup C_k$
  end if
end for
Output: Remaining configuration

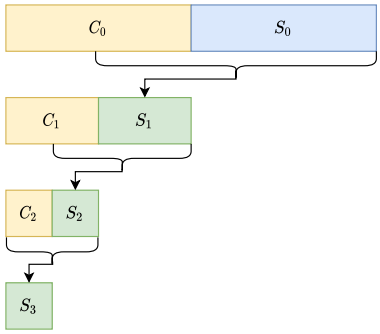


Figure 1: Illustration of how iSHA continues a previously conducted SHA run.

5 Incremental Successive Halving

Due to the static budget setting in SHA, the execution of SHA cannot simply be continued for an adapted parameterization, e.g., a higher maximum allocatable budget R. By rerunning SHA from scratch, however, knowledge about previously evaluated hyperparameter configurations (HPCs) is discarded and resources already allocated are wasted. As another extreme, ASHA and PASHA allow to dynamically increase the maximum allocatable budget R, devising a scheme

for asynchronous promotions to higher rungs. However, as we show in Sections 6 and 7.2, the asynchronous promotions in ASHA and PASHA can be erroneous and thus impede the identification of the optimal hyperparameter configurations.

With incremental successive halving (iSHA), we propose a middle ground for budget-constrained scenarios, i.e., scenarios in which we cannot rely on the law of large numbers as required by [Li *et al.*, 2020]. Similar to ASHA and PASHA, we allow the maximum allocatable budget to be increased after an SHA run, making SHA in principle stateful. Algorithm 1 translates this into pseudocode. Differences from the original SHA are highlighted in blue. While Algorithm 1 also covers the case of ASHA, adding a single configuration at a time, we assume $|S| + |C_0| = |C_0| \cdot \eta$ for our theoretical and empirical analysis, where C_0 are the configurations in the beginning of the previous run, S are the new configurations and $\eta \geq 1$ is a reduction factor.

In Figure 1 we see the mechanism underlying iSHA to continue a previously conducted run of SHA that resulted in the rungs C_0, C_1 and C_2. The initially sampled set of HPCs C_0 is padded with newly sampled HPCs S_0 to initially achieve the same number of HPCs as if SHA had been restarted. However, only the new configurations are executed (following the typical SHA budget allocation) and finally compared with the previous configurations from C_0. The already promoted configurations in C_1 from the previous SHA run will remain and only the required number of configurations will be promoted, i.e., S_1, such that the size of the union of C_1 and S_1 matches the size of the second rung if SHA had been restarted. This mechanism is then iteratively continued for subsequent rungs.

Intuitively speaking, the strategy of iSHA is to continue a previous SHA run in the most efficient, and thus, resource-saving way. However, similarly to ASHA and PASHA, this efficiency may come at the cost of a potential drop in performance, as previously made decisions cannot be revoked. More specifically, in the worst case, all promotions of the previous run would not have occurred if we had known the complete set of candidate HPCs from the start. Only filling up the rungs leaves less space for the desired candidates to be promoted to the highest rung.

Nevertheless, we prove in the next section that we are still able to identify near-optimal solutions with a high probability, which will be confirmed by empirical results in Section 7.2 later on. Furthermore, we demonstrate that this robustness gives iSHA an edge over ASHA and PASHA when it comes to the quality of returned hyperparameter configurations in settings with limited budget.

6 Theoretical Results

We split the theoretical results into four parts. First, we present a lower bound for the necessary budget for the ϵ-optimal arm identification problem. Second, we provide a theoretical analysis of ASHA. Third, we give some theoretical guarantees for iSHA, our extension of SHA, and fourth, we extend these guarantees to an incremental extension of Hyperband.

As a prerequisite for the theoretical results, we ease the notation by simply writing $\ell_{i,t}$ instead of $\ell_t(i)$. Since by assumption 2.1 there exists a limit value of the loss function for every arm, we denote the corresponding convergence speed by $\gamma(t) \geq \sup_i |\ell_{i,t} - \nu_i|$, $\forall t \in \mathbb{N}$.

6.1 Lower Bound

Although there is already literature tackling the problem of ϵ-optimal arm identification in MABs like [Li *et al.*, 2018], a lower bound for the necessary budget was missing until now.

Theorem 6.1 (Lower bound for $\epsilon/2$-optimal arm identification)**.** *If an algorithm Alg correctly identifies an $\epsilon/2$-optimal arm for any loss function ℓ, then there exist slightly modified limits $\{\tilde{\nu}_i\}_{i\in[n]}$ with $|\nu_i - \tilde{\nu}_i| \leq \epsilon/2$ for each $i \in \{1, \dots, n\}$ such that Alg needs at least $n \cdot \gamma^{-1}\left(\max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\}\right)$ total pulls of arms in expectation.*

The proof is deferred to Section D.

6.2 Theoretical Analysis of ASHA

We now analyze ASHA [Li *et al.*, 2020], which, to the best of our knowledge, is the only algorithm with a similar goal of more efficient resource use as our proposed iSHA.

Theorem 6.2 (Necessary Budget for ASHA)**.** *Fix n arms and assume $\nu_1 \leq \ldots \leq \nu_n$. Let*

$$z_{\mathrm{ASHA}} = n\big(\lfloor \log_\eta(n) \rfloor + 1\big) \cdot \max\Big\{ \max_{k\in[K]} \eta^{-k} \cdot \gamma^{-1}\Big(\tfrac{\nu_{\lfloor |rung_{k-1}|/\eta \rfloor + 1} - \nu_1}{2}\Big), \; \eta^{-K} \max_{i \in rung_K \setminus \{1\}} \gamma^{-1}\Big(\tfrac{\nu_i - \nu_1}{2}\Big)\Big\},$$

where $K \leq \lfloor \log_\eta(n) \rfloor$ is the top rung of ASHA. If ASHA's total number of pulls exceeds z_{ASHA}, then the best arm is returned.

The dependence is linear-logarithmic in n, and the limit gap from the best arm to the other arms occurs in the inverted convergence rate γ^{-1}. The first term in the maximum makes sure that the best arm reaches the top rung K, while the second term makes sure that the best arm will eventually be returned. In view of the lower bound in Theorem 6.1, ASHA is nearly optimal.

As a corollary of Theorem 6.2 (see the proof in Section C), we obtain the following result regarding the mechanism of the sampling process pursued by ASHA.

Corollary 6.3 (Worst Case Promotion Costs)**.** *Assume all rungs to be full, i.e., no promotion is possible, and the top rung K only contains the current incumbent arm. If at that time a new best arm (HPC) $\hat{i}$ is sampled, then promoting $\hat{i}$ to the sole solution of the new top rung $K+1$ requires the sampling of $\eta^K - 1$ additional arms (HPCs) and a total of η^{K+1} many jobs.*

From these results, we can draw two major conclusions. The more arms (HPCs) have already been considered in ASHA when $\hat{i}$ enters the pool of considered hyperparameter configurations, i.e., the later in the process, the more budget needs to be spent to promote $\hat{i}$ to the top rung. Particularly, in a scenario with a limited budget, e.g., limited by the overall budget (total number of pulls) or by the number of arms

to be sampled, ASHA fails to return $\hat{i}$, if the required budget for promoting the best configuration exceeds the remaining budget. A similar result can be shown for PASHA, since its sampling mechanism is similar to ASHA.

6.3 Theoretical Analysis of iSHA

For iSHA (Algorithm 1), we first prove a lower bound on the necessary budget to return a nearly optimal arm (configuration), when $n/\tilde{n} = \eta$ which corresponds to $|S| + |C_0| = |C_0| \cdot \eta$. The proof is given in Appendix B.1.

Theorem 6.4 (Necessary Budget for iSHA)**.** *Fix n arms from which $\tilde{n}$ arms were already considered in a previous run, and assume $\nu_1 \leq \cdots \leq \nu_n$ as well as $r \in (R/\eta^s)_{s=0,\ldots,\log_\eta(R)}$. For any $\epsilon > 0$ let*

$$z_{\text{iSHA}} = \eta\lceil \log_\eta(n) \rceil \cdot \max_{i=2,\ldots,n} i\Big(1 + \min\Big\{R, \gamma^{-1}\Big(\max\Big\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\Big\}\Big)\Big\}\Big).$$

If iSHA's total number of pulls exceeds z_{iSHA}, then an arm $\hat{\imath}$ is returned that satisfies $\nu_{\hat{\imath}} - \nu_1 \leq \epsilon/2$.

As the dependency on n and the gap is similar as for ASHA, we conclude from Theorem 6.1 that iSHA is nearly optimal as well. Further, we can specify the improvement of iSHA over the costly re-run of SHA.

Theorem 6.5 (Improvement of number of pulls of iSHA in comparison to SHA)**.** *Fix a maximal budget per arm of R, r and η. Assume that we have already run SHA on $\tilde{n}$ arms with R, r, and η. Now sample $n - \tilde{n}$ new arms with $n = \tilde{n}\eta$, and (re-)run SHA and iSHA over s rounds with the above variables. Then, for $\eta_- = \eta - 1$ and $s^+ = s + 1$ we have*

$$\frac{\#\{\textit{total pulls of iSHA}\}}{\#\{\textit{total pulls of SH}\}} \leq 1 - \frac{(s^+)(\tilde{n}R + \eta^s)(\eta_-) - (\eta^{s^+} - 1)(2R + n)}{(s^+)(nR + \eta^s)(\eta_-) - (\eta^{s^+} - 1)(R + n)}.$$

Again, as a corollary of Theorem 6.5 (see Section Appendix B.2), we obtain the following result regarding the "limit case", i.e., if we would increase the maximum size R infinitely often, or, equivalently, the number of possible rungs s infinitely often.

Corollary 6.6. *If we run iSHA and SHA infinitely often with*

(i) an ever-increasing maximum size R, and

(ii) such that the newly sampled number of configurations in each new run of iSHA fulfills $|S| + |C_0| = |C_0| \cdot x$, where C_0 is the number of configurations in the previous run and $x > 1$,

then the ratio of total pulls of iSHA and total pulls of SHA converges to $1 - x^{-1}$.

In our setting, we use $x = \eta$, so that the improvement ratio is $1 - \eta^{-1}$. Note that a comparison similar to Theorem 6.5 or Corollary 6.6 is difficult to make for ASHA or PASHA, since both do not include the parameter R. Finally it is worth mentioning that no similar statement as Corollary 6.3 holds for iSHA, since $\hat{\imath}$ can be still (and very likely will be) returned as an output for the available budget.

6.4 Incremental Hyperband

Like the original version of SHA and its extensions ASHA and PASHA, we can also employ iSHA as a subroutine in Hyperband. To this end, Hyperband needs to be made incremental itself, as done in Algorithm 4 in the appendix, which we call incremental Hyperband (iHB). In the following, we provide a theoretical analysis of this incremental version of Hyperband with iSHA as a subroutine. Figure 3 in the appendix illustrates how every Hyperband bracket is updated after increasing the maximum budget R.

Recall, that for $\epsilon > 0$, we aim to find an *ϵ-optimal configuration* $\hat{\boldsymbol{\lambda}}$ which was defined as $\nu_{\hat{\boldsymbol{\lambda}}} - \nu_{\boldsymbol{\lambda}^*} \leq \epsilon$. for $\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \Lambda} \nu_{\boldsymbol{\lambda}}$. To ensure that the search is possible by sampling merely a finite subset of HPCs, we make the following assumption similar to [Brandt *et al.*, 2023]:

Assumption 6.7. The proportion of ϵ-optimal configurations in Λ is $\alpha \in (0, 1)$.

Note that we now have at least one ϵ-optimal configuration in a sampled set of configurations with probability at least $1 - \delta$, if the sample size is at least $\lceil \log_{1-\alpha}(\delta) \rceil$ for a fixed failure probability $\delta \in (0, 1)$. With this, we can state the following theorem, the proof of which is given in Appendix B.3.

Theorem 6.8. *Let η, R, α and δ be fixed such that*

$$R \geq \max\Big\{ \lceil \log_{1-\alpha}(\delta) \rceil (\eta_-) + 1, \eta\bar{\gamma}^{-1}\Big(L_{\eta, L_{\eta,R}} + 4 + \frac{\lfloor L_{\eta,R} \rfloor}{2} - \frac{\sum_{k=1}^{\lfloor L_{\eta,R} \rfloor + 1} \log_\eta(k)}{\lfloor L_{\eta,R} \rfloor + 1}\Big)\Big\}$$

for $\bar{\gamma}^{-1} := \max_{s=0,\ldots,\lfloor L_{\eta,R} \rfloor} \max_{i=2,\ldots,n_s} i\Big(1 + \min\Big\{R, \gamma^{-1}\Big(\max\Big\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\Big\}\Big)\Big\}\Big)$

and $L_{\eta,R} = \log_\eta(R)$, then iHB finds an ϵ-optimal configuration with probability at least $1 - \delta$.

To conclude, despite the incremental extension of Hyperband, we can maintain the theoretical guarantees of the original Hyperband. Although promotions in iSHA are also to some extent performed asynchronously, we can still identify a nearly best arm when doing promotions in a batch, provided a sufficiently large batch size.

7 Empirical Evaluation

In addition to the theoretical results of the previous section, we evaluate iSHA empirically and compare it to PASHA [Bohdal *et al.*, 2023] and SHA [Jamieson and Talwalkar, 2016a]. We are especially interested in the following two research questions:

RQ1 Is iSHA able to retain the quality of returned HPCs as compared to applying SHA from scratch?

RQ2 How does the proposed iSHA compare to the state-of-the-art algorithms ASHA and PASHA?

7.1 Experiment Setup

In our experimental evaluation, we compare iSHA to two asynchronous extensions of SHA, namely ASHA and PASHA. For the comparison, we integrate all SHA variants as subroutines in Hyperband to answer the research questions **RQ1** and **RQ2**. To this end, we conduct extensive experiments tackling numerous HPO tasks, considering various

Benchmark	Model	# Inst.	Objective	Fidelity
lcbench	neural network	34	val_accuracy	epochs
rbv2_svm	SVM	106	acc	fraction
rbv2_ranger	random forest	119	acc	fraction
rbv2_xgboost	XGBoost	119	acc	fraction
nb301	neural network	1	val_accuracy	epochs

Table 1: List of considered benchmarks from YAHPO-Gym with the type of learner, number of considered datasets, objective function, and the type of budget that can be used as a fidelity parameter.

Benchmark / ϵ	0.001	0.005	0.01	0.05
lcbench	0.0004±0.0011	0.0042±0.0147	0.0095±0.0284	0.0919±0.1599
nb301	0.0001±0.0000	0.0001±0.0000	0.0078±0.0000	0.8962±0.0000
rbv2_svm	0.0092±0.0397	0.0554±0.1296	0.126±0.1995	0.4927±0.2995
rbv2_ranger	0.0026±0.0095	0.0365±0.1323	0.0732±0.1967	0.5336±0.3778
rbv2_xgboost	0.0123±0.0308	0.0213±0.0560	0.0316±0.0845	0.1541±0.2306

Table 2: Mean (± standard deviation) proportion of 10,000 randomly sampled hyperparameter configurations that are within an ϵ distance of the best hyperparameter configuration's performance.

types of learners and two different fidelity parameters: the number of *epochs* and the *fraction* of the training data used for fitting a model.

As a benchmark library, we use YAHPO Gym [Pfisterer *et al.*, 2022], which provides fast-to-evaluate surrogate benchmarks for HPO with particular support for multi-fidelity optimization, rendering it a perfect fit for our study. From YAHPO Gym, we select the benchmarks listed in Table 1. All the benchmarks consist of several datasets, which are referred to as benchmark instances, allowing for a broad comparison. Due to space limitations, we only present a summary of the results here, whereas detailed results can be found in Appendix E.

In Table 2, we show the mean fraction of 10,000 randomly sampled hyperparameter configurations to be at most ϵ worse than the best hyperparameter configuration. As can be seen, the considered benchmarks are of varying difficulty and the size of the ϵ-optimal fraction also varies substantially in size across the datasets contained in the corresponding benchmark suites as indicated by the standard deviation.

Furthermore, we set the initial max size $R_{t-1} = 16$ and increase it after the first run by a factor of η to $R_t = \eta R_{t-1}$, as this is a budget that is supported by all benchmark scenarios. Since ASHA and PASHA automatically increase the maximum budget depending on the observed performances, we only ensure an upper limit of R_t for both to ensure a fair comparison. As a termination criterion, we use that the number of HPCs would exceed the pool size of the Hyperband bracket. For benchmarks considering a fraction of the training dataset as a fidelity parameter, we translate a budget r by r/R_t into a fraction between 0 and 1.

Furthermore, we repeat each combination of algorithm, η, and benchmark instance for 30 seeds, resulting in a total amount of $30 \times 3 \times 2 \times 379 = 68,220$ hyperparameter optimization runs. We run all experiments on a single workstation equipped with 2xIntel Xeon Gold 5122 and 256GB RAM. The code is publicly available via GitHub[1].

[1]https://github.com/mwever/incremental-successive-halving

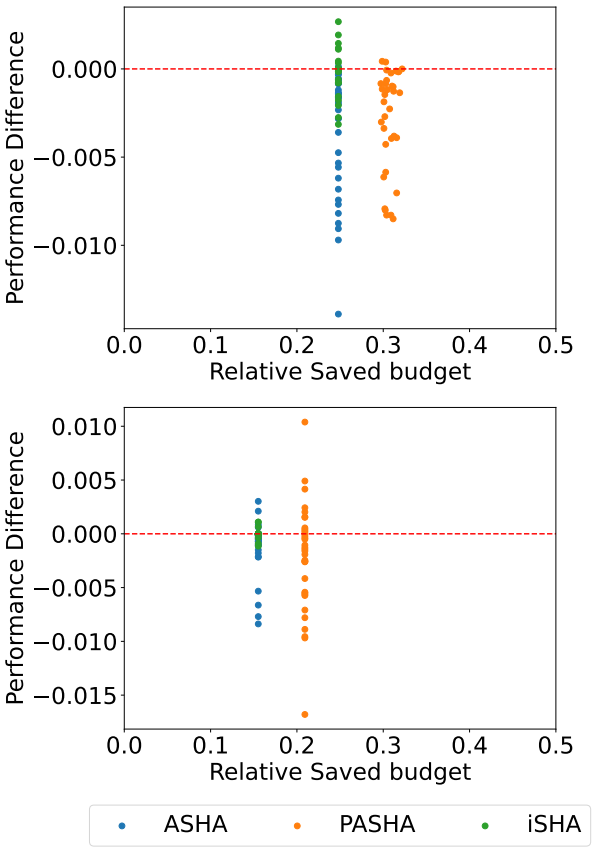


Figure 2: Scatter plots relating the performance on the y-axis and the consumed budget on the x-axis to the performance achieved and budget consumed by SHA. Note that the ranges for the performance and budget vary from $\eta = 2$ (top) to $\eta = 3$ (bottom). Higher values are better for both relative saved budget and relative performance.

7.2 Empirical Results

In Figure 2 we present the performance of the finally chosen hyperparameter configuration and the budget saved by ASHA, PASHA, and iSHA relative to the performance of the solution returned and the budget consumed by re-running SHA from scratch for the higher maximum budget R_t. Hence, a relative performance of 0.0 means that the solution quality matches the one returned by SHA, which is also indicated by the red dashed line, a larger (smaller) value means a performance improvement (degradation) w.r.t. SHA. The relative saved budget denotes the percentage of the budget that any of the approaches saves in contrast to the re-run of SHA. Therefore, a relative saved budget of 0 means that the consumed budget is on par with the budget of SHA. A higher relative saved budget correspondingly means that the approach was more efficient than SHA.

As can be seen, iSHA robustly yields competitive performance to re-running SHA from scratch for a larger maximum assignable budget R, while substantially reducing the consumed budget to roughly 75% for $\eta = 2$ and 84.5% for $\eta = 85\%$. Regarding **RQ1**, we can confirm that iSHA retains the quality of returned HPCs.

		Performance			Budget	
	Approach	Impr	Degr	Tie	Mean	Std
$\eta = 2$	PASHA	12	72	295	0.6926	0.007
	ASHA	9	81	289	0.7520	0.0
	iSHA	4	5	370	0.7520	0.0
$\eta = 3$	PASHA	49	114	216	0.7908	0.0
	ASHA	11	75	293	0.8448	0.0
	iSHA	4	14	361	0.8448	0.0

Table 3: Aggregated statistics across benchmark instances comparing the performance and budget to natively applying SHA. Differences in accuracy larger than 0.001 are considered for improvements or degradations.

On the contrary, the performances of ASHA and PASHA show way more variance, including variations. Since higher rungs are only introduced in PASHA whenever necessary, i.e., if the soft ranking over the configurations of the last two rungs changes, PASHA has the potential to reduce the consumed budget even more than iSHA or ASHA do. However, there is no guarantee that this will maintain performance. As can be seen for $\eta = 3$, PASHA is clearly less robust than ASHA suggesting that the progressive nature of PASHA is introducing even more variance.

This is again confirmed by the results in Table 3, where we simply count the number of benchmark instances for which an improvement, degradation, or tie w.r.t. the performance of re-running SHA is obtained. While PASHA gives the most improvements in terms of performance for both values of η, it also comes with the most performance degradations for $\eta = 3$ which outnumber the improvements by a factor of 2 to 3, whereas for $\eta = 2$ degradations occur more frequently than improvements by a factor of 4 to 5. Furthermore, we provide the average and the standard deviations for the relative budget consumed across the benchmark instances. On average, for both values of η, iSHA and ASHA reduce the budget consistently but PASHA can reduce the budget even more.

While, of course, performance improvements are desirable, the selection of the returned solution is made on the same set of candidates in all approaches, including SHA. Therefore, improvements cannot be interpreted as an advantage of one method over SHA but as random noise effects as these decisions highly depend on the order of hyperparameter configurations being considered in the successive halving variant. Assuming the original behavior of SHA to be the ground truth behavior, we can thereby define a consistency metric between ASHA, PASHA, iSHA and the ground truth behavior of SHA. In Table 4, we present consistencies of ASHA, PASHA, and iSHA to SHA. Fixing a benchmark, the consistency is calculated as follows:

$$(|M|)^{-1} \cdot \sum_i [\![|\mu_{SHA}(i) - \mu_{xSHA}(i)| \leq 0.001]\!],$$

where $[\![\cdot]\!]$ is the indicator function, $\mu_{SHA}(i)$ is the full budget performance for the returned hyperparameter configurations by SHA on instance i and $\mu_{xSHA}(i)$ the performance of another considered approach xSHA and M is the number of instances in the respective benchmark. In this comparison, iSHA stands out to be by far the most consistent approach.

η	ASHA	PASHA	iSHA
2	0.7625	0.7783	**0.9763**
3	0.5699	0.7731	**0.9525**

Table 4: Consistency of the performance of hyperparameter configurations returned by ASHA, PASHA, and iSHA with the performance of those returned by SHA.

From these results, we can conclude that iSHA is a robust and more resource-efficient incremental version of SHA, and the theoretical guarantees given in the previous section can be validated in practice as well. PASHA is able to reduce the consumed budget drastically. However, the reduced budget comes at the risk of losing consistency and lack of performance guarantees. In turn, ASHA reduces the consumed budget in the same way as iSHA does but also performs poorly in terms of consistent behavior with the original SHA version.

8 Conclusion and Future Work

In this paper, we proposed an extension to the well-known HPO method Successive Halving (SHA), called Incremental Successive Halving (iSHA), aiming to improve its efficiency when the max size hyperparameter R of SHA needs to be increased post-hoc. We derived theoretical guarantees on the quality of the final choice, as well as on the saved budget, when a previous SHA run is continued. Furthermore, we provide the first theoretical analysis of asynchronous SHA, emphasizing the price that needs to be paid for the asynchronous promotions. In an empirical study, we also find that iSHA yields results similar to the much more expensive baseline variant of SHA and often better results than the current state-of-art among the asynchronous variants of SHA. In fact, our approach only requires the budget of the sole run with the increased max size.

It is worth noting that we considered a synchronous scenario to have a fair comparison with our proposed method. When a (GPU) cluster is available, the asynchronous nature of both ASHA and PASHA might lead to a parallelization speedup for HPO. However, for many practitioners who do not have such a cluster available, but for instance only one GPU, the speedup of an asynchronous approach is lost. In such a case, a more reliable, incremental and synchronous method such as iSHA is exactly the desired approach.

In future work, we plan to combine our SHA extensions with more sophisticated strategies for sampling hyperparameter configurations, as for example done by [Awad *et al.*, 2021] or [Falkner *et al.*, 2018] and HyperJump, to improve iHB's efficacy and efficiency even further. Another interesting avenue of future research is outlined by PriorBand, where a prior distribution is incorporated for sampling new hyperparameter configurations [Mallik *et al.*, 2023].

Ethical Statement

There are no ethical issues.

Acknowledgments

This work was partially supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

Contribution Statement

Jasmin Brandt and Marcel Wever had an equal contribution on this paper. While Jasmin Brandt was responsable for the theoretical parts, Marcel Wever did the empirical analysis. All authors were involved in devoloping the ideas and writing the paper draft in multiple iterations.

References

[Abbasi-Yadkori *et al.*, 2018] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, Alan Malek, and Michal Valko. Best of Both Worlds: Stochastic & Adversarial Best-arm Identification. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 918–949. PMLR, 06–09 Jul 2018.

[Awad *et al.*, 2021] Noor H. Awad, Neeratyoy Mallik, and Frank Hutter. DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 2147–2153, 2021.

[Azizi *et al.*, 2022] Mohammad Javad Azizi, Branislav Kveton, and Mohammad Ghavamzadeh. Fixed-Budget Best-Arm Identification in Structured Bandits. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 2798–2804. ijcai.org, 2022.

[Bergstra *et al.*, 2011] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[Bischl *et al.*, 2023] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13, 2023.

[Bohdal *et al.*, 2023] Ondrej Bohdal, Lukas Balles, Martin Wistuba, Beyza Ermis, Cédric Archambeau, and Giovanni Zappella. PASHA: Efficient HPO and NAS with Progressive Resource Allocation. In *The 11th International Conference on Learning Representations*, 2023.

[Brandt *et al.*, 2022] 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.

[Brandt *et al.*, 2023] Jasmin Brandt, Elias Schede, Björn Haddenhorst, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. AC-Band: A Combinatorial Bandit-Based Approach to Algorithm Configuration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):12355–12363, 2023.

[Carpentier and Valko, 2015] Alexandra Carpentier and Michal Valko. Simple regret for infinitely many armed bandits. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1133–1141. JMLR.org, 2015.

[Falkner *et al.*, 2018] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1436–1445. PMLR, 2018.

[Feurer and Hutter, 2019] Matthias Feurer and Frank Hutter. Hyperparameter Optimization. In *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 3–33. Springer, 2019.

[Frazier, 2018] Peter I. Frazier. A Tutorial on Bayesian Optimization. *CoRR*, abs/1807.02811, 2018.

[Gong and Sellke, 2023] Xiao-Yue Gong and Mark Sellke. Asymptotically Optimal Pure Exploration for Infinite-Armed Bandits. *CoRR*, abs/2306.01995, 2023.

[Hutter *et al.*, 2011] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization - 5th International Conference*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2011.

[Jamieson and Talwalkar, 2016a] Kevin Jamieson and Ameet Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248. PMLR, 2016.

[Jamieson and Talwalkar, 2016b] Kevin G. Jamieson and Ameet Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 240–248, 2016.

[Karnin *et al.*, 2013] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost Optimal Exploration in Multi-Armed Bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR, 2013.

[Kato *et al.*, 2022] Masahiro Kato, Kaito Ariu, Masaaki Imaizumi, Masatoshi Uehara, Masahiro Nomura, and Chao Qin. Optimal Fixed-Budget Best Arm Identification using the Augmented Inverse Probability Weighting Estimator in Two-Armed Gaussian Bandits with Unknown Variances. *CoRR*, abs/2201.04469, 2022.

[Lattimore and Szepesvári, 2020] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[Li *et al.*, 2018] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[Li *et al.*, 2020] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A System for Massively Parallel Hyperparameter Tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020.

[Mallik *et al.*, 2023] Neeratyoy Mallik, Edward Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning. *arXiv preprint arXiv:2306.12370*, 2023.

[Mendes *et al.*, 2021] Pedro Mendes, Maria Casimiro, and Paolo Romano. HyperJump: Accelerating HyperBand via Risk Modelling. *arXiv preprint arXiv:2108.02479*, 2021.

[Pfisterer *et al.*, 2022] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. YAHPO Gym - An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization. In *International Conference on Automated Machine Learning, AutoML 2022*, volume 188 of *Proceedings of Machine Learning Research*, pages 3/1–39. PMLR, 2022.

[Rice, 1976] John R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65–118, 1976.

[Shen, 2019] Cong Shen. Universal Best Arm Identification. *IEEE Trans. Signal Process.*, 67(17):4464–4478, 2019.

[Tornede *et al.*, 2023] Tanja Tornede, Alexander Tornede, Jonas Hanselle, Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Towards green automated machine learning: Status quo and future directions. *Journal of Artificial Intelligence Research*, 77:427–457, 2023.

[Yao, 1977] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 222–227, 1977.

8 Conclusion and Outlook

Our general framework proposed in Chapter 5 is applicable to many different scenarios. For example, it can handle different set sizes in Combinatorial Bandits, including the special case of Dueling Bandits and different kinds of feedback like numerical rewards or preference-based winner information either sampled by a stochastic process or of a non-stochastic kind. In addition, it works for different aggregation functions for the observations like the arithmetical mean, but also, for example, risk measures. The user does not have to choose a different method for different kinds of problems anymore, but can simply always use the same approach. Due to our derived theoretical guarantees about the arm that is returned by the algorithm, the users can trust our method and its returned solutions. Since the derived sufficient budget depends on parameters that are sometimes unknown in practice, like the suboptimality gap between the arms, we have shown in the experiments, that the algorithm works well anyway for different values of the available budget.

Extending these results to the setting of Algorithm Configuration in Chapter 6 also allowed us to transfer the theoretical guarantees to this scenario. The value of theoretical worst-case guarantees is often underestimated in this field and thus, there are not many existing AC approaches that come with theoretical analyses. However, the user does not take any risk of getting a suboptimal parameter configuration when using a method with a proven quality guarantee about the returned parameter configuration. In addition to the trustworthiness the user gains by adhering to worst-case guarantees, the Configurators are also proven to be resource-efficient if the sufficient budget (almost) matches the necessary budget to identify the best configuration with high probability. Due to the lack of research in the area of theoretically grounded Algorithm Configurators, there is still a performance gap to heuristic Algorithm Configurators, that we narrowed down since our algorithm returns a well-performing solution much faster than the state-of-the-art methods.

Finally, we enabled a much more resource-efficient and thus also more sustainable Hyperparameter Optimization by enhancing the entire typical workflow by a sophisticated modification of the widely used HYPERBAND algorithm and its Multi-Armed Bandit subroutine SUCCESSIVE HALVING. With our modifications, both are able to reuse already observed information from previous runs, need only a small amount of new samples they can compare to the previously collected information, and still keep the theoretical guarantees they had when rerunning from scratch with a much greater amount of newly considered samples. We extended all existing guarantees to the resource-efficient extensions of the methods and confirmed these results in an experimental study.

8.1 Future Research Directions

In future research, it would be interesting to study specific instantiations of our Combinatorial Bandit approach in more detail. For example, an interesting question is whether we can derive some specific theoretical guarantees for choosing other aggregations functions of the observations than the arithmetic mean, e.g. some risk measures like in [CMZ18]. In addition, the portions of arms that are discarded in each iteration stay fixed for a whole run at the moment, but maybe it would make sense to adopt a more aggressive elimination strategy in the early stages — when a larger number of arms are being tested, and get more careful when eliminating arms in the last iterations where we have only arms left that are at least better than a lot of already discarded ones.

Even if our theoretical Algorithm Configurator AC-BAND makes a huge step to close the performance gap to heuristic Algorithm Configurators, this gap still exists. Here, the obvious goal is to narrow it further by adapting some heuristic strategies such that under specific assumptions some theoretical guarantees can be derived for them. A promising direction would be the creation of new parameter configuration samples. To ensure enough exploration, the theoretical methods like AC-BAND still sample the considered configurations uniformly at random from the space of parameter configurations. However, in practice also approaches like a local neighborhood search like [Hut+09] or evolutionary methods like [AST09] seem to perform well, even if we usually cannot assume a continuity in the space of parameter configurations. In this regard, it is worth thinking about appropriate, but realistic assumptions on the space to create promising parameter configurations, such that we can prove a theoretical guarantee about the quality of the returned configuration.

The same improvement could be possible for our proposed Hyperparameter Optimization method ISHA. Also here, a more sophisticated hyperparameter selection strategy than random sampling will likely massively improve the performance in the regard of necessary samples that must be evaluated to guarantee a sufficiently well-performing returned configuration. In addition, we can think of combining our method with some already existing improvements of HYPERBAND like HYPERJUMP [Men+23] for even more efficiency in the hyperparameter configuration evaluations.

Bibliography

[Abb+18a] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, Alan Malek, and Michal Valko. “Best of both worlds: Stochastic & adversarial best-arm identification”. In: *Proceedings of the 31st Conference On Learning Theory*. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 918–949 (cit. on p. 33).

[Abb+18b] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, Alan Malek, and Michal Valko. “Best of both worlds: Stochastic & adversarial best-arm identification”. In: *Proceedings of the 31st Conference On Learning Theory (COLT)*. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 918–949 (cit. on pp. 41, 43).

[AJA20] Arpit Agarwal, Nicholas Johnson, and Shivani Agarwal. “Choice Bandits”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. Curran Associates, Inc., 2020, pp. 18399–18410 (cit. on p. 29).

[Ans+15] 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. AAAI Press, 2015, pp. 733–739 (cit. on p. 10).

[AST09] 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*. Springer Berlin Heidelberg, 2009, pp. 142–157 (cit. on pp. 10, 80).

[ABL11] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. “Minimax Policies for Combinatorial Prediction Games”. In: *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*. Vol. 19. Proceedings of Machine Learning Research. PMLR, 2011, pp. 107–132 (cit. on p. 28).

[ABM10] Jean-Yves Audibert, Sébastien Bubeck, and Remi Munos. “Best Arm Identification in Multi-Armed Bandits”. In: 2010, pp. 41–53 (cit. on p. 2).

[AMH21] Noor Awad, Neeratyoy Mallik, and Frank Hutter. “DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 2147–2153 (cit. on pp. 41, 43).

[AKG21] Javad Azizi, Branislav Kveton, and Mohammad Ghavamzadeh. “Fixed-Budget Best-Arm Identification in Structured Bandits”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2021 (cit. on pp. 41, 43).

[Ben+21] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. “Preference-based online learning with dueling bandits: a survey”. In: *Journal of Machine Learning Research* 22.1 (2021) (cit. on p. 23).

[BHH24] Viktor Bengs, Björn Haddenhorst, and Eyke Hüllermeier. “Identifying Copeland Winners in Dueling Bandits with Indifferences”. In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 238. Proceedings of Machine Learning Research. PMLR, 2024, pp. 226–234 (cit. on p. 23).

[BH20] Viktor Bengs and Eyke Hüllermeier. “Preselection Bandits”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 778–787 (cit. on pp. 29, 30).

[BB12] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305 (cit. on pp. 14, 40, 42).

[Boh+23] Ondrej Bohdal, Lukas Balles, Martin Wistuba, et al. “PASHA: Efficient HPO and NAS with Progressive Resource Allocation”. In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2023 (cit. on pp. 41, 43).

[BT52] Ralph Allan Bradley and Milton E. Terry. “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons”. In: *Biometrika* 39.3/4 (1952), pp. 324–345 (cit. on p. 29).

[Bro+16] Brian Brost, Yevgeny Seldin, Ingemar J. Cox, and Christina Lioma. “Multi-Dueling Bandits and Their Application to Online Ranker Evaluation”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. Association for Computing Machinery, 2016, pp. 2161–2166 (cit. on p. 31).

[Bub12] Sébastien Bubeck. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Jan. 2012 (cit. on pp. 20, 21).

[BMS09] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. “Pure Exploration in Multi-armed Bandits Problems”. In: *Algorithmic Learning Theory*. Springer Berlin Heidelberg, 2009, pp. 23–37 (cit. on p. 18).

[BS12] Sébastien Bubeck and Aleksandrs Slivkins. “The Best of Both Worlds: Stochastic and Adversarial Bandits”. In: *Proceedings of the 25th Annual Conference on Learning Theory*. Vol. 23. Proceedings of Machine Learning Research. PMLR, 2012, pp. 42.1–42.23 (cit. on pp. 33, 36, 37).

[CL59] Violet R. Cane and R. Duncan Luce. “Individual Choice Behavior: A Theoretical Analysis.” In: 1959 (cit. on p. 29).

[CV15] Alexandra Carpentier and Michal Valko. “Simple regret for infinitely many armed bandits”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 1133–1141 (cit. on pp. 41, 43).

[CMZ18] Asaf Cassel, Shie Mannor, and Assaf Zeevi. “A General Approach to Multi-Armed Bandits Under Risk Criteria”. In: *Proceedings of the 31st Conference On Learning Theory (COLT)*. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1295–1306 (cit. on p. 80).

[CF17] Bangrui Chen and Peter I. Frazier. “Dueling Bandits with Weak Regret”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 731–739 (cit. on p. 27).

[CWY13] Wei Chen, Yajun Wang, and Yang Yuan. “Combinatorial Multi-Armed Bandit: General Framework and Applications”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. Vol. 28. Proceedings of Machine Learning Research 1. PMLR, 2013, pp. 151–159 (cit. on p. 28).

[CLM18] Xi Chen, Yuanzhi Li, and Jieming Mao. “A nearly instance optimal algorithm for top-k ranking under the multinomial logit model”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2018, pp. 2504–2522 (cit. on p. 30).

[DKC20] Yihan Du, Yuko Kuroki, and Wei Chen. “Combinatorial Pure Exploration with Full-Bandit or Partial Linear Feedback”. In: *AAAI Conference on Artificial Intelligence*. 2020 (cit. on pp. 36, 37).

[ES04] Niklas Eén and Niklas Sörensson. “An Extensible SAT-solver”. In: *Theory and Applications of Satisfiability Testing*. Springer Berlin Heidelberg, 2004, pp. 502–518 (cit. on p. 5).

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter. “BOHB: Robust and Efficient Hyperparameter Optimization at Scale”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1437–1446 (cit. on pp. 41, 43).

[GUC15] Pratik Gajane, Tanguy Urvoy, and Fabrice Clérot. “A Relative Exponential Weighing Algorithm for Adversarial Utility-based Dueling Bandits”. In: *International Conference on Machine Learning (ICML)*. 2015 (cit. on pp. 36, 37).

[GK16] Aurélien Garivier and Emilie Kaufmann. “Optimal Best Arm Identification with Fixed Confidence”. In: *Proceedings of the 29th Conference on Learning Theory, COLT*. Vol. 49. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 998–1027 (cit. on p. 20).

[GLR23] Devon R. Graham, Kevin Leyton-Brown, and Tim Roughgarden. “Utilitarian Algorithm Configuration”. In: *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*. 2023 (cit. on p. 12).

[Had+21] Björn Haddenhorst, Viktor Bengs, Jasmin Brandt, and Eyke Hüllermeier. “Testification of Condorcet Winners in dueling bandits”. In: *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*. Vol. 161. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1195–1205 (cit. on p. 27).

[HHL11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization*. Springer Berlin Heidelberg, 2011, pp. 507–523 (cit. on p. 10).

[Hut+09] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. “ParamILS: an automatic algorithm configuration framework”. In: *Journal of Artificial Intelligence Research (JAIR)* 36.1 (2009), pp. 267–306 (cit. on pp. 10, 80).

[JT16] Kevin Jamieson and Ameet Talwalkar. “Non-stochastic Best Arm Identification and Hyperparameter Optimization”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 51. Proceedings of Machine Learning Research. PMLR, 2016, pp. 240–248 (cit. on pp. 20, 33, 36, 37, 40–42).

[Jou+21] Marc Jourdan, Mojmír Mutný, Johannes Kirschner, and Andreas Krause. “Efficient Pure Exploration for Combinatorial Bandits with Semi-Bandit Feedback”. In: *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*. Ed. by Vitaly Feldman, Katrina Ligett, and Sivan Sabato. Vol. 132. Proceedings of Machine Learning Research. PMLR, 2021, pp. 805–849 (cit. on pp. 36, 37).

[Kat+22] Masahiro Kato, Kaito Ariu, Masaaki Imaizumi, Masahiro Nomura, and Chao Qin. *Optimal Best Arm Identification in Two-Armed Bandits with a Fixed Budget under a Small Gap*. 2022. arXiv: 2201.04469 (cit. on pp. 41, 43).

[KKM12] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. “Thompson Sampling: An Asymptotically Optimal Finite-Time Analysis”. In: *Algorithmic Learning Theory*. Springer Berlin Heidelberg, 2012, pp. 199–213 (cit. on p. 22).

[KLL17] Robert Kleinberg, Kevin Leyton-Brown, and Brendan Lucier. “Efficiency Through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2017, pp. 2023–2031 (cit. on p. 11).

[Kle+19] 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 (cit. on pp. 37, 39).

[Kur+20] Yuko Kuroki, Liyuan Xu, Atsushi Miyauchi, Junya Honda, and Masashi Sugiyama. “Polynomial-Time Algorithms for Multiple-Arm Identification with Full-Bandit Feedback”. In: *Neural Computation* 32.9 (2020), pp. 1733–1773 (cit. on pp. 36, 37).

[Lar+07] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. “An Eempirical Evaluation of Deep Architectures on Problems with Many Factors of Variation”. In: *Proceedings of the 24th International Conference on Machine Learning (ICML)*. Association for Computing Machinery, 2007, pp. 473–480 (cit. on pp. 14, 40, 42).

[Li+20] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, et al. “A System for Massively Parallel Hyperparameter Tuning”. In: *Proceedings of Machine Learning and Systems*. Vol. 2. 2020, pp. 230–246 (cit. on pp. 41, 43).

[Li+17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816 (cit. on pp. 14, 38, 39, 41, 42).

[Lóp+11] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. “The irace Package: Iterated Racing for Automatic Algorithm Configuration”. In: *Operations Research Perspectives* 3 (Jan. 2011) (cit. on p. 10).

[Mas+20] Blake Mason, Lalit Jain, Ardhendu Tripathy, and Robert Nowak. “Finding All ϵ-Good Arms in Stochastic Bandits”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. Curran Associates, Inc., 2020, pp. 20707–20718 (cit. on p. 26).

[Men+23] Pedro Mendes, Maria Casimiro, Paolo Romano, and David Garlan. “HyperJump: Accelerating HyperBand via Risk Modelling”. In: *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI’23/IAAI’23/EAAI’23. AAAI Press, 2023 (cit. on pp. 41, 43, 80).

[Pla75] R. L. Plackett. “The Analysis of Permutations”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24.2 (1975), pp. 193–202 (cit. on p. 29).

[Rob52] Herbert Robbins. “Some aspects of the sequential design of experiments”. In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535 (cit. on p. 18).

[SG18] Aadirupa Saha and Aditya Gopalan. “PAC Battling Bandits in the Plackett-Luce Model”. In: *International Conference on Algorithmic Learning Theory*. 2018 (cit. on pp. 30, 31).

[SG19] Aadirupa Saha and Aditya Gopalan. “PAC Battling Bandits in the Plackett-Luce Model”. In: *Algorithmic Learning Theory ALT*. Vol. 98. Proceedings of Machine Learning Research. PMLR, 2019, pp. 700–737 (cit. on pp. 29, 35, 36).

[SKM20] Aadirupa Saha, Tomer Koren, and Y. Mansour. “Adversarial Dueling Bandits”. In: *International Conference on Machine Learning (ICML)*. 2020 (cit. on pp. 36, 37).

[Sch+22] Elias Schede, Jasmin Brandt, Alexander Tornede, et al. “A Survey of Methods for Automated Algorithm Configuration”. In: *Journal of Artificial Intelligence Research* 75 (2022) (cit. on p. 1).

[She19] Cong Shen. “Universal Best Arm Identification”. In: *IEEE Transactions on Signal Processing* 67.17 (2019), pp. 4464–4478 (cit. on pp. 41, 43).

[SJR16] Max Simchowitz, Kevin Jamieson, and Benjamin Recht. “Best-of-K-bandits”. In: *29th Annual Conference on Learning Theory*. Vol. 49. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1440–1489 (cit. on pp. 36, 37).

[Sui+17] Yanan Sui, Vincent Zhuang, Joel W- Burdick, and Yisong Yue. “Multi-Dueling Bandits with Dependent Arms”. In: 2017 (cit. on pp. 29, 31).

[Tho33] William R. Thompson. “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples”. In: *Biometrika* 25.3/4 (1933), pp. 285–294 (cit. on p. 18).

[WGS18] G. Weisz, A. György, and Cs. Szepesvári. “LeapsAndBounds: A Method for Approximately Optimal Algorithm Configuration”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. July 2018 (cit. on p. 12).

[WGS19] 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*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6707–6715 (cit. on pp. 8, 38, 39).

[Wei+20] Gellert Weisz, András György, Wei-I Lin, et al. “ImpatientCapsAndRuns: Approximately Optimal Algorithm Configuration from an Infinite Pool”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 17478–17488 (cit. on pp. 38, 39).

[Wu+19] Jia Wu, Xiu-Yun Chen, Hao Zhang, et al. “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization”. In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40 (cit. on pp. 14, 40, 42).

[Yue+09] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. “The K-armed Dueling Bandits Problem.” In: vol. 78. 2009 (cit. on p. 23).

[YJ09] Yisong Yue and Thorsten Joachims. “Interactively optimizing information retrieval systems as a dueling bandits problem”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. Association for Computing Machinery, 2009, pp. 1201–1208 (cit. on p. 23).

[Zog+14] Masrour Zoghi, Shimon Whiteson, Remi Munos, and Maarten Rijke. “Relative Upper Confidence Bound for the K-Armed Dueling Bandit Problem”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Vol. 32. Proceedings of Machine Learning Research 2. PMLR, 2014, pp. 10–18 (cit. on p. 26).

Webpages

[@] *Berkeley AI Course, lecture 11*. URL: https://ai.berkeley.edu/lecture_slides.html (cit. on p. 20).

A

Appendix to Finding Optimal Arms in Non-stochastic Combinatorial Bandits with Semi-bandit Feedback and Finite Budget

A List of Symbols

The following table contains a list of symbols that are frequently used in the main paper as well as in the following supplementary material.

Basics	
$\mathbb{1}\{\cdot\}$	indicator function
$\mathbb{N}$	set of natural numbers (without 0), i.e., $\mathbb{N} = \{1, 2, 3, \dots\}$
$\mathbb{R}$	set of real numbers
D	observation domain (categorical or numerical)
$\mathcal{A} = [n]$	set of arms
n	number of arms
k	maximal possible subset size
B	budget for the learner
$\mathcal{Q}_{\leq k}$	all subsets of $\mathcal{A}$ of size $\leq k$: $\{Q \subseteq \mathcal{A} \mid 2 \leq \lvert Q\rvert \leq k\}$
$\mathcal{Q}_{\leq k}(i)$	all subsets in $\mathcal{Q}_{\leq k}$ which contain arm i: $\{Q \in \mathcal{Q}_{\leq k} \mid i \in Q\}$
$\mathcal{Q}_{=k}$	all subsets of $\mathcal{A}$ of size k : $\{Q \subseteq \mathcal{A} \mid \lvert Q\rvert = k\}$
$\mathcal{Q}_{=k}(i)$	all subsets of $\mathcal{A}$ of size k which contain arm i : $\{Q \in \mathcal{Q}_{=k} \mid i \in Q\}$
$\mathbf{o}_Q(t)$	observed feedback vector by querying Q for the t-th time
Modelling related	
s	relevant statistic for the decision making process
$s_{i\lvert Q}(t)$	statistics for arm $i \in Q$ derived by the observed feedback at the t-th usage of query set Q
$\mathbf{s}_Q(t)$	vector of statistics for all arms in the query set Q after its t-th usage: $(s_{i\lvert Q})_{i\in Q}(t)$
$S_{i\lvert Q}$	limit of the statistics for arm i in query set Q: $\lim_{t\to\infty} s_{i\lvert Q}(t)$
i^*	best arm or generalized Condorcet winner: $\forall Q \in \mathcal{Q}_{\leq k}$ with $i^* \in Q$ it holds that $S_{i^*\lvert Q} > S_{j\lvert Q}$ for any $j \in Q \backslash \{i^*\}$
$s_i^{\mathcal{B}}(t)$	Borda score of arm i at time t: $\sum_{Q\in\mathcal{Q}_{=k}(i)} s_{i\lvert Q}(t) / \lvert\mathcal{Q}_{=k}(i)\rvert$
$S_i^{\mathcal{B}}$	limit Borda score of arm i: $\lim_{t\to\infty} s_i^{\mathcal{B}}(t)$
$i^*_{\mathcal{B}}$	generalized Borda winner: $i^*_{\mathcal{B}} \in \arg\max_{i\in\mathcal{A}} S_i^{\mathcal{B}}$
$n_Q(t)$	number of times query set Q was used until time t
$\gamma_{i\lvert Q}(t)$	point-wise smallest non-increasing function bounding the difference $\lvert s_{i\lvert Q}(t) - S_{i\lvert Q}\rvert$ (rate of convergence)
$\bar{\gamma}_Q(t)$	maximal $\gamma_{i\lvert Q}(t)$ over all $i \in Q$
$\bar{\gamma}(t)$	maximal $\gamma_Q(t)$ over all $Q \in \mathcal{Q}_{\leq k}$
$\gamma^{-1}_{i\lvert Q}(\alpha)$	quasi-inverse of $\gamma_{i\lvert Q}$: $\min\{t \in \mathbb{N} \mid \gamma_{i\lvert Q}(t) \leq \alpha\}$
$\bar{\gamma}^{-1}_Q(t)$	minimal $\gamma_{i\lvert Q}(t)$ over all $i \in Q$
$\bar{\gamma}^{-1}(t)$	minimal $\gamma_Q(t)$ over all $Q \in \mathcal{Q}_{\leq k}$
$\hat{\gamma}_i(t)$	rate of convergence of the Borda score for arm i: $\frac{1}{\lvert\mathcal{Q}_{=k}(i)\rvert} \sum_{Q\in\mathcal{Q}_{=k}(i)} \gamma_{i\lvert Q}(t)$
$\hat{\gamma}^{\max}_{i,j}(t)$	$\max\{\hat{\gamma}_i(t), \hat{\gamma}_j(t)\}$.
$\Delta_{i\lvert Q}$	gap of the limit statistic of arm $i \in Q$ to the limit statistic of the generalized Condorcet winner: $\lvert S_{i^*\lvert Q} - S_{i\lvert Q}\rvert$ for any $Q \in \mathcal{Q}_{\leq k}(i) \cap \mathcal{Q}_{\leq k}(i^*)$
$S_{(l)\lvert Q}$, $\Delta_{(l)\lvert Q}$	l-th order statistic of $\{S_{i\lvert Q}\}_{i\in Q}$ for $l \in \{1, 2, \dots, \lvert Q\rvert\}$ and its gap $\Delta_{(l)\lvert Q} = S_{i^*\lvert Q} - S_{(l)\lvert Q}$
Algorithm related	
f	function from $[k]$ to $[k]$ specifying the nature of the arm elimination strategy
$R, R^{\mathbb{A}}$	number of rounds of the learning algorithm ($\mathbb{A}$)
$P_r, P_r^{\mathbb{A}}$	number of partitions of the learning algorithm ($\mathbb{A}$) in round r
$\mathbb{A}_{r,j}$	j-th partition in round r
$\mathbb{A}_r(i^*)$	the partition in round r containing i^* (emptyset otherwise)
b_r	budget used in round r for a partition
$z_{\mathbb{A}}$	sufficient budget for learning algorithm $\mathbb{A}$ to return i^* (or $i^*_{\mathcal{B}}$ if $\mathbb{A}$ is RoundRobin)
RoundRobin	the naïve algorithm introduced in Section C
CSE	the generic *combinatorial successive elimination* algorithm (Algorithm 1)
CSWS	the *combinatorial successive winner stays* algorithm resulting by using $f(x) = 1$ in CSE
CSR	the *combinatorial successive rejects* algorithm resulting by using $f(x) = x - 1$ in CSE
CSH	the *combinatorial successive halving* algorithm resulting by using $f(x) = \lceil x/2 \rceil$ in CSE
SH	the *successive halving* algorithm for pure exploration settings in standard multi-armed bandits (cf. [25])
GBW	Generalized Borda winner
GCW	Generalized Condorcet winner

B Proofs for Section 3

In this section, we prove the general lower bounds on the necessary budget for identifying the generalized Condorcet winner (GCW), the generalized Borda winner (GBW) or the generalized Copeland winner (GCopeW). For this purpose, let us first fix some further notation. If Alg is a possibly probabilistic algorithm and $\mathbf{s}$ is fixed, we write $\mathrm{Alg}(\mathbf{s})$ for the output of Alg executed on the instance $\mathbf{s}$. We restrict ourselves only to algorithms whose output is solely determined by the sequence of observations it has received as well as the corresponding statistics. Moreover, for $Q \in \mathcal{Q}_{\leq k}$, we write $B_Q(\mathrm{Alg}, \mathbf{s}) \in \mathbb{N} \cup \{\infty\}$ for the number of times Alg queries Q when started on instance $\mathbf{s}$. Note that $\mathrm{Alg}(\mathbf{s})$ as well as $B_Q(\mathrm{Alg}, \mathbf{s})$ and $B(\mathrm{Alg}, \mathbf{s}) = \sum_{Q \in \mathcal{Q}_{\leq k}} B_Q(\mathrm{Alg}, \mathbf{s})$ are random variables, because they depend on the innate randomness of Alg.
Given $\mathbf{s}$, let us write $\mathrm{GCW}(\mathbf{s})$, $\mathrm{GBW}(\mathbf{s})$ and $\mathrm{GCopeW}(\mathbf{s})$ for the set of all GCWs, GBWs and GCopeWs of $\mathbf{s}$, respectively. In case $|\mathrm{GCW}(\mathbf{s})| = 1$, $|\mathrm{GBW}(\mathbf{s})| = 1$ resp. $|\mathrm{GCopeW}(\mathbf{s})| = 1$, with a slight abuse of notation, we may denote by $\mathrm{GCW}(\mathbf{s})$, $\mathrm{GBW}(\mathbf{s})$ resp. $\mathrm{GCopeW}(\mathbf{s})$ simply the only GCW, GBW resp. GCopeW of $\mathbf{s}$. Recall that the GCW, the GBWs and the GCopeWs of $\mathbf{s}$ only depend on the limits $\mathbf{S} = (S_{i|Q})_{Q \in \mathcal{Q}_{\leq k}, i \in Q}$ with $S_{i|Q} = \lim_{t \to \infty} s_{i|Q}(t)$.

Definition B.1. *Let* Alg *be a (possibly probabilistic) sequential algorithm.*

(i) Alg *solves* $\mathcal{P}_{\mathrm{GCW}}(\mathbf{S}, \boldsymbol{\gamma})$ *if* $\mathbb{P}(\mathrm{Alg}(\mathbf{s}) \in \mathrm{GCW}(\mathbf{s})) = 1$ *for any* $\mathbf{s}$ *in* $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

(ii) Alg *solves* $\mathcal{P}_{\mathrm{GBW}}(\mathbf{S}, \boldsymbol{\gamma})$ *if* $\mathbb{P}(\mathrm{Alg}(\mathbf{s}) \in \mathrm{GBW}(\mathbf{s})) = 1$ *for any* $\mathbf{s}$ *in* $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

(iii) Alg *solves* $\mathcal{P}_{\mathrm{GCopeW}}(\mathbf{S}, \boldsymbol{\gamma})$ *if* $\mathbb{P}(\mathrm{Alg}(\mathbf{s}) \in \mathrm{GCopeW}(\mathbf{s})) = 1$ *for any* $\mathbf{s}$ *in* $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

B.1 Proof of Theorem 3.1 (i): Lower Bound for GCW Identification

The proof of (i) in Theorem 3.1 is prepared with the next lemma.

Lemma B.2. *Let* Alg *be a deterministic solution to* $\mathcal{P}_{\mathrm{GCW}}(\mathbf{S}, \boldsymbol{\gamma})$ *and* $\mathbf{s}, \mathbf{s}' \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

(i) If $\mathrm{Alg}(\mathbf{s}) \neq \mathrm{Alg}(\mathbf{s}')$*, then*

$$\exists Q \in \mathcal{Q}_{\leq k}, i \in Q,\, t \in \{1, \dots, \min\{B_Q(\mathrm{Alg}, \mathbf{s}), B_Q(\mathrm{Alg}, \mathbf{s}')\}\} \,:\, s_{i|Q}(t) \neq s'_{i|Q}(t).$$

(ii) If $\mathbf{s}$ *and* $\mathbf{s}'$ *coincide on* $\{t < B'\}$ *and on* $\widetilde{\mathcal{Q}} \subseteq \mathcal{Q}_{\leq k}$ *in the sense that*

$$\forall Q \in \mathcal{Q}_{\leq k}, \forall i \in Q, \forall t < B' \,:\, s_{i|Q}(t) = s'_{i|Q}(t) \tag{1}$$

and

$$\forall Q \in \widetilde{\mathcal{Q}}, \forall i \in Q, \forall t \in \mathbb{N} \,:\, s_{i|Q}(t) = s'_{i|Q}(t), \tag{2}$$

then $\mathrm{Alg}(\mathbf{s}) \neq \mathrm{Alg}(\mathbf{s}')$ *implies*

$$\exists Q \in \mathcal{Q}_{\leq k} \setminus \widetilde{\mathcal{Q}} \,:\, \min\{B_Q(\mathrm{Alg}, \mathbf{s}), B_Q(\mathrm{Alg}, \mathbf{s}')\} \geq B'.$$

Proof. (i) To prove the contraposition, suppose that

$$\forall Q \in \mathcal{Q}_{\leq k}, i \in Q,\, t \in \{1, \dots, \min\{B_Q(\mathrm{Alg}, \mathbf{s}), B_Q(\mathrm{Alg}, \mathbf{s}')\}\} \,:\, s_{i|Q}(t) = s'_{i|Q}(t) \tag{3}$$

holds.
Claim 1: $B_Q(\mathrm{Alg}, \mathbf{s}) = B_Q(\mathrm{Alg}, \mathbf{s}')$ for any $\mathcal{Q} \in \mathcal{Q}_{\leq k}$.
Proof: Assume this was not the case. Let $Q \in \mathcal{Q}_{\leq k}$ be the first set, for which Alg exceeds its budget on one of $\mathbf{s}$, $\mathbf{s}'$ but does not reach it on the other instance, and suppose w.l.o.g. $B_Q(\mathrm{Alg}, \mathbf{s}) > B_Q(\mathrm{Alg}, \mathbf{s}')$. Since Alg has observed until this point exactly the same feedback on $\mathbf{s}$ as on $\mathbf{s}'$, this is a contradiction as Alg is deterministic. ■
Combining Claim 1 and (3) yields that Alg observes on $\mathbf{s}$ exactly the same feedback as on $\mathbf{s}'$ until its termination. Since Alg is deterministic, this implies $\mathrm{Alg}(\mathbf{s}) = \mathrm{Alg}(\mathbf{s}')$.

(ii) If $\mathrm{Alg}(\mathbf{s}) \neq \mathrm{Alg}(\mathbf{s}')$, then (i) together with (2) yields

$$\exists Q \in \mathcal{Q}_{\leq k} \setminus \widetilde{\mathcal{Q}}, i \in Q, t \leq \min\{B_Q(\mathrm{Alg}, \mathbf{s}), B_Q(\mathrm{Alg}, \mathbf{s}')\} \,:\, s_{i|Q}(t) \neq s'_{i|Q}(t),$$

and thus (1) implies

$$\exists Q \in \mathcal{Q}_{\leq k} \setminus \widetilde{\mathcal{Q}} : \min\{B_Q(\mathrm{Alg}, \mathbf{s}), B_Q(\mathrm{Alg}, \mathbf{s}')\} \geq B'.$$

□

Lemma B.2 is the main ingredient for the proof of Theorem 3.1, as we first analyze the lower bound for deterministic algorithms and then apply Yao's minimax principle [51] to infer the lower bound for any randomized algorithm.

Proof of Theorem 3.1 (i). We split the proof into two parts.

Part 1: The statement holds in case Alg **is a deterministic algorithm.**
Abbreviate $B' \coloneqq \min_{Q \in \mathcal{Q}_{\leq k}} \min_{j \in Q} \gamma_{j|Q}^{-1}\left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}\right)$. Fix a family $\{\pi_Q\}_{Q \in \mathcal{Q}_{\leq k}}$ of permutations $\pi_Q : Q \mapsto Q$ such that $S_{\pi_Q(1)|Q} = S_{(1)|Q}$ holds for any $Q \in \mathcal{Q}_{\leq k}(1)$, and define $\mathbf{s} = (s_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$ via

$$s_{i|Q}(t) \coloneqq \begin{cases} \frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2}, & \text{if } t < B', \\ S_{\pi_Q(i)|Q}, & \text{if } t \geq B'. \end{cases}$$

Regarding our assumption on $\mathbf{S}$, $\mathrm{GCW}(\mathbf{s}) = 1$ holds by construction. For $t < B' \leq \gamma_{i|Q}^{-1}\left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}\right)$, which implies $\gamma_{i|Q}(t) \geq \frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}$, we have due to $S_{(1)|Q} \geq S_{i|Q} \geq S_{(|Q|)|Q}$ the inequality

$$\begin{aligned} &\left|s_{i|Q}(t) - \lim_{t \to \infty} s_{i|Q}(t)\right| = \left|\frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2} - S_{i|Q}\right| \\ &\leq \max\left\{S_{(1)|Q} - \frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2}, \frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2} - S_{(|Q|)|Q}\right\} \\ &= \frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2} \\ &\leq \gamma_{i|Q}(t) \end{aligned}$$

for any $i \in Q$. This shows $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

For any $l \in \{2, \ldots, n\}$ define an instance $\mathbf{s}^l = (s^l_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$ such that $s^l_{\cdot|Q}(\cdot) = s_{\cdot|Q}(\cdot)$ for any $Q \in \mathcal{Q}_{\leq k}$ with $l \notin Q$ and

$$s^l_{i|Q}(t) \coloneqq \begin{cases} \frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2}, & \text{if } t < B', \\ S_{(1)|Q}, & \text{if } t \geq B' \text{ and } i = l, \\ S_{l|Q}, & \text{if } t \geq B' \text{ and } i = \mathrm{argmax}_{j \in Q} S_{j|Q} \\ s_{i|Q}(t), & \text{else,} \end{cases}$$

for all $Q \in \mathcal{Q}_{\leq k}(l)$, $i \in Q$ and $t \in \mathbb{N}$. According to its definition, we have $\mathrm{GCW}(\mathbf{s}^l) = l$, and similarly as above one may check $\mathbf{s}^l \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

Since Alg solves $\mathcal{P}_{\mathrm{GCW}}(\mathbf{S}, \boldsymbol{\gamma})$, it satisfies $\mathrm{Alg}(\mathbf{s}) = 1 \neq 2 = \mathrm{Alg}(\mathbf{s}^2)$. Regarding that $\mathbf{s}$ and $\mathbf{s}^2$ coincide on $\{t < B'\}$ and on $\{Q \in \mathcal{Q}_{\leq k} \,|\, 1 \notin Q \text{ or } 2 \notin Q\}$ in the sense of (1) and (2), Lemma B.2 (ii) assures the existence of some $Q_1 \in \mathcal{Q}_{\leq k}$ with $1 \in Q_1$ and $i_1 \coloneqq 2 \in Q_1$ such that $B_{Q_1}(\mathrm{Alg}, \mathbf{s}) \geq \min\{B_{Q_1}(\mathrm{Alg}, \mathbf{s}), B_{Q_1}(\mathrm{Alg}, \mathbf{s}^{i_1})\} \geq B'$. Let $F_1 \coloneqq [n] \setminus Q_1$ and fix an arbitrary $i_2 \in F_1$. Then, $\mathrm{Alg}(\mathbf{s}) = 1 \neq i_2 = \mathrm{Alg}(\mathbf{s}^{i_2})$ and since $\mathbf{s}$ and $\mathbf{s}^{i_2}$ coincide on $\{t < B'\}$ and $\{Q \in \mathcal{Q}_{\leq k} \,|\, i_2 \notin Q\}$, Lemma B.2 (ii) yields the existence of some $Q_2 \in \mathcal{Q}_{\leq k}$ with $i_2 \in Q_2$ such that $B_{Q_2}(\mathrm{Alg}, \mathbf{s}) \geq \min\{B_{Q_2}(\mathrm{Alg}, \mathbf{s}), B_{Q_2}(\mathrm{Alg}, \mathbf{s}^{i_2})\} \geq B'$. From $i_2 \in F_1 = [n] \setminus Q_1$ and $i_2 \in Q_2$ we infer $Q_1 \neq Q_2$. With this, we define $F_2 \coloneqq F_1 \setminus Q_2 = [n] \setminus (Q_1 \cup Q_2)$.

Inductively, whenever $F_l \neq \emptyset$, we may select an element $i_{l+1} \in F_l$ and infer from Lemma B.2 (ii), due to $\mathrm{Alg}(\mathbf{s}) = 1 \neq i_{l+1} = \mathrm{Alg}(\mathbf{s}^{i_{l+1}})$ and the similarity of $\mathbf{s}$ and $\mathbf{s}^{i_{l+1}}$ on $\{t < B'\}$ and $\{Q \in \mathcal{Q}_{\leq k} | i_{l+1} \notin Q\}$, the existence of a set $Q_{l+1} \in \mathcal{Q}_{\leq k}$ with $i_{l+1} \in Q_{l+1}$ such that $B_{Q_{l+1}}(\mathrm{Alg}, \mathbf{s}) \geq B'$, and define $F_{l+1} \coloneqq F_l \setminus Q_{l+1}$. Then, $i_{l+1} \in F_l = [n] \setminus (Q_1 \cup \cdots \cup Q_l)$

and $i_{l+1} \in Q_{l+1}$ assure $Q_{l+1} \notin \{Q_1, \dots, Q_l\}$. This procedure terminates at the smallest l' such that $F_{l'} = \emptyset$, and $Q_1, \dots, Q_{l'}$ are distinct. Regarding that $|F_{l+1}| - |F_l| \leq |Q_l| \leq k$ for all $l \in \{1, \dots, l'-1\}$, we have $l' \geq \lceil \frac{n}{k} \rceil$. Consequently,

$$B(\mathrm{Alg}, \mathbf{s}) \geq \sum_{l=1}^{l'} B_{Q_l}(\mathrm{Alg}, \mathbf{s}) \geq \left\lceil \frac{n}{k} \right\rceil B'$$

holds, which shows the claim for deterministic algorithms with regard to the definition of B'.

Part 2: The statement holds for arbitrary Alg**.**
Let $\mathfrak{A}$ be the set of all deterministic algorithms[3] and $\mathbf{s}$ be the instance from the first part. Write $\delta_{\mathbf{s}}$ for the probability distribution on $\{\mathbf{s}\}$, which assigns $\mathbf{s}$ probability one, i.e., the Dirac measure on $\mathbf{s}$. Note that for any randomized algorithm Alg there exists a probability distribution P on $\mathfrak{A}$ such that $\mathrm{Alg} \sim P$. By applying Yao's minimax principle [51] and using part one we conclude

$$\begin{aligned} \mathbb{E}[B(\mathrm{Alg}, \mathbf{s})] = \mathbb{E}_{\mathrm{Alg}' \sim P}[B(\mathrm{Alg}', \mathbf{s})] &\geq \inf\nolimits_{\mathrm{Alg} \in \mathfrak{A}} \mathbb{E}_{\mathbf{s}' \sim \delta_{\mathbf{s}}}[B(\mathrm{Alg}, \mathbf{s}')] \\ &= \inf\nolimits_{\mathrm{Alg} \in \mathfrak{A}} B(\mathrm{Alg}, \mathbf{s}) \geq \left\lceil \frac{n}{k} \right\rceil B', \end{aligned}$$

where B' is as in part one. □

Remark B.3. *(i) The above proof reveals even a stronger version of Theorem 3.1 (i). Indeed, in the proof we explicitly construct n distinct instances $\mathbf{s}^1 := \mathbf{s}, \dots, \mathbf{s}^n \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$ with $\mathrm{GCW}(\mathbf{s}^l) = l$ for all $l \in [n]$, and in fact show: Any (possibly random) algorithm* Alg*, which is able to correctly identify the best arm for any $\mathbf{s}' \in \{\mathbf{s}_1, \dots, \mathbf{s}_n\}$ (i.e.,* Alg *does not necessarily have to solve $\mathcal{P}_{\mathrm{GCW}}(\mathbf{S}, \boldsymbol{\gamma})$) fulfills*

$$\mathbb{E}\left[B(\mathrm{Alg}, \mathbf{s})\right] \geq \left\lceil \frac{n}{k} \right\rceil \min_{Q \in \mathcal{Q}_{\leq k}} \min_{j \in Q} \gamma_{j|Q}^{-1} \left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2} \right).$$

(ii) Condition (iii) in the definition of $\mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$ assures that the term $S_{(1)|Q}$ resp. $S_{(|Q|)|Q}$ in our lower bound from Theorem 3.1 coincides with $S'_{(1)|Q}$ resp. $S'_{(|Q|)|Q}$, when $S'_{i|Q} := \lim_{t \to \infty} s_{i|Q}(t)$ for $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$.

B.2 Proof of Theorem 3.1 (ii): Lower Bound for GBW Identification

Recall that $\mathrm{GBW}(\mathbf{s})$ is the set of elements $i \in [n]$, for which the limits $S_{i|Q} = \lim_{t \to \infty} s_{i|Q}(t)$ have the highest Borda score

$$S_i^{\mathcal{B}} = \frac{\sum_{Q \in \mathcal{Q}_{=k}(i)} S_{i|Q}}{|\mathcal{Q}_{=k}(i)|} = \frac{\sum_{Q \in \mathcal{Q}_{=k}(i)} S_{i|Q}}{\binom{n-1}{k-1}}.$$

We call $\mathbf{S} = (S_{i|Q})_{Q \in \mathcal{Q}_{\leq k}, i \in Q}$ **homogeneous** if $(S_{(1)|Q}, \dots, S_{(|Q|)|Q})$ does not depend on Q. Thus, if $\mathbf{S}$ is homogeneous, we may simply write $S_{(l)}$ for $S_{(l)|Q}$ for any $Q \in \mathcal{Q}_{=k}$.

The next two lemmata serves as a preparation for the proof of (ii) and (iii) in Theorem 3.1.

Lemma B.4. *For any $\mathcal{W} \subseteq \mathcal{Q}_{=k}$ we have $\sum_{j=1}^{n} |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| = k|\mathcal{W}|$.*

Proof of Lemma B.4. Let $\mathcal{W} \subseteq \mathcal{Q}_{=k}$ be fixed. For any $Q = \{i_1, \dots, i_k\} \in \mathcal{Q}_{=k} \cap \mathcal{W}$ we have that $Q \in \mathcal{Q}_{=k}(i_l) \cap \mathcal{W}$ for any $l \in [k]$, whereas $Q \notin \mathcal{Q}_{=k}(j) \cap \mathcal{W}$ for any $j \in [n] \setminus \{i_1, \dots, i_k\}$. Hence,

$$\sum\nolimits_{j=1}^{n} |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| = k \left| \bigcup\nolimits_{j=1}^{n} (\mathcal{Q}_{=k}(j) \cap \mathcal{W}) \right| = k \left| \left(\bigcup\nolimits_{j=1}^{n} \mathcal{Q}_{=k}(j) \right) \cap \mathcal{W} \right| = k|\mathcal{W}|.$$

□

Lemma B.5. *For any $\mathcal{W}' \subseteq \mathcal{Q}_{=k}$ and $\mathcal{W} := \mathcal{Q}_{=k} \setminus \mathcal{W}'$ with $|\mathcal{W}'| < \frac{(1-1/n)k}{k+n-2} \binom{n}{k}$ there exists $j \in [n] \setminus \{1\}$ with $|\mathcal{Q}_{=k}(j) \cap \mathcal{W}| > |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'|$.*

[3] At any time $t \in \mathbb{N}$, a deterministic algorithm $\mathrm{Alg} \in \mathfrak{A}$ may either make a query $Q \in \mathcal{Q}_{\leq k}$ or terminate with a decision $\mathrm{X} \in \{1, \dots, n\}$. Thus, $\mathfrak{A}$ is a countable set.

Proof of Lemma B.5. For $j \in [n] \setminus \{1\}$ abbreviate $a_j \coloneqq |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| - |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'|$. Due to

$$\begin{aligned}
|\mathcal{W}| &= \binom{n}{k} - |\mathcal{W}'| \\
&> \binom{n}{k} - \left(1 - \frac{1}{n}\right) \frac{k}{k+n-2} \binom{n}{k} \\
&= \binom{n}{k} - \frac{k\binom{n}{k} - \frac{k}{n}\binom{n}{k}}{k+n-2} \\
&= \frac{1}{k+n-2} \left(\binom{n-1}{k-1} + (n-2)\binom{n}{k} \right)
\end{aligned}$$

we have

$$k|\mathcal{W}| - \binom{n-1}{k-1} - (n-2)\left(\binom{n}{k} - |\mathcal{W}|\right) > 0.$$

By using Lemma B.4 and the fact that $(\mathcal{W} \cap \mathcal{Q}_{=k}(1)) \cup (\mathcal{W}' \cap \mathcal{Q}_{=k}(1)) = \mathcal{Q}_{=k}(1)$ is a disjoint union, we obtain

$$\begin{aligned}
&\textstyle\sum_{j \neq 1} a_j = \sum_{j \neq 1} |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| - (n-1)|\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| \\
&= \textstyle\sum_{j \in [n]} |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| - |\mathcal{Q}_{=k}(1) \cap \mathcal{W}| - |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| - (n-2)|\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| \\
&= k|\mathcal{W}| - |\mathcal{Q}_{=k}(1)| - (n-2)|\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| \\
&\geq k|\mathcal{W}| - \binom{n-1}{k-1} - (n-2)|\mathcal{W}'| \\
&= k|\mathcal{W}| - \binom{n-1}{k-1} - (n-2)\left(\binom{n}{k} - |\mathcal{W}|\right) > 0.
\end{aligned}$$

Consequently, there exists $j \in [n] \setminus \{1\}$ with $a_j > 0$. □

Proof of Theorem 3.1 (ii). Similarly as in the proof of Theorem 3.1 (i), we proceed in two steps.

Part 1: The statement holds in case Alg **is deterministic.**
Abbreviate $B' \coloneqq \overline{\gamma}^{-1}\left(\frac{S_{(1)} - S_{(|Q|)}}{2}\right)$ and fix a family of permutations $(\pi_Q)_{Q \in \mathcal{Q}_{\leq k}}$ with $S_{(1)|Q} = S_{\pi_Q(1)|Q}$ for all $Q \in \mathcal{Q}_{\leq k}(1)$. Exactly as in the proof of Theorem 3.1 (i), we define $\mathbf{s} = (s_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$ via

$$s_{i|Q}(t) \coloneqq \begin{cases} \frac{S_{(1)|Q} + S_{(|Q|)|Q}}{2}, & \text{if } t < B' \\ S_{\pi_Q(i)|Q}, & \text{if } t \geq B'. \end{cases}$$

In the proof of Theorem 3.1 (i) we have already verified $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$. For any $j \in \{2, \dots, m\}$ and $Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)$ we have $S_{1|Q} > S_{j|Q}$, and using that $|\mathcal{Q}_{=k}(i') \setminus \mathcal{Q}_{=k}(j')|$ is the same for every distinct $i', j' \in [n]$ we thus have

$$\begin{aligned}
\textstyle\sum_{Q \in \mathcal{Q}_{=k}(1)} S_{1|Q} &= \textstyle\sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)} S_{1|Q} + S_{(1)} \cdot |\mathcal{Q}_{=k}(1) \setminus \mathcal{Q}_{=k}(j)| \\
&> \textstyle\sum_{Q \in \mathcal{Q}_{=k}(j) \cap \mathcal{Q}_{=k}(1)} S_{j|Q} + S_{(1)} \cdot |\mathcal{Q}_{=k}(j) \setminus \mathcal{Q}_{=k}(1)| \\
&> \textstyle\sum_{Q \in \mathcal{Q}_{=k}(j)} S_{j|Q}.
\end{aligned}$$

As $|\mathcal{Q}_{=k}(1)| = |\mathcal{Q}_{=k}(j)|$, this shows $\mathrm{GBW}(\mathbf{s}) = 1$.

In the following, we will show that

$$\mathcal{W}' \coloneqq \{Q \in \mathcal{Q}_{=k} \,:\, \text{Alg started on } \mathbf{s} \text{ queries } Q \text{ at least } B' \text{ times}\}$$

contains at least $\frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ elements. For this, let us assume on the contrary $|\mathcal{W}'| < \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ and write $\mathcal{W} \coloneqq \mathcal{Q}_{=k} \setminus \mathcal{W}'$. Lemma B.5 allows us to fix a $j \in [n] \setminus \{1\}$ with $|\mathcal{Q}_{=k}(j) \cap \mathcal{W}| >$

$|\mathcal{Q}_{=k}(1) \cap \mathcal{W}'|$. Now, define $\mathbf{s}' = (s'_{i|Q}(t))_{Q\in\mathcal{Q}_{\leq k}, i\in Q, t\in\mathbb{N}}$ via $s'_{\cdot|Q}(\cdot) = s_{\cdot|Q}(\cdot)$ for any $Q \in (\mathcal{Q}_{\leq k} \setminus (\mathcal{Q}_{=k}(1) \cup \mathcal{Q}_{=k}(j))) \cup \mathcal{W}'$ and[4]

$$s'_{i|Q}(t) \coloneqq \begin{cases} s_{i|Q}(t), & \text{if } t < B' \text{ or } \{1,j\} \not\subseteq Q, \\ S_{(1)}, & \text{if } i = j \in Q \text{ and } t \geq B', \\ S_{(|Q|)}, & \text{if } i = 1 \in Q \text{ and } t \geq B', \\ S_{1|Q}, & \text{if } t \geq B', i = \arg\min_{l'\in Q} S_{l'|Q} \text{ and } 1 \in Q \not\ni j, \\ S_{j|Q}, & \text{if } t \geq B', i = \operatorname{argmax}_{l'\in Q} S_{l'|Q} \text{ and } j \in Q \not\ni 1, \\ S_{i|Q}, & \text{otherwise,} \end{cases}$$

for $Q \in (\mathcal{Q}_{=k}(1) \cup \mathcal{Q}_{=k}(j)) \cap \mathcal{W}$. Similarly as for $\mathbf{s}$, we see $\mathbf{s}' \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$. The corresponding limit values $S'_{i|Q} = \lim_{t\to\infty} s'_{i|Q}(t)$ fulfill

$$\forall Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W} : S'_{1|Q} = S_{(|Q|)} \quad \text{and} \quad \forall Q \in \mathcal{Q}_{=k}(j) \cap \mathcal{W} : S'_{j|Q} = S_{(1)},$$

and trivially also $S_{(|Q|)} \leq S'_{i|Q} \leq S_{(1)}$ for any $Q \in \mathcal{Q}_{=k}, i \in Q$. Therefore, by choice of j, the corresponding Borda scores $(S')_i^{\mathcal{B}}$ for $\mathbf{s}'$ fulfill

$$\begin{aligned} \binom{n-1}{k-1}(S')_1^{\mathcal{B}} &= \sum\nolimits_{Q\in\mathcal{Q}_{=k}(1)} S'_{1|Q} = \sum\nolimits_{Q\in\mathcal{Q}_{=k}(1)\cap\mathcal{W}'} S_{(1)} + \sum\nolimits_{Q\in\mathcal{Q}_{=k}(1)\cap\mathcal{W}} S_{(|Q|)} \\ &= |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| \cdot S_{(1)} + |\mathcal{Q}_{=k}(1) \cap \mathcal{W}| \cdot S_{(|Q|)} \\ &< |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| \cdot S_{(1)} + |\mathcal{Q}_{=k}(j) \cap \mathcal{W}'| \cdot S_{(|Q|)} \\ &\leq \sum\nolimits_{Q\in\mathcal{Q}_{=k}(j)} S'_{j|Q} = \binom{n-1}{k-1}(S')_j^{\mathcal{B}}, \end{aligned}$$

where we have used that $|\mathcal{Q}_{=k}(1)\cap\mathcal{W}'|+|\mathcal{Q}_{=k}(1)\cap\mathcal{W}| = |\mathcal{Q}_{=k}(1)| = |\mathcal{Q}_{=k}(j)\cap\mathcal{W}'|+|\mathcal{Q}_{=k}(j)\cap\mathcal{W}|$. This show $1 \notin \mathrm{GBW}(\mathbf{s}')$. But since $s_{\cdot|\cdot}(\cdot) = s'_{\cdot|\cdot}(\cdot)$ holds on $\{t < B'\}$ as well as on $\mathcal{W}'$, Alg observes for $\mathbf{s}$ until termination exactly the same feedback as for $\mathbf{s}'$. Consequently, it outputs for both instances the same decision. Since $\mathrm{GBW}(\mathbf{s}) = 1 \notin \mathrm{GBW}(\mathbf{s}')$, it makes on at least one of the instances a mistake, which contradicts the correctness of Alg.
Thus, $|\mathcal{W}'| \geq \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ has to hold and we conclude

$$B(\mathrm{Alg}, \mathbf{s}) \geq \sum\nolimits_{Q\in\mathcal{W}'} B_Q(\mathrm{Alg}, \mathbf{s}) \geq |\mathcal{W}'| \cdot B' \geq \left(1 - \frac{1}{n}\right) \frac{k}{k+n-2}\binom{n}{k} B'.$$

Since $1 - \frac{1}{n} \geq 1/2$ and $k \leq n+2$ hold by assumption, we have in particular

$$B(\mathrm{Alg}, \mathbf{s}) \geq \frac{k}{4n}\binom{n}{k}\overline{\gamma}^{-1}\left(\frac{S_{(1)} - S_{(|Q|)}}{2}\right) = \frac{1}{4}\binom{n-1}{k-1}\overline{\gamma}^{-1}\left(\frac{S_{(1)} - S_{(|Q|)}}{2}\right) \in \Omega\left(\binom{n-1}{k-1}\right).$$

Part 2: The statement holds for arbitrary Alg**.**
Similarly as for the proof of (i) in Theorem 3.1, the proof follows by means of Yao's minimax principle. □

Remark B.6. *(i) To compare the bounds for* RoundRobin *in Theorem C.1 with the lower bound from Theorem 3.1 (ii) suppose in the following* $\mathbf{S}$ *to be homogeneous with* $S_{(1)} > S_{(2)}$ *and let* $\boldsymbol{\gamma}$ *be homogeneous in the sense that* $\gamma_{i|Q}(t) = \gamma(t)$ *for all* $Q \in \mathcal{Q}_{=k}, i \in Q, t \in \mathbb{N}$ *for some* $\gamma : \mathbb{N} \to [0, \infty)$*. Moreover, let* $\mathbf{s}$ *be the instance from the proof of Theorem 3.1 (ii), and denote by* $\mathbf{S}$ *the family of limits* $S_{i|Q} = \lim_{t\to\infty} s_{i|Q}(t)$*,* $Q \in \mathcal{Q}_{\leq k}, i \in Q$*. Let us write* $S^{\mathcal{B}}_{(1)}, \ldots, S^{\mathcal{B}}_{(n)}$ *for the order statistics of* $\{S^{\mathcal{B}}_i\}_{i\in[n]}$*, i.e.,* $S^{\mathcal{B}}_{(1)} \geq \cdots \geq S^{\mathcal{B}}_{(n)}$*. Then,* RoundRobin *returns a GBW of* $\mathbf{s} \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$ *if it is executed with a budget* B *at least*

$$z_{\mathrm{RR}} = \binom{n}{k} B_1 \quad \textit{with} \quad B_1 \coloneqq \overline{\gamma}^{-1}\left(\frac{S^{\mathcal{B}}_{(1)} - S^{\mathcal{B}}_{(2)}}{2}\right).$$

[4] That is, for constructing $\mathbf{s}'$, we proceed for $Q \in \mathcal{W}$ as follows: If $\{1, j\} \subseteq Q$, we exchange $S_{1|Q}$ with $S_{j|Q}$. If $1 \in Q \not\ni j$, we exchange $S_{1|Q}$ with $S_{(Q)|Q}$. And if $j \in Q \not\ni 1$, we exchange $S_{j|Q}$ with $S_{(1)|Q}$.

In comparison to this, the lower bound just shown reveals that any (possibly deterministic) solution to $\mathcal{P}_{\mathrm{GBW}}(\mathbf{S},\boldsymbol{\gamma})$ fulfills

$$\mathbb{E}[B(\mathrm{Alg},\mathbf{s})] \geq \left(1-\frac{1}{n}\right)\frac{k}{k+n-2}\binom{n}{k}B_2 \quad \textit{with} \quad B_2 \coloneqq \overline{\gamma}^{-1}\left(\frac{S_{(1)}-S_{(|Q|)}}{2}\right).$$

Consequently, the optimality-gap between the upper and lower bound is of the order

$$B_1^{-1}B_2\left(1-\frac{1}{n}\right)\frac{k}{k+n-2}.$$

(ii) In the proof of Theorem 3.1 (ii), where we showed that $|\mathcal{W}'| \geq \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ leads to a contradiction, we have constructed an instance $\mathbf{s}' \in \mathfrak{S}(\mathbf{S},\boldsymbol{\gamma})$ with $\mathrm{GBW}(\mathbf{s}) = 1 \notin \mathrm{GBW}(\mathbf{s}')$ such that Alg *observes on $\mathbf{s}$ the same feedback as on $\mathbf{s}'$. To finish the proof, we have only used that* Alg *is correct for $\mathbf{s}$ and for $\mathbf{s}'$, but we did not require correctness of* Alg *on any instance $\mathbf{s}'' \in \mathfrak{S}(\mathbf{S},\boldsymbol{\gamma}) \setminus \{\mathbf{s},\mathbf{s}'\}$. The construction of $\mathbf{s}'$ therein dependeds on the behaviour of* Alg *only by means of the choices of $\mathcal{W}$ and j in the proof, i.e., we have the dependence $\mathbf{s}' = \mathbf{s}'(\mathcal{W},j)$. Recall that for constructing $\mathbf{s}'$ we used that $|\mathcal{W}| = |\mathcal{Q}_{=k}| - |\mathcal{W}'| \geq \binom{n}{k} - \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$, so that for $j \in [n]\setminus\{1\}$, the set*

$$\left\{\mathbf{s}'(\mathcal{W},j)\,\middle|\,\mathcal{W} \subseteq \mathcal{Q}_{=k} \textit{ with } |\mathcal{W}| \geq \binom{n}{k} - \frac{(1-1/n)k}{k+n-2}\binom{n}{k} \textit{ and } j \in [n]\setminus\{1\}\right\}$$

of possible choices for $\mathbf{s}'$ has at most

$$N \coloneqq (n-1)\sum_{l=\left\lceil\binom{n}{k}-\frac{(1-1/n)k}{k+n-2}\binom{n}{k}\right\rceil}^{\binom{n}{k}}\binom{\binom{n}{k}}{l}$$

elements, say $\mathbf{s}'_1,\ldots,\mathbf{s}'_N$. Thus, the formulation of the theorem may be strengthened in the following way:
If $\mathbf{S}$ is homogeneous and $\boldsymbol{\gamma}$ fixed, then there exist $N+1$ instances $\mathbf{s},\mathbf{s}'_1,\ldots,\mathbf{s}'_N$ with the following property: Whenever a (possibly probabilistic) sequential testing algorithm Alg *correctly identifies the GBW for any of these $N+1$ instances, then*

$$\mathbb{E}\left[B(\mathcal{A},\mathbf{s})\right] \geq \left(1-\frac{1}{n}\right)\frac{k}{k+n-2}\binom{n}{k}\overline{\gamma}^{-1}\left(\frac{S_{(1)}-S_{(|Q|)}}{2}\right).$$

B.3 Proof of Theorem 3.1 (iii): Lower Bound for GCopeW Identification

Recall that $\mathrm{GCopeW}(\mathbf{s})$ is the set of elements $i \in [n]$, for which the limits $S_{i|Q} = \lim_{t\to\infty} s_{i|Q}(t)$ have the highest Copeland score

$$S_i^{\mathcal{C}} = \frac{\sum_{Q\in\mathcal{Q}_{=k}(i)}\mathbf{1}\{S_{i|Q}=S_{(1)|Q}\}}{|\mathcal{Q}_{=k}(i)|} = \frac{\sum_{Q\in\mathcal{Q}_{=k}(i)}\mathbf{1}\{S_{i|Q}=S_{(1)|Q}\}}{\binom{n-1}{k-1}}.$$

Proof of Theorem 3.1.(iii). Similarly as in the proofs (i) and (ii) Theorem 3.1, we proceed in two steps.

Part 1: The statement holds in case Alg **is deterministic.**
Abbreviate $B' \coloneqq \min_{Q\in\mathcal{Q}_{\leq k}}\min_{i\in Q}\ \gamma_{i|Q}^{-1}\left(\frac{S_{(1)|Q}-S_{(|Q|)|Q}}{2}\right)$ and fix a family of permutations $(\pi_Q)_{Q\in\mathcal{Q}_{\leq k}}$ with $S_{(1)|Q} = S_{\pi_Q(1)|Q}$ for all $Q \in \mathcal{Q}_{\leq k}(1)$. Exactly as in the proofs of the lower bounds for GCW and GBW identification, we define $\mathbf{s} = (s_{i|Q}(t))_{Q\in\mathcal{Q}_{\leq k},i\in Q,t\in\mathbb{N}}$ via

$$s_{i|Q}(t) \coloneqq \begin{cases}\frac{S_{(1)|Q}+S_{(|Q|)|Q}}{2}, & \text{if } t < B'\\ S_{\pi_Q(i)|Q}, & \text{if } t \geq B'.\end{cases}$$

In the proof of the lower bound of GCW identification we have already verified $\mathbf{s} \in \mathfrak{S}(\mathbf{S},\boldsymbol{\gamma})$. For any $j \in \{2,\ldots,m\}$ and $Q \in \mathcal{Q}_{=k}(1)\cap\mathcal{Q}_{=k}(j)$ we have $S_{1|Q} > S_{j|Q}$, and using that $|\mathcal{Q}_{=k}(i')\setminus\mathcal{Q}_{=k}(j')|$

is the same for every distinct $i', j' \in [n]$ we thus have

$$\begin{aligned}
&\sum_{Q \in \mathcal{Q}_{=k}(1)} \mathbb{1}\{S_{1|Q} = S_{(1)|Q}\} \\
&\quad = \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{1|Q} = S_{(1)|Q}\} + \sum_{Q \in \mathcal{Q}_{=k}(1) \setminus \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{1|Q} = S_{(1)|Q}\} \\
&\quad = \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{1|Q} = S_{(1)|Q}\} + |\mathcal{Q}_{=k}(1) \setminus \mathcal{Q}_{=k}(j)| \\
&\quad > \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{j|Q} = S_{(1)|Q}\} + |\mathcal{Q}_{=k}(j) \setminus \mathcal{Q}_{=k}(1)| \\
&\quad \geq \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{j|Q} = S_{(1)|Q}\} + \sum_{Q \in \mathcal{Q}_{=k}(j) \setminus \mathcal{Q}_{=k}(1)} \mathbb{1}\{S_{j|Q} = S_{(1)|Q}\} \\
&\quad = \sum_{Q \in \mathcal{Q}_{=k}(j)} \mathbb{1}\{S_{j|Q} = S_{(1)|Q}\}.
\end{aligned}$$

As $|\mathcal{Q}_{=k}(1)| = |\mathcal{Q}_{=k}(j)|$, this shows $\mathrm{GCopeW}(\mathbf{s}) = 1$.
Similarly as in the proof of (ii), we will show indirectly that

$$\mathcal{W}' \coloneqq \{Q \in \mathcal{Q}_{=k} \,:\, \text{Alg started on } \mathbf{s} \text{ queries } Q \text{ at least } B' \text{ times}\}$$

contains at least $\frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ elements. For this purpose, let us assume on the contrary $|\mathcal{W}'| < \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ and write $\mathcal{W} \coloneqq \mathcal{Q}_{=k} \setminus \mathcal{W}'$. Lemma B.5 allows us to fix a $j \in [n] \setminus \{1\}$ with $|\mathcal{Q}_{=k}(j) \cap \mathcal{W}| > |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'|$. Now, define $\mathbf{s}' = (s'_{i|Q}(t))_{Q \in \mathcal{Q}_{\leq k}, i \in Q, t \in \mathbb{N}}$ analogously as in the proof of (ii), i.e., via $s'_{\cdot|Q}(\cdot) = s_{\cdot|Q}(\cdot)$ for any $Q \in (\mathcal{Q}_{\leq k} \setminus (\mathcal{Q}_{=k}(1) \cup \mathcal{Q}_{=k}(j))) \cup \mathcal{W}'$ and

$$s'_{i|Q}(t) \coloneqq \begin{cases} s_{i|Q}(t), & \text{if } t < B' \text{ or } \{1, j\} \not\subseteq Q, \\ S_{(1)|Q}, & \text{if } i = j \in Q \text{ and } t \geq B', \\ S_{(|Q|)|Q}, & \text{if } i = 1 \in Q \text{ and } t \geq B', \\ S_{1|Q}, & \text{if } t \geq B', i = \arg\min_{l' \in Q} S_{l'|Q} \text{ and } 1 \in Q \not\ni j, \\ S_{j|Q}, & \text{if } t \geq B', i = \operatorname{argmax}_{l' \in Q} S_{l'|Q} \text{ and } j \in Q \not\ni 1, \\ S_{i|Q}, & \text{otherwise,} \end{cases}$$

for $Q \in (\mathcal{Q}_{=k}(1) \cup \mathcal{Q}_{=k}(j)) \cap \mathcal{W}$. Similarly as for $\mathbf{s}$, we see $\mathbf{s}' \in \mathfrak{S}(\mathbf{S}, \boldsymbol{\gamma})$. The corresponding limit values $S'_{i|Q} = \lim_{t \to \infty} s'_{i|Q}(t)$ fulfill

$$\forall Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W} : S'_{1|Q} = S_{(|Q|)|Q} \quad \text{and} \quad \forall Q \in \mathcal{Q}_{=k}(j) \cap \mathcal{W} : S'_{j|Q} = S_{(1)|Q},$$

and trivially also $S_{(|Q|)|Q} \leq S'_{i|Q} \leq S_{(1)|Q}$ for any $Q \in \mathcal{Q}_{=k}, i \in Q$. Therefore, by choice of j, the corresponding Copeland scores $(S')_i^{\mathcal{C}}$ for $\mathbf{s}'$ fulfill

$$\begin{aligned}
\binom{n-1}{k-1}(S')_1^{\mathcal{C}} &= \sum\nolimits_{Q \in \mathcal{Q}_{=k}(1)} \mathbf{1}\{S'_{1|Q} = S'_{(1)|Q}\} \\
&= \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W}'} \mathbf{1}\{S'_{1|Q} = S'_{(1)|Q}\} + \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W}} \mathbf{1}\{S'_{1|Q} = S'_{(1)|Q}\} \\
&= \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W}'} \mathbf{1}\{S_{1|Q} = S_{(1)|Q}\} + \sum_{Q \in \mathcal{Q}_{=k}(1) \cap \mathcal{W}} \mathbf{1}\{S_{(|Q|)|Q} = S_{(1)|Q}\} \\
&= |\mathcal{Q}_{=k}(1) \cap \mathcal{W}'| \\
&< |\mathcal{Q}_{=k}(j) \cap \mathcal{W}| \\
&= \sum_{\mathcal{Q}_{=k}(j) \cap \mathcal{W}} \mathbf{1}\{S_{(1)|Q} = S_{(1)|Q}\} \\
&= \sum_{\mathcal{Q}_{=k}(j) \cap \mathcal{W}} \mathbf{1}\{S'_{j|Q} = S'_{(1)|Q}\} \\
&\leq \sum_{\mathcal{Q}_{=k}(j) \cap \mathcal{W}} \mathbf{1}\{S'_{j|Q} = S'_{(1)|Q}\} + \sum_{\mathcal{Q}_{=k}(j) \cap \mathcal{W}'} \mathbf{1}\{S'_{j|Q} = S'_{(1)|Q}\} \\
&= \sum_{Q \in \mathcal{Q}_{=k}(j)} \mathbf{1}\{S'_{j|Q} = S'_{(1)|Q}\} \\
&= \binom{n-1}{k-1}(S')_j^{\mathcal{C}},
\end{aligned}$$

where we used that $S_{(1)|Q} = S'_{(1)|Q}$. This shows $1 \notin \mathrm{GCopeW}(\mathbf{s}')$. But since $s_{\cdot|\cdot}(\cdot) = s'_{\cdot|\cdot}(\cdot)$ holds on $\{t < B'\}$ as well as on $\mathcal{W}'$, Alg observes for $\mathbf{s}$ until termination exactly the same feedback as for $\mathbf{s}'$. Consequently, it outputs for both instances the same decision. Since $\mathrm{GCopeW}(\mathbf{s}) = 1 \notin \mathrm{GCopeW}(\mathbf{s}')$, it makes on at least one of the instances a mistake, which contradicts the correctness of Alg.
Thus, $|\mathcal{W}'| \geq \frac{(1-1/n)k}{k+n-2}\binom{n}{k}$ has to hold and we conclude

$$B(\mathrm{Alg}, \mathbf{s}) \geq \sum\nolimits_{Q \in \mathcal{W}'} B_Q(\mathrm{Alg}, \mathbf{s}) \geq |\mathcal{W}'| \cdot B' \geq \left(1 - \frac{1}{n}\right) \frac{k}{k+n-2} \binom{n}{k} B'.$$

Since $1 - \frac{1}{n} \geq 1/2$ and $k \leq n+2$ hold by assumption, we have in particular

$$\begin{aligned}
B(\mathrm{Alg}, \mathbf{s}) &\geq \frac{k}{4n}\binom{n}{k} \min_{Q \in \mathcal{Q}_{\leq k}} \min_{i \in Q} \gamma_{i|Q}^{-1}\left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}\right) \\
&= \frac{1}{4}\binom{n-1}{k-1} \min_{Q \in \mathcal{Q}_{\leq k}} \min_{i \in Q} \gamma_{i|Q}^{-1}\left(\frac{S_{(1)|Q} - S_{(|Q|)|Q}}{2}\right) \in \Omega\left(\binom{n-1}{k-1}\right).
\end{aligned}$$

Part 2: The statement holds for arbitrary Alg**.**
Similarly as for the proofs of the lower bound of (i) and (ii) of this theorem, the proof follows by means of Yao's minimax principle. □

C Generalized Borda Winner Identification

Let RoundRobin be the algorithm, which enumerates all possible subsets of the fixed subset size k, chooses each subset in a round-robin fashion and returns the arm with the highest empirical Borda score $s_i^{\mathcal{B}}$ after the available budget is exhausted. It is a straightforward baseline method, which we analyze theoretically in terms of the sufficient and necessary budget to return a generalized Borda winner (GBW) $i^*_{\mathcal{B}}$. For this purpose, let $\hat{\gamma}_i(t) = \frac{1}{|\mathcal{Q}_{=k}(i)|}\sum_{Q \in \mathcal{Q}_{=k}(i)} \gamma_{i|Q}(t)$ and $\hat{\gamma}_{i,j}^{\max}(t) = \max\{\hat{\gamma}_i(t), \hat{\gamma}_j(t)\}$.

Theorem C.1. ROUNDROBIN *returns $i^*_{\mathcal{B}}$ if it is executed with a budget $B \geq z_{\mathrm{RR}}$, where*

$$z_{\mathrm{RR}} \coloneqq \binom{n}{k} \max_{\rho \in \mathcal{A}, \rho \neq i^*_{\mathcal{B}}} \left(\hat{\gamma}^{\max}_{i^*_{\mathcal{B}}, \rho}\right)^{-1} \left(\frac{S^{\mathcal{B}}_{i^*_{\mathcal{B}}} - S^{\mathcal{B}}_{\rho}}{2}\right).$$

The latter bound is tight in a worst-case scenario, as the following result shows (cf. Sec. D.1 for the proofs).

Theorem C.2. *For any asymptotical Borda scores $S^{\mathcal{B}}_1, \ldots, S^{\mathcal{B}}_n$, there exists a corresponding instance* $\mathbf{s}$ *such that if $B < z_{\mathrm{RR}}$ then* ROUNDROBIN *will not return $i^*_{\mathcal{B}}$.*

Thus, ROUNDROBIN is already nearly-optimal (up to a factor $\mathcal{O}(n/k)$) with respect to worst-case scenarios due to Theorem 3.1 (see Rem. B.6 for a more detailed discussion.).

D Proofs of Section 4

In this section we provide the detailed proofs of Section 4. We assume throughout that $\frac{B}{\binom{n}{k}}$ is a natural number, i.e., the budget is a multiple of $\binom{n}{k}$

D.1 Proof of Theorems C.1 and C.2

Proof of Theorem C.1. After relabeling the arms in round r we may assume w.l.o.g. $i^*_{\mathcal{B}} = 1$. We will prove the theorem by contradiction and therefore assume

$$\begin{aligned}
&\rho = \mathrm{argmax}_{i \in \mathcal{A}}\, s^{\mathcal{B}}_i\left(\frac{B}{\binom{n}{k}}\right) \neq 1 \\
\Rightarrow\ & s^{\mathcal{B}}_1\left(\frac{B}{\binom{n}{k}}\right) < \max_{j=2,\ldots n} s^{\mathcal{B}}_j\left(\frac{B}{\binom{n}{k}}\right) = s^{\mathcal{B}}_{\rho}\left(\frac{B}{\binom{n}{k}}\right) \\
\Rightarrow\ & S^{\mathcal{B}}_1 - S^{\mathcal{B}}_{\rho} < s^{B}_{\rho}\left(\frac{B}{\binom{n}{k}}\right) - S^{\mathcal{B}}_{\rho} + S^{\mathcal{B}}_1 - s^{\mathcal{B}}_1\left(\frac{B}{\binom{n}{k}}\right) \\
&= \frac{1}{|\mathcal{Q}_{=k}(\rho)|} \sum_{Q \in \mathcal{Q}_{=k}(\rho)} \left(s_{\rho|Q}\left(\frac{B}{\binom{n}{k}}\right) - S_{\rho|Q}\right) + \frac{1}{|\mathcal{Q}_{=k}(1)|} \sum_{Q \in \mathcal{Q}_{=k}(1)} \left(S_{1|Q} - s_{1|Q}\left(\frac{B}{\binom{n}{k}}\right)\right) \\
\Rightarrow\ & S^{\mathcal{B}}_1 - S^{\mathcal{B}}_{\rho} < \hat{\gamma}_{\rho}\left(\frac{B}{\binom{n}{k}}\right) + \hat{\gamma}_1\left(\frac{B}{\binom{n}{k}}\right) \\
\Rightarrow\ & S^{\mathcal{B}}_1 - S^{\mathcal{B}}_{\rho} < 2 \cdot \hat{\gamma}^{\max}_{1,\rho}\left(\frac{B}{\binom{n}{k}}\right),
\end{aligned}$$

where $\hat{\gamma}_i(t) = \frac{1}{|\mathcal{Q}_{=k}(i)|} \sum_{Q \in \mathcal{Q}_{=k}(i)} \gamma_{i|Q}(t)$ and $\hat{\gamma}^{\max}_{i,j}(t) = \max\{\hat{\gamma}_i(t), \hat{\gamma}_j(t)\}$. With this, however, we can derive

$$\Rightarrow\ z_{RR} = \left(\hat{\gamma}^{\max}_{1,\rho}\right)^{-1} \left(\frac{S^{\mathcal{B}}_1 - S^{\mathcal{B}}_{\rho}}{2}\right) \binom{n}{k} \geq B,$$

which contradicts the assumption we make on the budget B. Thus, it holds that the returned arm is $\rho = 1$. □

Proof of Theorem C.2. Let $\beta(t)$ be an arbitrary, monotonically decreasing function of t with $\lim_{t\to\infty} \beta(t) = 0$. We define for all $j \in \mathcal{A}$ with $j \neq i^*_{\mathcal{B}}$ the empirical Borda scores to be $s^{\mathcal{B}}_j(t) = S^{\mathcal{B}}_j + \beta(t)$ and $s^{\mathcal{B}}_{i^*_{\mathcal{B}}}(t) = S^{\mathcal{B}}_{i^*_{\mathcal{B}}} - \beta(t)$, where $(S^{\mathcal{B}}_i)_{i \in [n]}$ are arbitrary real values such that $S^{\mathcal{B}}_{i^*_{\mathcal{B}}}$ is the unique maximum for some $i^*_{\mathcal{B}} \in [n]$. We can again assume after relabeling all arms

that w.l.o.g. that $i^*_{\mathcal{B}} = 1$ and $\operatorname{argmax}_{j=2,\dots,n} S^{\mathcal{B}}_j = 2$. Note that $\hat{\gamma}_i(t) = \beta(t)$ for all $i \in \mathcal{A}$. In light of these considerations, ROUNDROBIN returns 1 as the best arm if and only if

$$\begin{aligned}
s^{\mathcal{B}}_1\left(\frac{B}{\binom{n}{k}}\right) > \max_{j=2,\dots,n} s^{\mathcal{B}}_j\left(\frac{B}{\binom{n}{k}}\right) \quad &\Leftrightarrow S^{\mathcal{B}}_1 - \hat{\gamma}_1\left(\frac{B}{\binom{n}{k}}\right) > \max_{j=2,\dots,n} S^{\mathcal{B}}_j + \hat{\gamma}_j\left(\frac{B}{\binom{n}{k}}\right) \\
&\Leftrightarrow S^{\mathcal{B}}_1 - \hat{\gamma}_1\left(\frac{B}{\binom{n}{k}}\right) > S^{\mathcal{B}}_2 + \hat{\gamma}_2\left(\frac{B}{\binom{n}{k}}\right) \\
&\Leftrightarrow \hat{\gamma}_1\left(\frac{B}{\binom{n}{k}}\right) + \hat{\gamma}_2\left(\frac{B}{\binom{n}{k}}\right) < S^{\mathcal{B}}_1 - S^{\mathcal{B}}_2 \\
&\Leftrightarrow 2 \cdot \hat{\gamma}^{\max}_{1,2}\left(\frac{B}{\binom{n}{k}}\right) < S^{\mathcal{B}}_1 - S^{\mathcal{B}}_2 \\
&\Leftrightarrow B \geq \binom{n}{k}\left(\hat{\gamma}^{\max}_{1,2}\right)^{-1}\left(\frac{S^{\mathcal{B}}_1 - S^{\mathcal{B}}_2}{2}\right).
\end{aligned}$$

Thus, the necessary budget is z_{RR} in this case concluding the claim. □

D.2 Proofs of Theorem 4.1 and 4.2

Proof of Theorem 4.1. For the sake of convenience, let us abbreviate $[R] \coloneqq \{1, \dots, R\}$ and $\mathbb{A}_{rj} \coloneqq \mathbb{A}_{r,j}$ in the following. By possibly relabeling the arms and query sets queried by the algorithm, we can assume w.l.o.g. $i^* = 1$ and $\mathbb{A}_r(1) = \mathbb{A}_{r1}$ for all $r \in [R]$ in the following. In particular, we have $S_{1|\mathbb{A}_{r1}} = S_{(1)|\mathbb{A}_{r1}}$ for all $r \in [R]$. We prove the correctness of the algorithm indirectly. Thus, we start by assuming that the best arm is not contained in the last partition (i.e., the remaining active arm):

$$\begin{aligned}
&\mathbb{A}_{R+1} \neq \{1\} \\
\Leftrightarrow &\exists r \in [R] : 1 \notin \mathbb{A}_{r+1} \wedge 1 \in \mathbb{A}_r \\
\Rightarrow &\exists r \in [R] : \sum_{i \in \mathbb{A}_{r1}} \mathbf{1}\{s_{i|\mathbb{A}_{r1}}(b_r) \geq s_{1|\mathbb{A}_{r1}}(b_r)\} > f(|\mathbb{A}_{r1}|) \\
\Rightarrow &\exists r \in [R] : \sum_{i \in \mathbb{A}_{r1}} \mathbf{1}\{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} \leq S_{1|\mathbb{A}_{r1}} - s_{1|\mathbb{A}_{r1}}(b_r) - S_{i|\mathbb{A}_{r1}} + s_{i|\mathbb{A}_{r1}}(b_r)\} > f(|\mathbb{A}_{r1}|) \\
\Rightarrow &\exists r \in [R] : \sum_{i \in \mathbb{A}_{r1}} \mathbf{1}\{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} \leq |S_{1|\mathbb{A}_{r1}} - s_{1|\mathbb{A}_{r1}}(b_r)| + |S_{i|\mathbb{A}_{r1}} - s_{i|\mathbb{A}_{r1}}(b_r)|\} > f(|\mathbb{A}_{r1}|) \\
\Rightarrow &\exists r \in [R] : \sum_{i \in \mathbb{A}_{r1}} \mathbf{1}\{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} \leq 2\bar{\gamma}_{\mathbb{A}_{r1}}(b_r)\} > f(|\mathbb{A}_{r1}|) \\
\Rightarrow &\exists r \in [R] : S_{1|\mathbb{A}_{r1}} - S_{(f(\mathbb{A}_{r1})+1)|\mathbb{A}_{r1}} \leq 2\bar{\gamma}_{\mathbb{A}_{r1}}(b_r) \\
\Rightarrow &\exists r \in [R] : \left\lfloor \frac{B}{P_r R} \right\rfloor = b_r \leq \bar{\gamma}^{-1}_{\mathbb{A}_{r1}}\left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(\mathbb{A}_{r1})+1)|\mathbb{A}_{r1}}}{2}\right) \\
\Rightarrow &\exists r \in [R] : B \leq P_r R \left\lceil \bar{\gamma}^{-1}_{\mathbb{A}_{r1}}\left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2}\right)\right\rceil \\
\Rightarrow &B \leq R \max_{r \in [R]} P_r \left\lceil \bar{\gamma}^{-1}_{\mathbb{A}_{r1}}\left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2}\right)\right\rceil = z\left(f, R, \{P_r\}_{1 \leq r \leq R}\right),
\end{aligned}$$

which contradicts the assumption we make on the budget B. Thus, it holds that the remaining active arm in round $R+1$ is $i^* = 1$. □

Remark D.1. *Using the definition of $\bar{\gamma}_Q(t)$ and $\bar{\gamma}(t)$ we can derive the following more coarser bounds on the sufficient budget:*

$$\begin{aligned}
z_1\left(f, R, \{P_r\}_{1 \leq r \leq R}\right) &= R \max_{r \in [R]} P_r \left\lceil \bar{\gamma}^{-1}\left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2}\right)\right\rceil, \\
z_2\left(f, R, \{P_r\}_{1 \leq r \leq R}\right) &= R \left(\max_{r \in [R]} P_r\right) \max_{Q \in \mathcal{Q}_{\leq k}} \left\lceil \bar{\gamma}^{-1}\left(\frac{S_{1|Q} - S_{(f(|Q|)+1)|Q}}{2}\right)\right\rceil.
\end{aligned}$$

Proof of Theorem 4.2. After relabeling, we may suppose w.l.o.g. $i^* = 1$. Let $\beta : \mathbb{N} \to (0, \infty)$ be an arbitrary strictly decreasing function with $\beta(t) \to 0$ as $t \to \infty$ and

$$\left\{ \frac{S_{1|Q} - S_{j|Q}}{2} : Q \in \mathcal{Q}_{\leq k}, j \in Q \right\} \subseteq \beta(\mathbb{N}).$$

Then, β is invertible on $\beta(\mathbb{N})$ and its inverse function $\beta^{-1} : \beta(\mathbb{N}) \to \mathbb{N}$ trivially fulfills $\beta^{-1}(\alpha) = \min\{t \in \mathbb{N} : \beta(t) \leq \alpha\}$ for all $\alpha \in \beta(\mathbb{N})$. Define for any $Q \in \mathcal{Q}_{\leq k}$ and $i \in Q$ the family of statistics by means of

$$s_{i|Q}(t) \coloneqq \begin{cases} S_{i|Q} - \beta(t), & \text{if } i = \operatorname{argmax}_{j \in Q} S_{j|Q}, \\ S_{i|Q} + \beta(t), & \text{otherwise,} \end{cases}$$

and note that $\overline{\gamma}_Q(t) = \beta(t)$ for all $Q \in \mathcal{Q}_{\leq k}$ and $t \in \mathbb{N}$. Writing $b_r = \left\lfloor \frac{B}{RP_r} \right\rfloor$ we obtain due to the choice of β that

$$\begin{aligned}
&B < R \max_{r \in [R]} P_r \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2} \right) \\
\Rightarrow &\exists r \in [R] : B < RP_r \min \left\{ t \in \mathbb{N} : \bar{\gamma}_{\mathbb{A}_{r1}}(t) \leq \frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2} \right\} \\
\Rightarrow &\exists r \in [R] : b_r < \min \left\{ t \in \mathbb{N} : \beta(t) \leq \frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2} \right\} = \beta^{-1} \left(\frac{S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}}{2} \right) \\
\Rightarrow &\exists r \in [R] : 2\beta(b_r) > S_{1|\mathbb{A}_{r1}} - S_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}} = s_{1|\mathbb{A}_{r1}}(b_r) + \beta(b_r) - \left(s_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}(b_r) - \beta(b_r) \right) \\
\Rightarrow &\exists r \in [R] : s_{1|\mathbb{A}_{r1}}(b_r) < s_{(f(|\mathbb{A}_{r1}|)+1)|\mathbb{A}_{r1}}(b_r) \\
\Rightarrow &\exists r \in [R] : 1 \notin \mathbb{A}_{r+1} \\
\Rightarrow &1 \notin \mathbb{A}_{R+1}.
\end{aligned}$$

This shows that $z\left(f, R, \{P_r\}_{1 \leq r \leq R}\right)$ is the necessary budget for returning the best arm i^* in this scenario. □

D.3 Proof of Corollary 4.3

For sake of convenience, we provide the entire pseudo-code of CSWS in Algorithm 3, which results by using $f(x) = x - 1$ as well as P_r^{CSWS} and R^{CSWS} as defined in Section 4.1 in Algorithm 1.

Proof of Corollary 4.3 (CSWS case). Suppose $B > 0$ to be arbitrary but fixed. First, note that there are at most $\lceil \log_k(n) \rceil$ rounds within the first while-loop and at most 1 in the second, so that we have at most $\lceil \log_k(n) \rceil + 1$ many rounds in total. The total number of partitions in round $r \in \{1, \ldots, \lceil \log_k(n) \rceil + 1\}$ is at most $\left\lceil \frac{n}{k^r} \right\rceil$. Abbreviating $R \coloneqq R^{\mathrm{CSWS}}$ and $P_r \coloneqq P_r^{\mathrm{CSWS}}$ for the moment, the budget allocated to a partition in round r is by definition $b_r = \lfloor \frac{B}{RP_r} \rfloor = \left\lfloor \frac{B}{\left\lceil \frac{n}{k^r} \right\rceil \cdot (\lceil \log_k(n) \rceil + 1)} \right\rfloor$. Hence, the total budget used by CSWS is

$$\sum_{r=1}^{\lceil \log_k(n) \rceil + 1} \#\{\text{partitions in round r}\} \cdot b_r = \sum_{r=1}^{\lceil \log_k(n) \rceil + 1} \left\lceil \frac{n}{k^r} \right\rceil \left\lfloor \frac{B}{\left\lceil \frac{n}{k^r} \right\rceil \cdot (\lceil \log_k(n) \rceil + 1)} \right\rfloor \leq B.$$

Thus, the stated correctness of CSWS follows directly from Theorem 4.1.

□

For sake of convenience, we provide the entire pseudo-code of CSR in Algorithm 4, which results by using $f(x) = 1$ as well as P_r^{CSR} and R^{CSR} as defined in Section 4.1 in Algorithm 1.

Proof of Corollary 4.3 (CSR case). Suppose $B > 0$ to be arbitrary but fixed. First, note that there are at most $\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil$ rounds within the first while-loop and at most $k - 1$ in the second, so that we have at most $\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1$ many rounds in total. The total number of partitions in round

Algorithm 3 Combinatorial Successive Winner Stays (CSWS)

Input: set of arms $[n]$, subset size $k \leq n$, sampling budget B

Initialization: For each $r \in \{1, \dots, \lceil \log_k(n) \rceil + 1\}$ let $b_r := \left\lfloor \frac{B}{\lceil \frac{n}{k^r} \rceil \cdot (\lceil \log_k(n) \rceil + 1)} \right\rfloor$, $\mathbb{A} \leftarrow [n]$, $r \leftarrow 1$

while $|\mathbb{A}_r| \geq k$ **do**
  $J = \lceil \frac{n}{k^r} \rceil$
  $\mathbb{A}_{r1}, \mathbb{A}_{r2}, \dots, \mathbb{A}_{r,J} \leftarrow Partition(\mathbb{A}_r, k)$
  if $|\mathbb{A}_{r,J}| < k$ **then**
    $\mathcal{R} \leftarrow \mathbb{A}_{r,J}$, $J \leftarrow J - 1$
  else
    $\mathcal{R} \leftarrow \emptyset$
  end if
  $\mathbb{A}_{r+1} \leftarrow \emptyset$
  for $j \in [J]$ **do**
    Play the set $\mathbb{A}_{r,j}$ for b_r times
    For all $i \in \mathbb{A}_{r,j}$, update $s_{i|\mathbb{A}_{r,j}}(b_r)$
    Let $w \in \operatorname{argmax}_i s_{i|\mathbb{A}_{r,j}}(b_r)$
    $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \{w\}$
  end for
  $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \mathcal{R}$
  $r \leftarrow r + 1$
end while
$\mathbb{A}_{r+1} \leftarrow \emptyset$
while $|\mathbb{A}_r| > 1$ **do**
  Play the set $\mathbb{A}_r$ for b_r times
  For all $i \in \mathbb{A}_r$, update $s_{i|\mathbb{A}_r}(b_r)$
  Let $w \in \operatorname{argmax}_i s_{i|\mathbb{A}_r}(b_r)$
  $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \{w\}$
  $r \leftarrow r + 1$
end while

Output: The remaining item in $\mathbb{A}_r$

$r \in \{1, \dots, \left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1\}$ is at most $\left\lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \right\rceil$. The budget allocated to a partition in round r (i.e., b_r) is by definition given by

$$b_r = \lfloor B/(R^{\mathrm{CSR}} P_r^{\mathrm{CSR}}) \rfloor = \left\lfloor \frac{B}{\left\lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \right\rceil \left(\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1 \right)} \right\rfloor.$$

Consequently, the total budget used by CSR is

$$\begin{aligned}
&\sum_{r=1}^{\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1} \#\{\text{partitions in round r}\} \cdot b_r \\
&= \sum_{r=1}^{\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1} \left\lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \right\rceil \left\lfloor \frac{B}{\left\lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \right\rceil \left(\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \right\rceil + k - 1 \right)} \right\rfloor \\
&\leq B.
\end{aligned}$$

Therefore, the statement follows from Theorem 4.1.

□

For sake of convenience, we provide the entire pseudo-code of CSH in Algorithm 5, which results by using $f(x) = \lceil x/2 \rceil$ as well as P_r^{CSH} and R^{CSH} as defined in Section 4.1 in Algorithm 1.

Proof of Corollary 4.3 (CSH case). Suppose $B > 0$ to be arbitrary but fixed. First, note that there are at most $\lceil \log_2(n) \rceil$ rounds within the first while-loop and at most $\lceil \log_2(k) \rceil$ in the second, so that

Algorithm 4 Combinatorial Successive Reject (CSR)

Input: set of arms $[n]$, subset size $k \leq n$, sampling budget B

Initialization: For each $r \in \{0, \dots, \lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\rceil\}$ let $b_r := \left\lfloor \frac{B}{\left\lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \right\rceil \left(\left\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil + k - 1\right)} \right\rfloor$,

$\mathbb{A} \leftarrow [n], r \leftarrow 1$

while $|\mathbb{A}_r| \geq k$ **do**
  $J = \lceil \frac{n(1-\frac{1}{k})^{r-1}}{k} \rceil$
  $\mathbb{A}_{r1}, \mathbb{A}_{r2}, \dots, \mathbb{A}_{r,J} \leftarrow Partition(\mathbb{A}_r, k)$
  if $|\mathbb{A}_{r,J}| < k$ **then**
    $\mathcal{R} \leftarrow \mathbb{A}_{r,J}$, $J \leftarrow J - 1$
  else
    $\mathcal{R} \leftarrow \emptyset$
  end if
  $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_r$
  for $j \in [J]$ **do**
    Play the set $\mathbb{A}_{r,j}$ for b_r times
    For all $i \in \mathbb{A}_{r,j}$, update $s_{i|\mathbb{A}_{r,j}}(b_r)$
    Let $w \in \arg\min_i \ s_{i|\mathbb{A}_{r,j}}(b_r)$
    $\mathbb{A}_{r+1} = \mathbb{A}_{r+1} \backslash \{w\}$
  end for
  $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \mathcal{R}$
  $r \leftarrow r + 1$
end while
$\mathbb{A}_{r+1} \leftarrow \mathbb{A}_r$
while $|\mathbb{A}_r| > 1$ **do**
  Play the set $\mathbb{A}_r$ for b_r times
  For all $i \in \mathbb{A}_r$, update $s_{i|\mathbb{A}_r}(b_r)$
  Let $w \in \arg\min_i \ s_{i|\mathbb{A}_r}(b_r)$
  $\mathbb{A}_{r+1} = \mathbb{A}_{r+1} \backslash \{w\}$
  $r \leftarrow r + 1$
end while

Output: The remaining item in $\mathbb{A}_r$

we have at most $\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil$ many rounds in total. The total number of partitions in round $r = 1, \dots, \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil$ is at most $\left\lceil \frac{n}{2^{r-1}k} \right\rceil$. The budget allocated to a partition in round r is

$$b_r = \lfloor B/(R^{\mathrm{CSH}} P_r^{\mathrm{CSH}}) \rfloor = \left\lfloor \frac{B}{\left\lceil \frac{n}{2^{r-1}k} \right\rceil \cdot (\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil)} \right\rfloor .$$

In particular, the total budget used by CSH is

$$\begin{aligned} &\sum_{r=1}^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil} \#\{\text{partitions in round r}\} \cdot b_r \\ &= \sum_{r=1}^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil} \left\lceil \frac{n}{2^{r-1}k} \right\rceil \cdot \left\lfloor \frac{Bk}{\lceil \frac{n}{2^{r-1}} \rceil (\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil)} \right\rfloor \\ &\leq B. \end{aligned}$$

Once again, Theorem 4.1 allows us to conclude the proof. □

E Proofs of Section 5

E.1 Stochastic Numerical Feedback: Proof of Corollary 5.1

A rich class of statistics can be obtained by applying a linear functional $U(F) = \int r(x) \mathrm{d}F(x)$, where F is a cumulative distribution function and $r : \mathbb{R} \to \mathbb{R}$ some measurable function, on the empirical distribution function [49], i.e., for any $x \in \mathbb{R}$ and any multiset of (reward) observations O

Algorithm 5 Combinatorial Successive Halving (CSH)

Input: set of arms $[n]$, subset size $k \leq n$, sampling budget B

Initialization: For each $r \in \{0, \dots, \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil\}$ let $b_r := \left\lfloor \frac{Bk}{\lceil \frac{n}{2^{r-1}} \rceil (\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil)} \right\rfloor$,

$\mathbb{A} \leftarrow [n]$, $r \leftarrow 1$

while $|\mathbb{A}_r| \geq k$ **do**
  $J = \lceil \frac{n}{2^{r-1}k} \rceil$
  $\mathbb{A}_{r1}, \mathbb{A}_{r2}, \dots, \mathbb{A}_{r,J} \leftarrow Partition(\mathbb{A}_r, k)$
  if $|\mathbb{A}_{r,j}| < k$ **then**
    $\mathcal{R} \leftarrow \mathbb{A}_{r,j}$, $J \leftarrow J - 1$
  else
    $\mathcal{R} \leftarrow \emptyset$
  end if
  for $j \in [J]$ **do**
    Play the set $\mathbb{A}_{r,j}$ for b_r times
    For all $i \in \mathbb{A}_{r,j}$, update $s_{i|\mathbb{A}_{r,j}}(b_r)$
    Define $\bar{s} \leftarrow$ Median$(\{s_{i|\mathbb{A}_{r,j}}(b_r)\}_{i \in \mathbb{A}_{r,j}})$
    $\mathbb{A}_{r+1} \leftarrow \{i \in \mathbb{A}_{r,j} | s_{i|\mathbb{A}_{r,j}}(b_r) \leq \bar{s}\}$
  end for
  $\mathbb{A}_{r+1} \leftarrow \mathbb{A}_{r+1} \cup \mathcal{R}$
  $r \leftarrow r + 1$
end while
$\mathbb{A}_r \leftarrow \mathbb{A}_r \cup \{k - |\mathbb{A}_r|$ random elements from $[n] \backslash \mathbb{A}_r\}$
while $|\mathbb{A}_r| > 1$ **do**
  Play the set $\mathbb{A}_r$ for b_r times
  For all $i \in \mathbb{A}_r$, update $s_{i|\mathbb{A}_r}(b_r)$
  Define $\bar{s} \leftarrow$ Median$(\{s_{i|\mathbb{A}_r}(b_r)\}_{i \in \mathbb{A}_r})$
  $\mathbb{A}_{r+1} \leftarrow \{i \in \mathbb{A}_r | s_{i|\mathbb{A}_r}(b_r) \leq \bar{s}\}$
  $r \leftarrow r + 1$
end while

Output: The remaining item in $\mathbb{A}_r$

$$\tilde{s}(O, x) = \tfrac{1}{|O|} \textstyle\sum_{o \in O} \mathbb{1}\{x \leq o\}.$$

This leads to the statistics

$$s_{i|Q}(t) = U(\tilde{s}(o_{i|Q}(1), \dots, o_{i|Q}(t), \cdot)) = \sum_{s=1}^{t} \frac{r(o_{i|Q}(s))}{t},$$

which converge to $S_{i|Q} = \mathbb{E}_{X \sim \nu_{i|Q}}[r(X)]$ by the law of large numbers, provided these expected values exist. In this section we show the following result which generalizes Corollary 5.1 for statistics of the above kind.

Corollary E.1. *Let f, R and $\{P_r\}_{r \in [R]}$ be as in Theorem 4.1 and suppose that $r(o_{i|Q}(t))$ are σ-sub-Gaussian and such that their means $S_{i|Q} := \mathbb{E}_{X \sim \nu_{i|Q}}[r(X)]$ satisfy (A2). Then, there is a function*

$$C(\delta, \varepsilon, k, R, \sigma) \in \mathcal{O}\left(\sigma^2 \varepsilon^{-2} \ln\left({}^{kR}/_{\delta} \ln\left({}^{kR\sigma}/_{\varepsilon\delta}\right)\right)\right)$$

with the following property: If i^* is the GCW and $\sup_{Q \in \mathcal{Q}_{\leq k}(i^*)} \Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$, then Algorithm 1 used with a budget B larger than $C(\delta, \varepsilon, k, R, \sigma) \cdot R \max_{r \in [R]} P_r$ returns i^* with probability at least $1 - \delta$.

Note that we immediately obtain the proof for Corollary 5.1 as a special case of Corollary E.1 by using the the identity function $r(x) = x$.

The following two lemmata serve as a preparation for the proof of Corollary E.1. The proof of Lemma E.2 is an adaptation of the proof of Lemma 3 in [24].

Lemma E.2. *Let $X_1, X_2, \dots \sim \mathcal{X}$ be iid real-valued random variables and $r : \mathbb{R} \to \mathbb{R}$ such that $r(\mathcal{X})$ is σ^2-sub-Gaussian. For any $\epsilon \in (0, 1)$ and $\delta \in (0, \log(1 + \epsilon)/e)$ one has with probability at*

least $1 - \frac{(2+\epsilon)}{\epsilon}\left(\frac{\delta}{\log(1+\epsilon)}\right)^{(1+\epsilon)}$ *for any* $t \geq 1$

$$\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)] \leq (1+\sqrt{\epsilon})\sqrt{2\sigma^2(1+\epsilon)t\log\left(\frac{\log((1+\epsilon)t)}{\delta}\right)}.$$

Moreover, the same concentration inequality holds for $-\left(\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]\right)$ *as well.*

Proof. We denote in the following $\psi(x) = \sqrt{2\sigma^2 x\log\left(\frac{\log(x)}{\delta}\right)}$ and $R_t = \sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]$ and define a sequence of integers (u_k) as $u_0 = 1$ and $u_{k+1} = \lceil(1+\epsilon)u_k\rceil$. The maximal *Azuma-Hoeffding Inequality* states that for any martingale difference sequence $S_1, S_2, \ldots$ with each element being σ^2-sub-Gaussian, it holds that for any $\alpha > 0$, $n \geq 1$:

$$\mathbb{P}\left(\max_{i\in[n]} S_i - S_0 \geq \alpha\right) \leq \exp\left(-\frac{\alpha^2}{2\sum_{j=1}^{n}\sigma_j^2}\right).$$

In the following let $\mathcal{F}_0 = \{\emptyset, \Omega\}$ be the trivial σ-algebra and for $k \in \{1, \ldots, n\}$ let $\mathcal{F}_k = \sigma(X_1, \ldots, X_k)$ be the σ-algebra generated by the observations $X_1, \ldots, X_k$. Then

$$\begin{aligned}\mathbb{E}[R_{t+1}|\mathcal{F}_t] &= \mathbb{E}[r(X_{t+1}) - \mathbb{E}_{X\sim\mathcal{X}}[r(X)] + R_t|\mathcal{F}_t] \\ &= \mathbb{E}[r(X_{t+1})|\mathcal{F}_t] - \mathbb{E}_{X\sim\mathcal{X}}[r(X)] + \mathbb{E}[R_t|\mathcal{F}_t] \\ &= R_t\end{aligned}$$

which shows the martingale property of R_t. Note, that $R_0 = 0$ and $R_{t+1} - R_t = r(X_{t+1}) - \mathbb{E}_{X\sim\mathcal{X}}[r(X)]$, which is according to the assumption σ^2-sub-Gaussian and has zero mean, for any $t \in \mathbb{N}$. Thus, we can apply the maximal Azuma-Hoeffding inequality for $R_1, R_2, \ldots, R_t$.

Step 1.
In the first step of the proof we derive a bound for the probability of a lower bound of R_{u_k} for $k \geq 1$. For this we use the union bound, the maximal Azuma-Hoeffding inequality, the fact that $u_k \geq (1+\epsilon)^k$, a sum-integral comparison and some simple transformations and obtain

$$\begin{aligned}&\mathbb{P}\left(\exists k \geq 1 \,:\, R_{u_k} \geq \sqrt{1+\epsilon}\psi(u_k)\right) \\ &\leq \sum_{k=1}^{\infty}\mathbb{P}\left(R_{u_k} \geq \sqrt{1+\epsilon}\psi(u_k)\right) \\ &\leq \sum_{k=1}^{\infty}\exp\left(-\frac{(1+\epsilon)\psi(u_k)^2}{2u_k\sigma^2}\right) \\ &= \sum_{k=1}^{\infty}\exp\left(-(1+\epsilon)\log\left(\frac{\log(u_k)}{\delta}\right)\right) \\ &\leq \sum_{k=1}^{\infty}\exp\left(-(1+\epsilon)\log\left(\frac{\log((1+\epsilon)^k)}{\delta}\right)\right) \\ &= \sum_{k=1}^{\infty}\left(\frac{\delta}{k\log((1+\epsilon))}\right)^{(1+\epsilon)} \\ &= \left(\frac{2\delta}{\log((1+\epsilon))}\right)^{(1+\epsilon)}\sum_{k=1}^{\infty}\left(\frac{1}{k}\right)^{(1+\epsilon)} \\ &= \left(\frac{\delta}{\log((1+\epsilon))}\right)^{(1+\epsilon)}\left(1+\sum_{k=2}^{\infty}\left(\frac{1}{k}\right)^{(1+\epsilon)}\right) \\ &\leq \left(\frac{\delta}{\log((1+\epsilon))}\right)^{(1+\epsilon)}\left(1+\int_{k=1}^{\infty}\left(\frac{1}{k}\right)^{(1+\epsilon)}\right)\end{aligned}$$

$$= \left(\frac{\delta}{\log((1+\epsilon))}\right)^{(1+\epsilon)} \left(1 + \left[-\frac{1}{\epsilon}\left(\frac{1}{k}\right)^{\epsilon}\right]_1^{\infty}\right)$$
$$= \left(\frac{\delta}{\log((1+\epsilon))}\right)^{(1+\epsilon)} \left(1 + \frac{1}{\epsilon}\right).$$

Step 2.
Next, we bound the probability that the difference between some R_s and R_t exceeds a lower bound for some $s = u_k$, $k \in \mathbb{N}$ and $s \leq t \leq u_{k+1}$. Note that $R_t - R_{u_k}$ and R_{t-u_k} have the same distribution, such that we obtain

$$\begin{aligned}
&\mathbb{P}\left(\exists t \in \{u_k+1, \dots, u_{k+1}-1\} \,:\, R_t - R_{u_k} \geq \sqrt{\epsilon}\psi(u_{k+1})\right) \\
&= \mathbb{P}\left(\exists t \in [u_{k+1}-u_k-1] \,:\, R_t \geq \sqrt{\epsilon}\psi(u_{k+1})\right) \\
&\leq \exp\left(-\frac{\epsilon\psi(u_{k+1})^2}{2\sigma^2(u_{k+1}-u_k-1)}\right) \\
&= \exp\left(-\frac{\epsilon u_{k+1}}{u_{k+1}-u_k-1}\log\left(\frac{\log(u_{k+1})}{\delta}\right)\right) \\
&\leq \exp\left(-\frac{\epsilon u_{k+1}}{(1+\epsilon)u_k+1-u_k-1}\log\left(\frac{\log(u_{k+1})}{\delta}\right)\right) \\
&= \exp\left(-\frac{u_{k+1}}{u_k}\log\left(\frac{\log(u_{k+1})}{\delta}\right)\right) \\
&\leq \exp\left(-(1+\epsilon)\log\left(\frac{\log(u_{k+1})}{\delta}\right)\right) \\
&\leq \left(\frac{\delta}{(k+1)\log(1+\epsilon)}\right)^{1+\epsilon},
\end{aligned}$$

where we used once again the maximal Azuma-Hoeffding inequality and that $u_{k+1} \geq (1+\epsilon)u_k$ as well as that $\frac{u_{k+1}}{u_k} \geq 1+\epsilon$. For all possible $k \in \mathbb{N}$ we get with the union bound and a similar sum-integral comparison as above

$$\begin{aligned}
&\mathbb{P}\left(\exists k \in \mathbb{N},\, \exists t \in \{u_k+1, \dots, u_{k+1}-1\} \,:\, R_t - R_{u_k} \geq \sqrt{\epsilon}\psi(u_{k+1})\right) \\
&\leq \sum_{k=1}^{\infty}\left(\frac{\delta}{(k+1)\log(1+\epsilon)}\right)^{1+\epsilon} \\
&= \sum_{k=2}^{\infty}\left(\frac{\delta}{k\log(1+\epsilon)}\right)^{1+\epsilon} \\
&\leq \int_{k=1}^{\infty}\left(\frac{\delta}{k\log(1+\epsilon)}\right)^{1+\epsilon} \\
&= \left(\frac{\delta}{\log(1+\epsilon)}\right)^{1+\epsilon}\frac{1}{\epsilon}
\end{aligned}$$

Step 3.
Finally, by combining Step 1 and 2 we can infer that for any $k \geq 0$ and $t \in \{u_k+1, \dots, u_{k+1}-1\}$ it holds

$$\begin{aligned}
R_t &= R_t - R_{u_k} + R_{u_k} \\
&\leq \sqrt{\epsilon}\psi(u_{k+1}) + \sqrt{1+\epsilon}\psi(u_k) \\
&\leq \sqrt{\epsilon}\psi((1+\epsilon)t) + \sqrt{1+\epsilon}\psi(t) \\
&\leq (1+\sqrt{\epsilon})\psi((1+\epsilon)t),
\end{aligned}$$

with probability at least $1 - \frac{2+\epsilon}{\epsilon}\left(\frac{\delta}{\log(1+\epsilon)}\right)^{1+\epsilon}$ leading to the first claim of the lemma.

Step 4.

Note that $\tilde{R}_t = t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)] - \sum_{i=1}^{t} r(X_i)$ is a martingale difference sequence with $\tilde{R}_{t+1} - \tilde{R}_t = -R_t + R_{t+1} = \mathbb{E}_{X\sim\mathcal{X}}[r(X)] - r(X_{t+1})$, which is according to the assumption σ^2-sub-Gaussian and has zero mean, for any $t \in \mathbb{N}$. Thus, repeating Step 1–3 for $\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_t$ shows the second claim of the lemma. □

Lemma E.3. *Let $X_1, X_2, \ldots \sim \mathcal{X}$ be iid real-valued random variables and $r : \mathbb{R} \to \mathbb{R}$ such that $r(\mathcal{X})$ is σ^2-sub-Gaussian. For any $\gamma \in (0,1)$ we have*

$$\mathbb{P}\left(\exists t \in \mathbb{N} : \left|\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]\right| > (1+\sqrt{1/2})\sqrt{3\sigma^2 t \ln\left(\frac{10^{2/3}\ln(3t/2)}{\gamma^{2/3}\ln(3/2)}\right)}\right) \leq \gamma.$$

Proof. Let $\gamma \in (0,1)$ be fixed and $\varepsilon \coloneqq 1/2$. Then, $\gamma' \coloneqq \left(\frac{\gamma}{10}\right)^{2/3}\ln(3/2)$ fulfills

$$\frac{2+\varepsilon}{\varepsilon}\left(\frac{\gamma'}{\ln(1+\varepsilon)}\right)^{1+\varepsilon} = 5\left((\gamma/10)^{2/3}\right)^{3/2} = \gamma/2$$

and moreover $\gamma' < (1/10)^{2/3}\ln(3/2) < e^{-1}\ln(3/2)$. Consequently, Lemma E.2 yields with

$$\tilde{c}_\gamma(t) \coloneqq (1+\sqrt{\varepsilon})\sqrt{2\sigma^2(1+\varepsilon)t\ln\left(\frac{\ln((1+\varepsilon)t)}{\gamma'}\right)} = (1+\sqrt{1/2})\sqrt{3\sigma^2 t\ln\left(\frac{10^{2/3}\ln(3t/2)}{\gamma^{2/3}\ln(3/2)}\right)}$$

that

$$\mathbb{P}\left(\exists t \in \mathbb{N} : \sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)] > \tilde{c}_\gamma(t)\right) \leq \gamma/2.$$

as well as

$$\mathbb{P}\left(\exists t \in \mathbb{N} : -\left(\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]\right) > \tilde{c}_\gamma(t)\right) \leq \gamma/2.$$

Thus, we obtain

$$\begin{aligned}
&\mathbb{P}\left(\exists t \in \mathbb{N} : \left|\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]\right| > \tilde{c}_\gamma(t)\right) \\
&\quad\leq \mathbb{P}\left(\exists t \in \mathbb{N} : \sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)] > \tilde{c}_\gamma(t)\right) \\
&\quad\quad+ \mathbb{P}\left(\exists t \in \mathbb{N} : -\left(\sum_{i=1}^{t} r(X_i) - t \cdot \mathbb{E}_{X\sim\mathcal{X}}[r(X)]\right) > \tilde{c}_\gamma(t)\right) \\
&\quad\leq \gamma/2 + \gamma/2 = \gamma.
\end{aligned}$$

□

We are now ready to prove Corollary E.1.

Proof of Corollary E.1. Recall the definition of $\tilde{c}_\gamma(t)$ from the proof of Lemma E.3 and let

$$c_\gamma(t) \coloneqq \frac{2}{t}\tilde{c}_\gamma(t) = 2(1+\sqrt{1/2})\sqrt{\frac{3\sigma^2}{t}\ln\left(\frac{10^{2/3}\ln(3t/2)}{\gamma^{2/3}\ln(3/2)}\right)}$$

for any $\gamma \in (0,1)$, $t \in \mathbb{N}$. For any fixed γ, $c_\gamma : \mathbb{N} \to (0,\infty), t \mapsto c_\gamma(t)$ is strictly monotonically decreasing with $\lim_{t\to\infty} c_\gamma(t) = 0$. Contraposition of (1) in [24] states

$$t > \frac{1}{c}\ln\left(\frac{2\ln((1+\varepsilon)/(c\omega))}{\omega}\right) \;\Rightarrow\; c > \frac{1}{t}\ln\left(\frac{\ln((1+\varepsilon)t)}{\omega}\right) \qquad \forall t \geq 1, \varepsilon \in (0,1), c > 0, \omega \leq 1.$$

For any $\alpha > 0$ and $\gamma \in (0,1)$, using this with $\omega = \frac{\gamma^{2/3}\ln(3/2)}{10^{2/3}}, c = \frac{\alpha^2}{12(1+\sqrt{1/2})^2\sigma^2}$ and $\varepsilon = 1/2$ reveals

$$\begin{aligned} c_\gamma^{-1}(\alpha) &= \min\{t \in \mathbb{N} : c_\gamma(t) \leq \alpha\} \\ &= \min\left\{t \in \mathbb{N} : \frac{1}{t}\ln\left(\frac{10^{2/3}\ln(3t/2)}{\gamma^{2/3}\ln(3/2)}\right) \leq \frac{\alpha^2}{12(1+\sqrt{1/2})^2\sigma^2}\right\}. \end{aligned}$$

Thus, we have $c \geq \frac{1}{t}\ln\left(\frac{\ln((1+\varepsilon)t)}{\omega}\right)$ and we know, that this statement is true if $t \geq \frac{1}{c}\ln\left(\frac{2\ln((1+\varepsilon)/(c\omega))}{\omega}\right)$. In particular also for the smallest such t, for which holds $t \leq \left\lceil \frac{1}{c}\ln\left(\frac{2\ln((1+\varepsilon)/(c\omega))}{\omega}\right)\right\rceil$ + 1. It follows

$$c_\gamma^{-1}(\alpha) \leq \left\lceil \frac{12(1+\sqrt{1/2})^2\sigma^2}{\alpha^2}\ln\left(\frac{2\cdot 10^{2/3}}{\gamma^{2/3}\ln(3/2)}\ln\left(\frac{18\cdot 10^{2/3}(1+\sqrt{1/2})^2\sigma^2}{\gamma^{2/3}\ln(3/2)\,\alpha^2}\right)\right)\right\rceil + 1,$$

which is of the order $\mathcal{O}(\sigma^2\alpha^{-2}\ln\ln(\alpha^{-1}\sigma)\ln\gamma^{-1})$.

Now, suppose $\max_{Q\in\mathcal{Q}_{\leq k}(i^*)}\Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$ and that Algorithm 1 is started with a budget B larger than

$$c_{\delta/(kR)}^{-1}(\varepsilon/2)\cdot R\max_{r\in[R]}P_r.$$

Recall that $\gamma_{i|Q}(t) = |s_{i|Q}(t) - S_{i|Q}|$, $s_{i|Q}(t) = \frac{1}{t}\sum_{s=1}^t r(o_{i|Q}(s))$ and $S_{i|Q} = \mathbb{E}_{X\sim\nu_{i|Q}}[r(X)]$. With this, we obtain for any possible sequence of partitions $(E_r)_{r\in[R]} \in (\mathcal{Q}_{\leq k})^R$ with $\mathbb{P}(\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]) > 0$ that

$$\begin{aligned} &\mathbb{P}\left(\exists t\in\mathbb{N}, r\in[R], i\in E_r : \gamma_{i|E_r}(t) \geq c_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &\leq \sum_{r\in[R]}\sum_{i\in E_r}\mathbb{P}\left(\exists t\in\mathbb{N} : \gamma_{i|E_r}(t) \geq c_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &= \sum_{r\in[R]}\sum_{i\in E_r}\mathbb{P}\left(\exists t\in\mathbb{N} : \left|\frac{1}{t}\sum_{t'=1}^t r(o_{i|E_r}(t')) - \mathbb{E}_{X\sim\nu_{i|E_r}}[r(X)]\right| \geq c_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &= \sum_{r\in[R]}\sum_{i\in E_r}\mathbb{P}\left(\exists t\in\mathbb{N} : \frac{1}{t}\left|\sum_{t'=1}^t r(o_{i|E_r}(t')) - t\cdot\mathbb{E}_{X\sim\nu_{i|E_r}}[r(X)]\right| \geq c_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &= \sum_{r\in[R]}\sum_{i\in E_r}\mathbb{P}\left(\exists t\in\mathbb{N} : \left|\sum_{t'=1}^t r(o_{i|E_r}(t')) - t\cdot\mathbb{E}_{X\sim\nu_{i|E_r}}[r(X)]\right| \geq \tilde{c}_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &\leq \sum_{r\in[R]}\sum_{i\in E_r}\frac{\delta}{kR} \leq \delta, \end{aligned}$$

where we used Lemma E.3 for the second last inequality. Using the law of total probability for all possible sequences of partitions $(E_r)_{r\in[R]}$, we see that the event

$$\mathcal{E} := \left\{\exists t\in\mathbb{N}, r\in[R], i\in\mathbb{A}_r(i^*) : \gamma_{i|\mathbb{A}_r(i^*)}(t) \geq c_{\delta/(kR)}(t)\right\}$$

occurs with probability

$$\begin{aligned} &\mathbb{P}(\mathcal{E}) \\ &= \sum_{(E_r)_{r\in[R]}}\mathbb{P}\left(\exists t\in\mathbb{N}, r\in[R], i\in E_r : \gamma_{i|E_r}(t) \geq c_{\delta/(kR)}(t)\,\middle|\,\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]\right) \\ &\quad\times\ \mathbb{P}(\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]) \\ &\leq \delta\sum\nolimits_{(E_r)_{r\in[R]}}\mathbb{P}(\mathbb{A}_r(i^*) = E_r\,\forall r\in[R]) = \delta. \end{aligned}$$

On $\mathcal{E}^c$ we have $\overline{\gamma}_{\mathbb{A}_r(i^*)}(t) < c_{\delta/(kR)}(t)$ for all $t\in\mathbb{N}, r\in[R]$ and thus in particular $\overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}(\alpha) \geq c^{-1}_{\delta/(kR)}(\alpha)$ for any $\alpha\in(0,\infty)$. Since $\max_{Q\in\mathcal{Q}_{\leq k}(i^*)}\Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$, Theorem 4.1 thus lets us

conclude

$$\begin{aligned}\mathbb{P}\left(\text{Alg. 1 returns } i^*\right) &\geq \mathbb{P}\left(B > R\max_{r\in[R]} P_r \overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\frac{\Delta_{(f(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right)\right)\\ &\geq \mathbb{P}\left(\left\{B > R\max_{r\in[R]} P_r \overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\varepsilon/2\right)\right\} \cap \mathcal{E}^c\right)\\ &\geq \mathbb{P}\left(\left\{B > R\max_{r\in[R]} P_r c^{-1}_{\delta/(kR)}\left(\varepsilon/2\right)\right\} \cap \mathcal{E}^c\right)\\ &= \mathbb{P}(\mathcal{E}^c) \geq 1-\delta,\end{aligned}$$

where the equality holds due to the assumption on B. Consequently, we can conclude the proof by defining

$$\begin{aligned}C(\delta,\varepsilon,k,R) &\coloneqq c^{-1}_{\delta/(kR)}(\varepsilon/2)\\ &\leq \left\lceil \frac{48(1+\sqrt{1/2})^2\sigma^2}{\varepsilon^2}\ln\left(\frac{2(10kR)^{2/3}}{\delta^{2/3}\ln(3/2)}\ln\left(\frac{72\cdot(10kR)^{2/3}(1+\sqrt{1/2})^2\sigma^2}{\delta^{2/3}\varepsilon^2\ln(3/2)}\right)\right)\right\rceil + 1\\ &\in \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2}\ln\left(\frac{kR}{\delta}\ln\left(\frac{kR\sigma}{\varepsilon\delta}\right)\right)\right).\end{aligned}$$

□

E.2 Stochastic Preference Winner Feedback: Proof of Corollary 5.2

The following two lemmata serve as a preparation for the proof of Corollary 5.2. But first let us introduce the $(k-1)$-simplex

$$\mathcal{S}_k = \left\{(p_i)_{i\in[k]} \in [0,1]^k \,:\, \sum\nolimits_{i=1}^k p_i = 1 \wedge \forall i : p_i \geq 0\right\}.$$

Lemma E.4 (Dvoretzky-Kiefer-Wolfowitz inequality for categorical random variables)**.** *Let $\{X_t\}_{t\in\mathbb{N}}$ be a sequence of iid random variables $X_t \sim \mathrm{Cat}(\mathbf{p})$ for some $\mathbf{p} \in \mathcal{S}_k$. For $t \in \mathbb{N}$ let $\hat{\mathbf{p}}^t$ be the corresponding empirical distribution after the t observations $X_1,\dots,X_t$, i.e., $\hat{p}^t_i = \frac{1}{t}\sum_{s=1}^t \mathbf{1}_{\{X_s=i\}}$ for all $i \in [k]$. Then, we have for any $\varepsilon > 0$ and $t \in \mathbb{N}$ the estimate*

$$\mathbb{P}\left(\left|\left|\hat{\mathbf{p}}^t - \mathbf{p}\right|\right|_\infty > \varepsilon\right) \leq 4e^{-t\varepsilon^2/2}.$$

Proof. Confer [19, 35] as well as Theorem 11.6 in [29]. Moreover, note that the cumulative distribution functions F resp. $\hat{F}^t$ of $X_1 \sim \mathrm{Cat}(\mathbf{p})$ resp. $\hat{\mathbf{p}}^t$ fulfill $p_j = F(j) - F(j-1)$ and $\hat{p}^t_j = \hat{F}^t(j) - \hat{F}^t(j-1)$ and thus

$$|\hat{p}^t_j - p_j| \leq |\hat{F}^t(j) - F(j)| + |\hat{F}^t(j-1) - F(j-1)|.$$

for each $j \in [k]$. □

Lemma E.5. *For every $\beta \in [1, e/2]$, $c_1, c_2 > 0$ the number*

$$x \coloneqq \frac{\beta}{c_1}\left(\ln\left(\frac{c_2 e}{c_1^\beta}\right) + \ln\ln\left(\frac{c_2}{c_1^\beta}\right)\right)$$

fulfills $c_1 x \geq \ln(c_2 x^\beta)$.

Proof. This is Lemma 18 in [20]. □

Proof of Corollary 5.2. For $t \in \mathbb{N}$ and $\gamma \in (0,1)$ define

$$c_\gamma(t) \coloneqq \sqrt{\frac{4\ln(2\pi^2 t^2/(3\gamma))}{t}}$$

and note that, for any fixed γ, the function $c_\gamma : \mathbb{N} \to (0,\infty), t \mapsto c_\gamma(t)$ is strictly monotonically decreasing with $\lim_{t\to\infty} c_\gamma(t) = 0$. For any $\alpha > 0$, $\gamma \in (0,1)$, we obtain via Lemma E.5 with the choices $\beta = 1$, $c_1 = \frac{\alpha^2}{8}$ and $c_2 = \sqrt{2/(3\gamma)}\pi$ the estimate

$$\begin{aligned} c_\gamma^{-1}(\alpha) &= \min\left\{t \in \mathbb{N} \,:\, 4\ln(2\pi^2 t^2/(3\gamma)) \leq t\alpha^2\right\} \\ &= \min\left\{t \in \mathbb{N} \,:\, \ln\left(\sqrt{2/(3\gamma)}\pi t\right) \leq \frac{\alpha^2}{8} t\right\} \\ &\leq \left\lceil \frac{8}{\alpha^2}\left(\ln\left(\frac{8\sqrt{2/(3\gamma)}\pi e}{\alpha^2}\right) + \ln\ln\left(\frac{8\sqrt{2/(3\gamma)}\pi}{\alpha^2}\right)\right)\right\rceil + 1. \end{aligned}$$

Now, suppose $\max_{Q\in\mathcal{Q}_{\leq k}(i^*)} \Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$ and that Algorithm 1 is started with a budget B larger than

$$c_{\delta/R}^{-1}(\varepsilon/2) \cdot R \max_{r\in[R]} P_r.$$

Recall that in this preference-based setting we use as the statistic the empirical mean of the (winner) observations we obtained for arm i after querying Q (with $i \in Q$) for t many times. In particular, we set

$$s_{i|Q}(t) = \frac{w_{i|Q}(t)}{t} = \frac{1}{t}\sum\nolimits_{t'=1}^{t} o_{i|Q}(t'),$$

where $o_{i|Q}(t') = 1$ if arm i is the preferred (or winning) arm among the arms in Q, if Q is queried for the t'-th time, and 0 otherwise. Thus, $w_{i|Q}(t)$ is the total number of times arm i has won in the query set Q after t queries. Moreover, $\gamma_{i|Q}(t) = |s_{i|Q}(t) - S_{i|Q}|$, where $S_{i|Q} = p_{i|Q}$ and $o_{i|Q}(t') \sim \mathrm{Cat}(\mathbf{p_Q})$. With this, we obtain for any $t \in \mathbb{N}$ and any possible sequence of partitions $(E_r)_{r\in[R]} \in (\mathcal{Q}_{\leq k})^R$ with $\mathbb{P}(\mathbb{A}_r(i^*) = E_r\ \forall r \in [R]) > 0$ that

$$\begin{aligned} &\mathbb{P}\left(\exists r \in [R] : \gamma_{i|E_r}(t) \geq c_{\delta/R}(t) \,\middle|\, \mathbb{A}_r(i^*) = E_r\ \forall r \in [R]\right) \\ &\leq \sum_{r\in[R]} \mathbb{P}\left(\gamma_{i|E_r}(t) \geq c_{\delta/R}(t) \,\middle|\, \mathbb{A}_r(i^*) = E_r\ \forall r \in [R]\right) \\ &= \sum_{r\in[R]} \mathbb{P}\left(\max\nolimits_{i\in E_r} \left|\frac{1}{t}\sum_{t'=1}^{t} o_{i|E_r}(t') - S_{i|E_r}\right| > \sqrt{\frac{4\ln(2\pi^2 t^2/(3\gamma))}{t}} \,\middle|\, \mathbb{A}_r(i^*) = E_r\ \forall r \in [R]\right) \\ &\leq \frac{6\delta}{\pi^2 t^2}, \end{aligned}$$

where we used Lemma E.4 in the last inequality. Using the law of total probability for all possible sequences of partitions $(E_r)_{r\in[R]}$, we see that the event

$$\mathcal{E} \coloneqq \left\{\exists t \in \mathbb{N}, r \in [R] : \overline{\gamma}_{\mathbb{A}_r(i^*)}(t) \geq c_{\delta/R}(t)\right\}$$

occurs with probability

$$\begin{aligned} \mathbb{P}(\mathcal{E}) &\leq \sum_{t\in\mathbb{N}} \sum_{(E_r)_{r\in[R]}} \mathbb{P}\left(\exists r \in [R] : \overline{\gamma}_{E_r}(t) \geq c_{\delta/R}(t) \,\middle|\, \mathbb{A}_r(i^*) = E_r\ \forall r \in [R]\right) \\ &\quad \times\ \mathbb{P}(\mathbb{A}_r(i^*) = E_r\ \forall r \in [R]) \\ &\leq \sum\nolimits_{t\in\mathbb{N}} \frac{6\delta}{\pi^2 t^2} \sum\nolimits_{(E_r)_{r\in[R]}} \mathbb{P}(\mathbb{A}_r(i^*) = E_r\ \forall r \in [R]) \leq \delta. \end{aligned}$$

On $\mathcal{E}^c$ we have $\overline{\gamma}_{\mathbb{A}_r(i^*)}(t) < c_{\delta/R}(t)$ for all $t \in \mathbb{N}, r \in [R]$ and thus in particular $\overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}(\alpha) \geq c^{-1}_{\delta/R}(\alpha)$ for any $\alpha \in (0,\infty)$. Since $\max_{Q\in\mathcal{Q}_{\leq k}(i^*)} \Delta_{(f(|Q|)+1)|Q} \leq \varepsilon$, Theorem 4.1 thus lets us conclude

$$\begin{aligned} \mathbb{P}\left(\text{Alg. 1 returns } i^*\right) &\geq \mathbb{P}\left(B > R\max\nolimits_{r\in[R]} P_r \overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\frac{\Delta_{(f(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right)\right) \\ &\geq \mathbb{P}\left(\left\{B > R\max\nolimits_{r\in[R]} P_r \overline{\gamma}^{-1}_{\mathbb{A}_r(i^*)}(\varepsilon/2)\right\} \cap \mathcal{E}^c\right) \\ &\geq \mathbb{P}\left(\left\{B > R\max\nolimits_{r\in[R]} P_r c^{-1}_{\delta/R}(\varepsilon/2)\right\} \cap \mathcal{E}^c\right) \\ &= \mathbb{P}(\mathcal{E}^c) \geq 1 - \delta, \end{aligned}$$

where the equality holds due to the assumption on B. Consequently, the statement holds with

$$\begin{aligned} C(\delta, \varepsilon, k, R) &\coloneqq c_{\delta/R}^{-1}(\varepsilon/2) \\ &\leq \left\lceil \frac{32}{\varepsilon^2} \left(\ln \left(\frac{32\sqrt{2R/(3\delta)}\pi e}{\varepsilon^2} \right) + \ln \ln \left(\frac{32\sqrt{2R/(3\delta)}\pi}{\varepsilon^2} \right) \right) \right\rceil + 1 \\ &\in \mathcal{O}\left(\frac{1}{\varepsilon^2} \ln \left(\frac{R}{\delta \varepsilon^4} \right) \right). \end{aligned}$$

□

F Comparisons of the Algorithms

In the following we summarize the theoretical results obtained for our proposed algorithms in a concise way. First of all, we give an overview of the individual key quantities of each algorithm in Table 2, where we assume w.l.o.g. that $\binom{n}{k}$ is a divisor of B in RoundRobin to make the assignments of R, P_r and $f(s)$ for RoundRobin well-defined. The maximal number of different query sets is derived in Section F.1.

Table 2: Comparison of the maximal number of rounds, the maximal number of partitions per round, the amount of retained arms from each partition and the maximal number of query sets for RoundRobin and our proposed algorithms CSWS, CSR and CSH.

Alg.	R	P_r	$f(x)$	$\max \#$query_sets
RoundRobin	1	$\binom{n}{k}$	$x \mapsto x$	$\binom{n}{k}$
CSWS	$\lceil \log_k(n) \rceil + 1$	$\lceil \frac{n}{k^r} \rceil$	$x \mapsto 1$	$R^{CSWS} + n \cdot \left(\frac{1-1/k^{\lceil \log_k(n) \rceil + 1}}{k-1} \right)$
CSR	$\left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1$	$\left\lceil \frac{n(1-\frac{1}{k})^{(r-1)}}{k} \right\rceil$	$x \mapsto x - 1$	$R^{CSR} + n \left(1 - \left(1 - \frac{1}{k} \right)^{\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \rceil + k - 1} \right)$
CSH	$\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil$	$\lceil \frac{n}{2^{r-1}k} \rceil$	$x \mapsto \lceil \frac{x}{2} \rceil$	$R^{CSH} + \frac{2n}{k} \left(1 - 1/2^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil} \right)$

Using Remark D.1 we can derive the following sufficient budgets of the algorithms summarized in the following table, where $\pi(Q) \in Q$ be the $\lfloor \frac{|Q|}{2} \rfloor + 1$- th best arm with respect to $(S_{i|Q})_{i \in Q}$.

Table 3: Comparison of the sufficient budget for RoundRobin and our proposed algorithms CSWS, CSR and CSH.

Algorithm	Sufficient budget
RoundRobin	$\binom{n}{k} \max_{i \in \mathcal{A}, i \neq i^*_{\mathcal{B}}} \left(\hat{\gamma}^{\max}_{i^*_{\mathcal{B}}, i} \right)^{-1} \left(\frac{S^{\mathcal{B}}_{i^*_{\mathcal{B}}} - S^{\mathcal{B}}_i}{2} \right)$
CSWS	$\lceil \frac{n}{k} \rceil \left(\lceil \log_k(n) \rceil + 1 \right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \max_{i \in Q \setminus \{i^*\}} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\|Q} - S_{i\|Q}}{2} \right) \right\rceil$
CSR	$\lceil \frac{n}{k} \rceil \left(\left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 \right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \min_{i \in Q \setminus \{i^*\}} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\|Q} - S_{i\|Q}}{2} \right) \right\rceil$
CSH	$\lceil \frac{n}{k} \rceil \left(\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil \right) \cdot \max_{Q \in \mathcal{Q}_{\leq k} : i^* \in Q} \left\lceil \bar{\gamma}^{-1} \left(\frac{S_{i^*\|Q} - S_{\pi(Q)\|Q}}{2} \right) \right\rceil$

In Section F.2 we compare these quantities for the special case, in which the gaps $\Delta_{i|Q} = S_{i^*|Q} - S_{i|Q}$ are all equal to some $\Delta > 0$, while in Section F.3 we derive the sufficient budgets resulting from Corollaries 5.1 and 5.2 for the reward setting and preference-based setting, respectively, to return the best arm with high probability in the stochastic setting. Note that if $\gamma_{i|Q}(t) = \gamma(t)$ and $S_{(2)|Q} = \cdots = S_{(|Q|)|Q}$ are fulfilled for all $Q \in \mathcal{Q}_{\leq k}, i \in Q$ and $t \in \mathbb{N}$, then the lower bound in Theorem 3.1 (i) matches the above upper bound for CSWS up to a factor $C = \lceil \log_k(n) \rceil + 1$.

F.1 Maximal Number of Different Query Sets

The maximal number of required query sets for each algorithm is $\sum_{r=1}^{R} P_r$. Note that this is a geometric series and thus the partial sum can easily be computed for each of our proposed algorithms.

CSWS By using the specified valued of R and P_r for CSWS, we obtain that the number of different query set is at most

$$\begin{aligned}
\sum_{r=1}^{R^{CSWS}} P_r^{CSWS} &= \sum_{r=1}^{\lceil \log_k(n) \rceil + 1} \left\lceil \frac{n}{k^r} \right\rceil \\
&\leq \lceil \log_k(n) \rceil + 1 + \sum_{r=1}^{\lceil \log_k(n) \rceil + 1} \frac{n}{k^r} \\
&= \lceil \log_k(n) \rceil + 1 + n \cdot \left(\sum_{r=0}^{\lceil \log_k(n) \rceil + 1} \left(\frac{1}{k} \right)^r - 1 \right) \\
&= \lceil \log_k(n) \rceil + 1 + n \cdot \left(\frac{1 - 1/k^{\lceil \log_k(n) \rceil + 2}}{1 - 1/k} - 1 \right) \\
&= \lceil \log_k(n) \rceil + 1 + n \cdot \left(\frac{1 - 1/k^{\lceil \log_k(n) \rceil + 1}}{k - 1} \right),
\end{aligned}$$

where we used for the inequality that $\lceil x \rceil \leq x + 1$ for any $x \in \mathbb{R}$.

CSR For CSR we get as an upper bound on the number of different query sets:

$$\begin{aligned}
\sum_{r=1}^{R^{CSR}} P_r^{CSR} &= \sum_{r=1}^{\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \rceil + k - 1} \left\lceil \frac{n \left(1 - \frac{1}{k}\right)^{r-1}}{k} \right\rceil \\
&\leq \left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 + \sum_{r=1}^{\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \rceil + k - 1} \frac{n \left(1 - \frac{1}{k}\right)^{r-1}}{k} \\
&= \left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 + \frac{n}{k} \sum_{r=0}^{\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \rceil + k - 2} \left(1 - \frac{1}{k}\right)^r \\
&= \left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 + \frac{n}{k} \frac{\left(1 - \left(1 - \frac{1}{k}\right)^{\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \rceil + k - 1}\right)}{\left(1 - \left(1 - \frac{1}{k}\right)\right)} \\
&= \left\lceil \log_{1-\frac{1}{k}} \left(\frac{1}{n} \right) \right\rceil + k - 1 + n \left(1 - \left(1 - \frac{1}{k}\right)^{\lceil \log_{1-\frac{1}{k}}\left(\frac{1}{n}\right) \rceil + k - 1}\right).
\end{aligned}$$

CSH Similarly, we can obtain for CSH the following maximum number of different query sets:

$$\begin{aligned}
\sum_{r=1}^{R^{CSH}} P_r^{CSH} &= \sum_{r=1}^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil} \left\lceil \frac{n}{2^{r-1}k} \right\rceil \\
&\leq \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil + \sum_{r=1}^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil} \frac{n}{2^{r-1}k} \\
&= \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil + \frac{n}{k} \sum_{r=0}^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil - 1} \left(\frac{1}{2}\right)^r \\
&= \lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil + \frac{2n}{k} \left(1 - 1/2^{\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil}\right).
\end{aligned}$$

These upper bounds on the maximum number of different query sets are summarized in Table 2. Note that RoundRobin by design queries the possible query sets of $\mathcal{Q}_{=k}$ in a round-robin fashion, so that the number of different query sets is indeed $|\mathcal{Q}_{=k}| = \binom{n}{k}$.

F.2 Comparison of Sufficient Budgets

In order to compare the derived sufficient budgets of the different algorithms (see Table 3), we consider in the following the setting where the generalized Condorcet winner coincides with the generalized Borda winner. In addition we assume that the limit statistic $S_{i|Q}$ for each arm $i \in \mathcal{A}$ has always the same difference to the limit of the optimal arm $S_{i^*|Q}$ if $i^* \in Q$. More precisely, for each arms $i \in \mathcal{A}$ and each query set $Q \in \mathcal{Q}_{\leq k}$ we have $\Delta_{i|Q} = \Delta$ for some fixed $\Delta > 0$. In this way, the γ-dependent term present in the sufficient budget for each algorithm is simply $\left\lceil \bar{\gamma}^{-1}\left(\frac{\Delta}{2}\right)\right\rceil$. As a consequence, we can neglect this term as it has no influence on the differences in the desired budgets for the various algorithms and the remaining term based on the product of the number of rounds, i.e. R, and the number of partitions in round 1, i.e. P_1, is driving the (rough) sufficient budget bounds (see Table 2). However, the number of partitions in round 1 is the same for all algorithms, so that we can neglect this term as well. With a slight abuse of denotation, we refer to this remainder term simply as the sufficient budget in the following. With these considerations, it is easy to see that RoundRobin requires the highest sufficient budget even for moderate sizes of n if k is sufficiently lower than n. To get an impression how the sufficient budget behaves for the more sophisticated algorithms based on the successive elimination strategy, we plot these in Figure 2 as curves depending on the number of arms n for different subset sizes k. Note, that in contrast to CSWS and CSH, the sufficient budget of CSR is higher for bigger subset sizes k, since only a smaller proportion of all arms is discarded after each round. In the case $k = 2$ the number of rounds are all the same, so that consequently the sufficient budget is the same for all three algorithms.

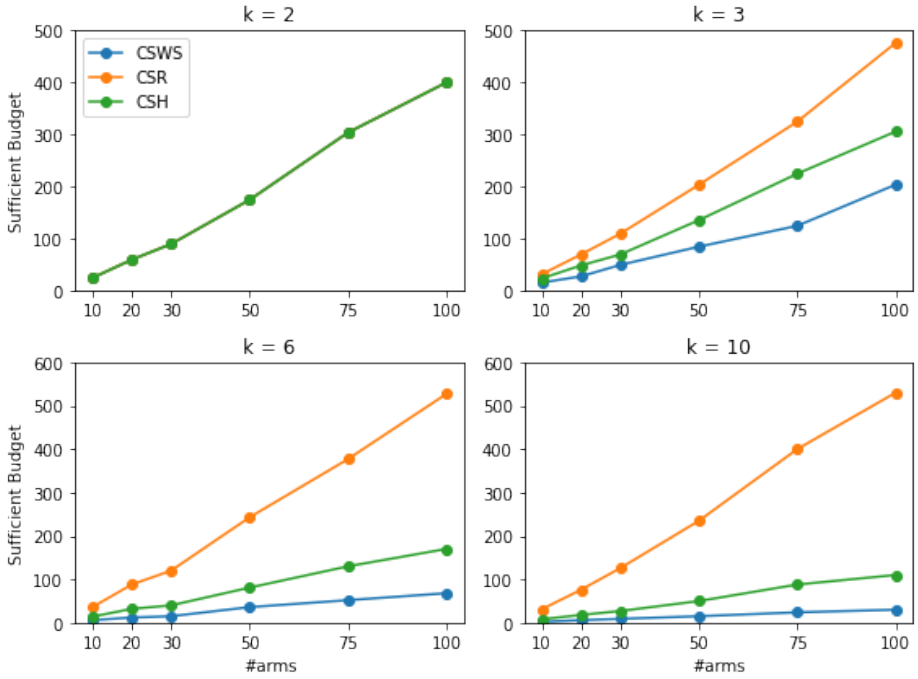


Figure 2: Comparison of required budget for our proposed algorithms for different values of the number of arms n and the subset size k.

F.3 Applications to Stochastic Settings

In Table 4 the sufficient budgets for our proposed algorithms in the stochastic setting with reward feedback and preference-based feedback are listed. Note, that these results are simply derived by applying Corollary 5.1 and resp. Corollary 5.2 with the specific instantiations of R and P_r for our algorithms (see Tables 2 and 3).

Table 4: Comparison of the sufficient budgets for our proposed algorithms CSWS, CSR and CSH in the reward and preference-based setting.

Alg.	Budget in reward setting
CSWS	$\frac{1}{\epsilon^2}\ln\left(\frac{k(\lceil\log_k(n)\rceil+1)}{\delta}\ln\left(\frac{k(\lceil\log_k(n)\rceil+1)}{\epsilon\delta}\right)\right)\cdot(\lceil\log_k(n)\rceil+1)\left\lceil\frac{n}{k}\right\rceil$
CSR	$\frac{1}{\epsilon^2}\ln\left(\frac{k\left(\left\lceil\log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil+k-1\right)}{\delta}\ln\left(\frac{k\left(\left\lceil\log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil+k-1\right)}{\epsilon\delta}\right)\right)\cdot\left(\left\lceil\log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil+k-1\right)\left\lceil\frac{n}{k}\right\rceil$
CSH	$\frac{1}{\epsilon^2}\ln\left(\frac{k(\lceil\log_2(n)\rceil+\lceil\log_2(k)\rceil)}{\delta}\ln\left(\frac{k(\lceil\log_2(n)\rceil+\lceil\log_2(k)\rceil)}{\epsilon\delta}\right)\right)\cdot(\lceil\log_2(n)\rceil+\lceil\log_2(k)\rceil)\left\lceil\frac{n}{k}\right\rceil$
Alg.	Budget in preference-based setting
CSWS	$\frac{1}{\epsilon^2}\ln\left(\frac{\lceil\log_k(n)\rceil+1}{\delta\epsilon^4}\right)\cdot(\lceil\log_k(n)\rceil+1)\left\lceil\frac{n}{k}\right\rceil$
CSR	$\frac{1}{\epsilon^2}\ln\left(\frac{\left\lceil\log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil+k-1}{\delta\epsilon^4}\right)\cdot\left(\left\lceil\log_{1-\frac{1}{k}}\left(\frac{1}{n}\right)\right\rceil+k-1\right)\left\lceil\frac{n}{k}\right\rceil$
CSH	$\frac{1}{\epsilon^2}\ln\left(\frac{\lceil\log_2(n)\rceil+\lceil\log_2(k)\rceil}{\delta\epsilon^4}\right)\cdot(\lceil\log_2(n)\rceil+\lceil\log_2(k)\rceil)\left\lceil\frac{n}{k}\right\rceil$

G Further Experiments

In the following, we present some further experiments comparing our proposed algorithms with each other on synthetic data including a detailed description of the data generation and the experiment setting.

G.1 Synthetic Data

For each $Q \in \mathcal{Q}_{\leq k}$ with $Q = \{i_1, \dots, i_{|Q|}\}$ we consider the case where the observation vector $\mathbf{o}_Q$ is a random sample from a multivariate Gaussian distribution with mean $\mu_Q = (\mu_{i_1|Q}, \dots, \mu_{i_{|Q|}|Q})^\top$ and a diagonal covariance matrix $\mathrm{diag}(\sigma_{i_1|Q}, \dots, \sigma_{i_{|Q|}|Q})$. Here, $\mu_{i_j|Q}$ are values in $[0, 1]$ for $i_j \neq i^*$ and $\sigma_{i|Q}$ in $[0.05, 0.2]$ (all randomly sampled). For any Q with $i^* \in Q$ we set $\mu_{i^*|Q} = \max_{j \in Q, j \neq i^*} \mu_{j|Q} + \varepsilon$ for some $\varepsilon > 0$, which ensures (A2) to hold for the expected values. In our experiments we always use a value of $\varepsilon = 0.1$. In the following we vary the values of $n \in \{50, 100\}$, $k \in \{2, 4, 6, 8, 10\}$ and $B \in \{50, 100, 200, 300, 500\}$.

We consider a reward setting and use the empirical mean as the statistic (see Section 5). We do not force the generalized Borda winner to be the same as the generalized Condorcet winner, but they naturally coincidence in most of the runs by sampling the observation vector as defined above.

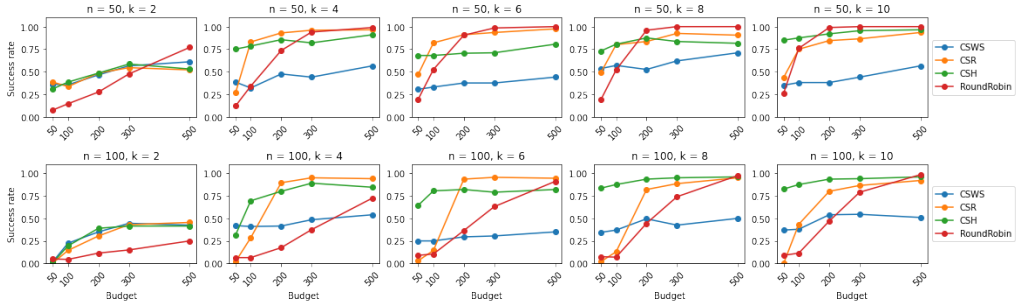


Figure 3: Success rates of our proposed algorithms for varying n, k and budget B in the reward setting.

The success rates of our proposed algorithms for identifying i^* given a budget B are shown in Figure 3. It is visible, that in particular for the challenging scenario, where the budget B and the subset size k are small and the number of arms n is large, both CSH and CSR perform well. Especially CSH has overall a solid performance.

Reward setting. In contrast to the experiments with reward feedback shown in the main paper, we try in the following experiments to force the generalized Borda winner to be different from the generalized Condorcet winner. For this purpose, we fix one random arm $i^*_{\mathcal{B}} \in [n] \backslash \{i^*\}$ as the prospective generalized Borda winner and set its expected value to $\mu_{i^*_{\mathcal{B}}|Q} = \max_{j \in Q, j \neq i^*_{\mathcal{B}}} \mu_{j|Q} + 2\epsilon$ for any $Q \in \mathcal{Q}_{\leq k}$ with $i^*_{\mathcal{B}} \in Q$ and $i^* \notin Q$. Thus, $i^*_{\mathcal{B}}$ is likely the generalized Borda winner and is different from the generalized Condorcet winner. Since our goal is to find the generalized Condorcet winner i^*, RoundRobin will probably fail most of the times in finding i^*. This is due to the fact that RoundRobin focuses on identifying $i^*_{\mathcal{B}}$, i.e., the the generalized Borda winner, which, however, does not coincidence with the generalized Condorcet winner i^*.

This suspicion is confirmed by the results of the experiments shown in Figure 4 illustrating the empirical success rates for finding the generalized Condorcet winner in the setting described above. Except for some cases where the subset size k is relatively large in comparison to the total number of arms, such that the generalized Condorcet winner is already contained in most of the seen subsets and hence is automatically also the generalized Borda winner, RoundRobin performs poorly in finding the generalized Condorcet winner and is always outperformed by the algorithms based on the combinatorial successive elimination strategy in Section 4.1.

Preference-based setting with different GCW and GBW. In the preference-based setting we ignore the explicit numerical values of the observation vector and only use the information which

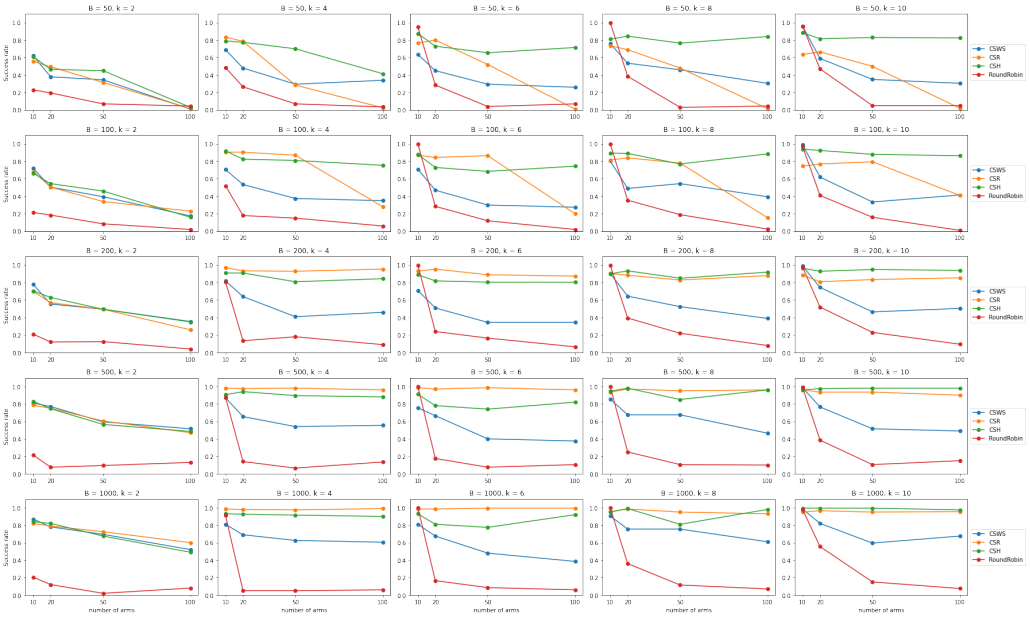


Figure 4: Success rates of our proposed algorithms for varying n, k and budget B in the reward setting with different generalized Condorcet winner and generalized Borda winner.

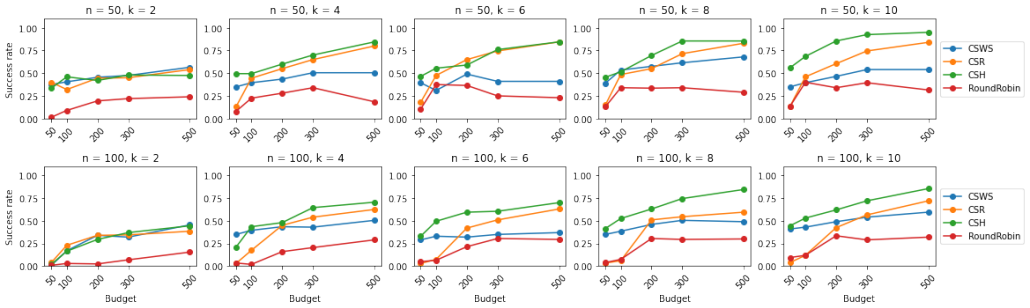


Figure 5: Success rates of our proposed algorithms for varying n, k and budget B in the preference-based setting with different generalized Condorcet winner and generalized Borda winner.

arm was (not) the winner, i.e., which had (not) the highest observation value in the query set used , formally $s_{i_j|Q}(t) = \frac{1}{t}\sum_{s=1}^{t} \mathbb{1}\{o_{i_j|Q}(s) = \max_{i=i_1,\dots i_{|Q|}} o_{i|Q}(s)\}$. Additionally, we fix one arm $i^*_{\mathcal{B}} \in [n]\backslash\{i^*\}$ and set $\mu_{i^*_{\mathcal{B}}|Q} = \max_{j\in Q, j\neq i^*_{\mathcal{B}}} \mu_{j|Q} + 2\varepsilon$ for any Q with $i^*_{\mathcal{B}} \in Q$ and $i^* \notin Q$. In this way, $i^*_{\mathcal{B}}$ is the generalized Borda winner and different from i^*.

The success rates of our proposed algorithms for identifying i^* in this setting are shown in Figure 5. As expected our methods outperform ROUNDROBIN in all scenarios.

Preference-based setting. We now investigate the case, in which we do not force the generalized Borda winner and the generalized Condorcet winner to be different, thus they will naturally coincidence in most of the cases. This is achieved by considering the problem configuration as in the reward setting specified in Section 6, and ignoring the explicit numerical values (as in the preference-setting above).

The resulting success rates for finding the generalized Condorcet winner illustrated in Figure 6 are similar to the results in the reward setting for matching generalized Condorcet winner and generalized

Borda winner. This means that, in particular, when the budget is small, the number of arms is large and the subset size is small, the algorithms following the combinatorial successive elimination strategy outperform RoundRobin. Note that this setting is arguably the most relevant setup for practical applications. Moreover, Figure 6 illustrates the natural effect one would expect for the number of arms n on success rates, namely that success rates decrease with a larger number of arms.

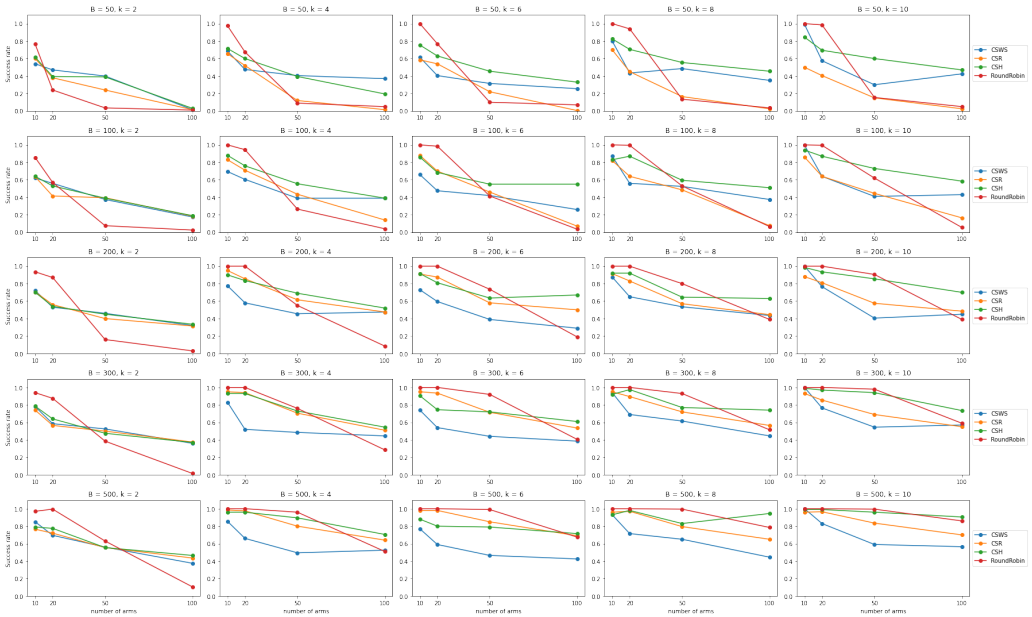


Figure 6: Success rates of our proposed algorithms for varying n, k and budget B in the preference-based setting with (mostly) matching generalized Condorcet winner and generalized Borda winner.

G.2 Statistics beyond the Arithmetic Mean

We consider in the following the reward setting, where each observation is random sampled from the following distribution

$$\boldsymbol{o}_Q(t) \sim \mathcal{N}\left(\begin{pmatrix} \mu_{1|Q} \\ \vdots \\ \mu_{|Q||Q} \end{pmatrix}, \begin{pmatrix} \sigma_{1|Q} \\ \vdots \\ \sigma_{|Q||Q} \end{pmatrix}\right)$$

for $\mu_{i|Q}$ is sampled randomly from $[0, 1]$ and $\sigma_{i|Q}$ from $[0.05, 0.2]$ for each arm $i \in Q$.

Median An alternative to the arithmetic mean would be to measure the quality of the arms by the median of the seen observations. In particular, when the observations are prone to outliers, the median provides a more robust statistic: $s_{i|Q}(t) = \textsc{Median}(o_{i|Q}(1), \dots, o_{i|Q}(t))$ for each arm $i \in Q$. The results for this setting are illustrated in Figure 7.

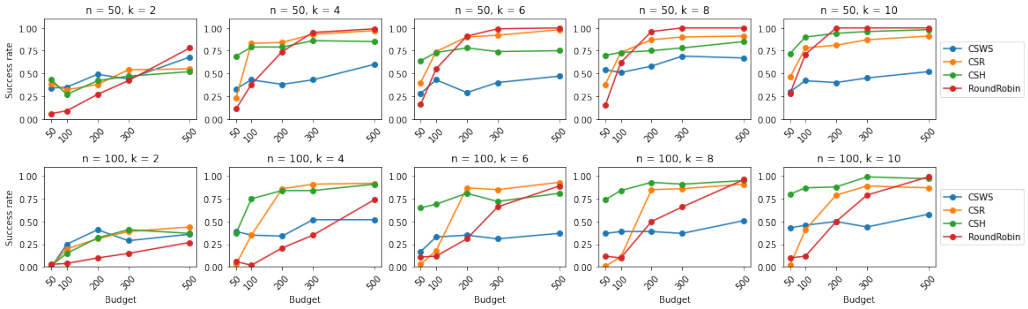


Figure 7: Success rates of our proposed algorithms for varying n, k and budget B in the rewards setting with (mostly) matching generalized Condorcet winner and generalized Borda winner and using the median as the statistic.

Power-Mean Another possibility is to use the so called power-mean, which is a compromise between the maximum and the arithmetic mean for a (multi)set of observations. Since the arithmetic mean is known to underestimate the true quality of an arm, while the maximum overestimates it, the power mean is often a good compromise, as it lies between the two. It is defined by $s_{i|Q}(t) = \left(\frac{1}{t}\sum_{t'=1}^{t} o_{i|Q}(t')^q\right)^{1/q}$ for each arm $i \in Q$ and a fixed $q \in \mathbb{N}$. We use in the following $q = 2$. The results for this setting are illustrated in Figure 8.

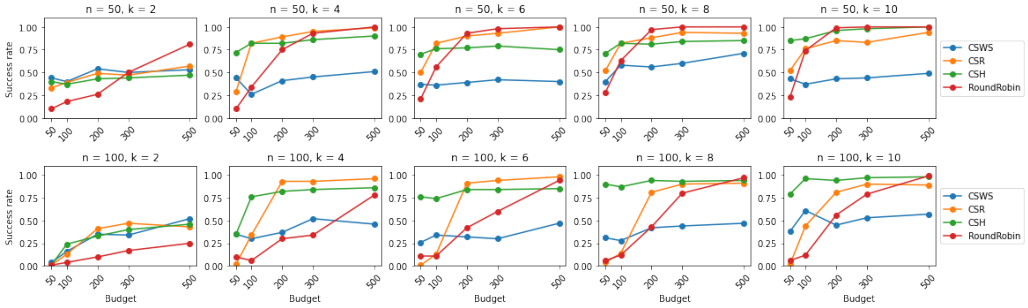


Figure 8: Success rates of our proposed algorithms for varying n, k and budget B in the rewards setting with (mostly) matching generalized Condorcet winner and generalized Borda winner and using the power mean as the statistic.

B

Appendix to AC-Band: A Combinatorial Bandit-Based Approach to Algorithm Configuration

A List of Symbols

The following table contains a list of symbols that are frequently used in the main paper as well as in the following supplementary material.

	Basics
$\mathbb{1}\{\cdot\}$	Indicator function
$\mathbb{N}$	Set of natural numbers (without 0), i.e., $\mathbb{N} = \{1, 2, 3, \dots\}$
$\mathbb{R}$	Set of real numbers
$\mathcal{I}$	Space of problem instances
$\mathcal{P}$	Probability distribution over instance space $\mathcal{I}$
$\mathcal{A}$	Target algorithm that can be run on problem instance $i \in \mathcal{I}$
Θ	Configuration search space consisting all feasible parameter configurations
$\mathcal{A}_\theta$	Instantiation of $\mathcal{A}$ with configuration θ
$c(i, \theta)$	$c : \mathcal{I} \times \Theta \to \mathbb{R}$ costs of using $\mathcal{A}_\theta$ on problem instance $i \in \mathcal{I}$
$\mathcal{AC}$	Algorithm configurator
k	Maximal possible subset size
B	Budget for the learner (algorithm configurator)
$\Theta_{[2,k]}$	All subsets of Θ of size at least 2 and at most k: $\{\tilde{\Theta} \subseteq \Theta \mid 2 \leq \lvert\tilde{\Theta}\rvert \leq k\}$
$\Theta_{[2,k]}(\theta)$	All subsets in $\Theta_{[2,k]}$ which contain configuration θ: $\{\tilde{\Theta} \in \Theta_{[2,k]} \mid \theta \in \tilde{\Theta}\}$
$c_{\tilde{\Theta}}$	Cost of running θ on problem instance $i \in \mathcal{I}$ in parallel with configurations $\theta' \in \tilde{\Theta} \in \Theta_{[2,k]} \backslash \{\theta\}$
	Modelling related
s	Statistic mapping costs to a numerical value: $s : \cup_{t \in \mathbb{N}} \mathbb{R}^t \to \mathbb{R}$, $(c(i_1, \theta), \dots, c(i_t, \theta)) \mapsto s(c(i_1, \theta), \dots, c(i_t, \theta))$
$s_{\theta\lvert\tilde{\Theta}}(t)$	$s_{\theta\lvert\tilde{\Theta}}(t) = s(c_{\tilde{\Theta}}(i_1, \theta), \dots, c_{\tilde{\Theta}}(i_t, \theta))$ statistic of $\theta \in \Theta$ after running $\tilde{\Theta}$ in parallel
$S_{\theta\lvert\tilde{\Theta}}$	Limit of the statistics for configuration θ in query set $\tilde{\Theta}$, i.e., $\lim_{t\to\infty} s_{\theta\lvert\tilde{\Theta}}(t)$ (see Assumption (A1))
ϵ	Near-optimality parameter, $\epsilon \in (0, 1)$
α	Proportion of ϵ-best configurations in Θ, $\alpha \in (0, 1)$ (see Assumption (A2))
δ	Fixed error probability for identifying an ϵ-best configuration, $\delta \in (0, 1)$
$N_{\alpha,\delta}$	$N_{\alpha,\delta} = \lceil \log_{1-\alpha}(\delta) \rceil$ number of configurations that have to be sampled to ensure that at least one ϵ-best configuration is contained with probability at least $1 - \delta$
$\gamma_{\theta\lvert\tilde{\Theta}}$	Pointwise minimal function such that $\lvert s_{\theta\lvert\tilde{\Theta}}(t) - S_{\theta\lvert\tilde{\Theta}}\rvert \leq \gamma_{\theta\lvert\tilde{\Theta}}(t)$ for all t
$\gamma^{-1}_{\theta\lvert\tilde{\Theta}}(t)$	$\min\{t' \in \mathbb{N} \mid \lvert s_{\theta\lvert\tilde{\Theta}}(t') - S_{\theta\lvert\tilde{\Theta}}\rvert \leq t\}$ (exists due to Assumption (A1))
$\bar{\gamma}^{-1}_{\tilde{\Theta}}(t)$	Minimal $\gamma^{-1}_{\theta\lvert\tilde{\Theta}}(t) = \min\{t' \in \mathbb{N} \mid \lvert s_{\theta\lvert\tilde{\Theta}}(t') - S_{\theta\lvert\tilde{\Theta}}\rvert \leq t\}$ over all $\theta \in \tilde{\Theta}$ (exists due to Assumption (A1))
$S_{(l)\lvert\tilde{\Theta}}$, $\Delta_{(l)\lvert\tilde{\Theta}}$	l-th order statistic of $\{S_{\theta\lvert\tilde{\Theta}}\}_{\theta\in\tilde{\Theta}}$ for $l \in \{1, 2, \dots, \lvert\tilde{\Theta}\rvert\}$ and its gap $\Delta_{(l)\lvert\tilde{\Theta}} = S_{\theta^*\lvert\tilde{\Theta}} - S_{(l)\lvert\tilde{\Theta}}$
	Algorithm related
CSE	The generic *combinatorial successive elimination* algorithm (Algorithm 1)
f_ρ	Function from $[k]$ to $[k]$, $f_\rho(x) = \lfloor x/2^\rho \rfloor$ for a $\rho \in (0, \log_2(k)]$ specifying the nature of the configuration elimination strategy
n_0	Variable to specify the initial sample size, $n_0 \in (N_{\alpha,\delta}, 2N_{\alpha,\delta}]$
n_e	$n_e = \lceil n_0/2^e \rceil + 1$ number of considered configurations in epoch $e \in [E]$
E	Number of epochs $E = \lceil \log_2(n_0/n_0 - N_{\alpha,\delta}) \rceil$
C_1	Internal constant variable for AC-Band, $C_1 = \log_{1+\frac{k-1}{E}}(2)$
C_2	Internal constant variable for AC-Band, $C_2 = 1 + \log_{1+\frac{k-1}{E}}\left(n_0 + 4\frac{n_0}{n_0 - N_{\alpha,\delta}}\right)$
C_3	Internal constant variable for AC-Band, $C_3 = \left\lceil \log_{1+\frac{k-1}{E}}(k) \right\rceil$
c_e	Internal variable for AC-Band in epoch $e \in [E]$: $c_e = \frac{(C_1 E - (2^E - 1)(2C_1 - C_2 - C_3))2^e}{2^E(-eC_1 + C_2 + C_3)}$
B_e	Budget for call of CSE in epoch $e \in [E]$: $B_e = B/c_e$ for an overall budget of B for AC-Band
R^{ρ_e,k,n_e}	Number of rounds in CSE in epoch $e \in [E]$
$P_r^{\rho_e,k,n_e}$	Number of partitions in CSE in epoch $e \in [E]$ and round $r \in [R^{\rho_e,k,n_e}]$
$\mathbb{A}_r(\theta)$	The partition in round r of CSE containing θ (emptyset otherwise)
b_r	Budget used in round r of CSE for a partition

B Extension of CSE for Finding ϵ-best Arms

B.1 Adjustments of Algorithm Parameters

We modify the definition of the function f in (Brandt et al. 2022) and thus define $f_\rho(x) = \lfloor \frac{x}{2^\rho} \rfloor$ for a $\rho \in (0, \log_2(k)]$ and obtain for

- $\rho = 1$: Combinatorial Succesive Halving (CSH) with $f_\rho(k) = \lfloor \frac{k}{2} \rfloor$
- $\rho = \log_2(k)$: Combinatorial Succesive Winner Stays (CSWS) with $f_\rho(k) = 1$
- $\rho \to 0$: Combinatorial Succesive Rejects (CSR) with $f_\rho(k) = k - 1$.

Note that for a fixed subset size k and for a fixed $\rho \in (0, \log_2(k)]$, one can derive the number of rounds and number of partitions per round as follows. The number of rounds in the first while loop of Algorithm 1 can be computed as

$$R_1^{\rho,k,n} = \{\min x \ : \ g^{(\circ x)}(n) \leq k\} \text{ for } g(x) = f_\rho(k) \cdot \left\lfloor \frac{x}{k} \right\rfloor + x \mod k,$$

where $g^{(\circ x)}$ denotes the x-times composition of g. Furthermore, it holds that $R_1^{\rho,k,n} \leq \lceil \log_{\frac{k}{f_\rho(k)}}(n) \rceil$, which we use for the sake of ease to estimate $R_1^{\rho,k,n}$ in the following in our theoretical analyses. In the second while-loop of Algorithm 1, we have only k arms left, thus we can calculate the number of rounds as

$$R_2^{\rho,k} = \left\{\min x \ : \ f_\rho^{(\circ x)}(k) \leq 1\right\}.$$

For all $k \in \mathbb{N}$, $R_2^{\rho,k} \leq \lceil \log_{\frac{k}{f_\rho(k)}}(k) \rceil$. The overall number of rounds is therefore $R^{\rho,k,n} = R_1^{\rho,k,n} + R_2^{\rho,k} \leq \lceil \log_{\frac{k}{f_\rho(k)}}(n) \rceil + \lceil \log_{\frac{k}{f_\rho(k)}}(k) \rceil$. The number of partitions per round $r \in \{1, \ldots, R^{f_\rho,k,n}\}$ is given by

$$P_r^{\rho,k,n} = \left\lfloor \frac{n}{k} \left(\frac{f_\rho(k)}{k} \right)^{r-1} \right\rfloor.$$

Thus, we can automatically compute the number of rounds and partitions in Algorithm 1 if the parameter ρ for the discard function f_ρ, the subset size k, and the number of arms n are given. In contrast to (Brandt et al. 2022), we do not need to estimate and specify R and $\{P_r\}$ by hand before we can run Algorithm 1.

B.2 Theoretical Guarantees

The first step for extending Algorithm 1 to the context of AC is to extend the theoretical guarantees to the context of finding an ϵ-best arm of a finite set of arms (configurations in our terminology) and to derive a sufficient budget for CSE that is necessary to find such an ϵ-best arm, provided that an ϵ-best arm is defined as follows.

Definition B.1. *Assume we have n arms (configurations) $\theta_1, \ldots, \theta_n$. Let θ^* be such that*

$$\forall \theta^* \in \Theta_{[2,k]}(\theta^*) \ S_{\theta^*|\tilde{\Theta}} \geq S_{(1)|\tilde{\Theta}} - \epsilon.$$

We call θ^* an ϵ-best arm.

Note that we only have a finite set of n arms (configurations) $\theta_1, \ldots, \theta_n$, which we identify simply by their indices $1, \ldots, n$ in the following. Further assume in the following that an ϵ-best arm exists. We identify this arm by i^*, and write $\Delta_{i|\tilde{\Theta}} = S_{(1)|\tilde{\Theta}} - S_{i|\tilde{\Theta}}$. In addition, let $\mathbb{A}_r$ be the set of active arms in round $r \in [R^{\rho,k,n}]$, which will then be partitioned into $\mathbb{A}_{r,1}, \ldots, \mathbb{A}_{r,P_r^{\rho,k,n}}$.

Theorem B.2 (Sufficient budget for CSE for finding an ϵ-best arm)**.** *Using Algorithm 1 with n arms, a discard function f_ρ, and a subset size k returns an arm i^*, which is an ϵ-best arm if the budget B is larger than*

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

where $\mathbb{A}_r(i^*)$ denotes the partition in round r of CSE containing i^* (or θ^*).

Proof of Theorem B.2. Step 1: Algorithm 1 never requires a number of problem instances that exceeds the budget B:

$$\sum_{r=1}^{R^{\rho,k,n}} P_r^{\rho,k,n} \cdot b_r = \sum_{r=1}^{R^{\rho,k,n}} P_r^{\rho,k,n} \cdot \left\lfloor \frac{B}{P_r^{\rho,k,n} \cdot R^{\rho,k,n}} \right\rfloor \leq \sum_{r=1}^{R^{\rho,k,n}} \frac{B}{R^{\rho,k,n}} = B.$$

Step 2: Assume in the following $B \geq z(\rho, k, n, \epsilon)$, then we have for each round $r \in [R^{\rho,k,n}]$ that

$$\begin{aligned} b_r &\geq \frac{B}{P_r^{\rho,k,n} \cdot R^{\rho,k,n}} - 1 \\ &\geq \max_{r \in [R^{\rho,k,n}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right)\right) - 1 \\ &= \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right). \end{aligned}$$

We can assume in the following w.l.o.g. $i^* = 1$ and $\mathbb{A}_r(1) = \mathbb{A}_{r1}$ by relabeling the arms (configurations) and query sets (sets of configurations). Now, we first show, that $s_{1|\mathbb{A}_{r1}}(t) - s_{i|\mathbb{A}_{r1}}(t) \geq 0$ for all rounds $r \in [R^{\rho,k,n}]$, all arms $i \in \mathbb{A}_{r1}$ and all $t \geq \tau_i := \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{i|\mathbb{A}_{r1}}}{2}\right)$. Define $\gamma_{\theta|\tilde{\Theta}}$ as the pointwise minimal function such that $|s_{\theta|\tilde{\Theta}}(t) - S_{\theta|\tilde{\Theta}}| \leq \gamma_{\theta|\tilde{\Theta}}(t)$ for all t and $\gamma_{\tilde{\Theta}} = \max_{\theta \in \tilde{\Theta}} \gamma_{\theta|\tilde{\Theta}}$. Thus, $\tau_i \geq \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\frac{\Delta_{i|\mathbb{A}_{r1}}}{2}\right)$ for all $r \in [R^{\rho,k,n}]$, and according to the definition of $\gamma_{\tilde{\Theta}}$ we have

$$|s_{i|\mathbb{A}_{r1}}(t) - S_{i|\mathbb{A}_{r1}}| \leq \gamma_{\mathbb{A}_{r1}}(t) \leq \frac{\Delta_{i|\mathbb{A}_{r1}}}{2} \quad \text{for } t \geq \tau_i.$$

Thus, for all $t \geq \tau_i$,

$$\begin{aligned} s_{1|\mathbb{A}_{r1}}(t) - s_{i|\mathbb{A}_{r1}}(t) &= s_{1|\mathbb{A}_{r1}}(t) - S_{1|\mathbb{A}_{r1}} + S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} + S_{i|\mathbb{A}_{r1}} - s_{i|\mathbb{A}_{r1}}(t) \\ &= s_{1|\mathbb{A}_{r1}}(t) - S_{1|\mathbb{A}_{r1}} - (s_{i|\mathbb{A}_{r1}}(t) - S_{i|\mathbb{A}_{r1}}) + S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} \\ &\geq -2\gamma_{\mathbb{A}_{r1}}(t) + S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} \\ &\geq -2\frac{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}}}{2} + S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}} = 0. \end{aligned}$$

In this scenario, arm i will be eliminated before arm 1, since the $f_\rho(|\mathbb{A}_r|)$ arms with the lowest statistic s are discarded in each round $r \in [R^{\rho,k,n}]$.

Now consider a round $r \in [R^{\rho,k,n}]$ and assume that by reordering the arms, w.l.o.g., $S_{1|\mathbb{A}_{r1}} \geq S_{2|\mathbb{A}_{r1}} \geq \cdots \geq S_{|\mathbb{A}_{r1}||\mathbb{A}_{r1}}$. Since $S_{i|\mathbb{A}_{r1}}$ is non-increasing in i, the τ_i are non-increasing in i: $\tau_2 \geq \tau_3 \geq \ldots$. Thus,

$$t \geq \tau_i \Rightarrow s_{1|\mathbb{A}_{r1}}(t) \geq s_{i|\mathbb{A}_{r1}}(t). \tag{2}$$

Assume now that $1 \in \mathbb{A}_r \wedge 1 \notin \mathbb{A}_{r+1}$

$$\begin{aligned} &\Rightarrow \sum_{i \in \mathbb{A}_{r1}} \mathbb{1}\{s_{i|\mathbb{A}_{r1}}(b_r) > s_{1|\mathbb{A}_{r1}}(b_r)\} > f_\rho(|\mathbb{A}_{r1}|) \\ &\Rightarrow \sum_{i \in \mathbb{A}_{r1}} \mathbb{1}\{b_r < \tau_i\} > f_\rho(|\mathbb{A}_{r1}|) \\ &\Rightarrow b_r < \tau_{f_\rho(|\mathbb{A}_{r1}|)+1}. \end{aligned}$$

This implies

$$1 \in \mathbb{A}_r \wedge b_r \geq \tau_{f_\rho(|\mathbb{A}_{r1}|)+1} \Rightarrow 1 \in \mathbb{A}_{r+1}. \tag{3}$$

Recall that $b_r \geq \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right)$ and $\tau_{f(|\mathbb{A}_{r1}|)+1} = \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{f_\rho(|\mathbb{A}_{r1}|)+1|\mathbb{A}_{r1}}}{2}\right)$.

Case 1: $\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{f_\rho(|\mathbb{A}_{r1}|)+1|\mathbb{A}_{r1}}}{2} \geq \frac{\epsilon}{2}$ and $1 \in \mathbb{A}_r$.
We have $b_r \geq \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{f_\rho(|\mathbb{A}_{r1}|)+1|\mathbb{A}_{r1}}}{2}\right) = \tau_{f_\rho(|\mathbb{A}_{r1}|)+1}$. By (3) we have that $1 \in \mathbb{A}_{r+1}$.

Case 2: $\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{f_\rho(|\mathbb{A}_{r1}|)+1|\mathbb{A}_{r1}}}{2} < \frac{\epsilon}{2}$ and $1 \in \mathbb{A}_r$.
We have $b_r \geq \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\frac{\epsilon}{2}\right)$ and
$\max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\frac{\epsilon}{2}\right) < \max_{r \in [R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}} \left(\max_{r \in [R^{\rho,k,n}]} \frac{\Delta_{f_\rho(|\mathbb{A}_{r1}|)+1|\mathbb{A}_{r1}}}{2}\right) = \tau_{f_\rho(|\mathbb{A}_{r1}|)+1}$.
If $1 \in \mathbb{A}_{r+1}$, Algorithm 1 continues in round $r+1$ with case 1 or case 2. Now assume $1 \notin \mathbb{A}_{r+1}$ and let

$$p := \min\left\{i \in \mathbb{A}_{r1} \ : \ \max_{r \in [R^{\rho,k,n}]} \frac{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}}}{2} \geq \frac{\epsilon}{2}\right\}.$$

By the assumption of the case 2, we have $p > f_\rho(|\mathbb{A}_{r1}|) + 1$ and

$$b_r \geq \max_{r\in[R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}}\left(\frac{\epsilon}{2}\right) \geq \max_{r\in[R^{\rho,k,n}]} \bar{\gamma}^{-1}_{\mathbb{A}_{r1}}\left(\max_{r\in[R^{\rho,k,n}]} \frac{\Delta_{i|\mathbb{A}_{r1}}}{2}\right) = \tau_i \text{ for all } i \geq p.$$

Thus, by (2) we have $s_{1|\mathbb{A}_{r1}}(b_r) \geq s_{i|\mathbb{A}_{r1}}(b_r)$ for $i \geq p$, so all arms $i \geq p$ are eliminated before arm 1. Moreover, we have

$$\max_{i\in\mathbb{A}_{r+1}} S_{i|\mathbb{A}_{r1}} \geq \max_{i<p} S_{i|\mathbb{A}_{r1}} \geq S_{1|\mathbb{A}_{r1}} - \epsilon$$

since

$$\frac{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}}}{2} < \max_{r\in[R^{\rho,k,n}]} \frac{S_{1|\mathbb{A}_{r1}} - S_{i|\mathbb{A}_{r1}}}{2} < \frac{\epsilon}{2}.$$

In addition, by definition of p it holds even for all $r \in [R^{\rho,k,n}]$ that $S_{i|\mathbb{A}_{r1}} > S_{1|\mathbb{A}_{r1}} - \epsilon$. Thus, although CSE might discard i^*, it is ensured that an ε-best arm is still active. Consequently, we relabel the arms such that the still active ε-best arm is now denoted by i^*.

Case 3: $1 \notin \mathbb{A}_r$.
Since $1 \in \mathbb{A}_0$, there was a round $\tilde{r} < r$ such that $1 \in \mathbb{A}_{\tilde{r}}$ and $1 \notin \mathbb{A}_{\tilde{r}+1}$. For this round $\tilde{r}$ only case 2 was possible, otherwise $1 \in \mathbb{A}_{\tilde{r}+1}$. Since case 2 was true and $1 \notin \mathbb{A}_{\tilde{r}+1}$, we have

$$\max_{i\in\mathbb{A}_{\tilde{r}+1}} S_{i|\mathbb{A}_{\tilde{r}}} \geq S_{1|\mathbb{A}_{\tilde{r}}} - \epsilon$$

and also for all other rounds $r \in [R^{\rho,k,n}]$ that $S_{i|\mathbb{A}_{r1}} > S_{1|\mathbb{A}_{r1}} - \epsilon$.

□

C Guarantees for AC-Band

C.1 Number of Epochs

Let $N_{\alpha,\delta} = \lceil \frac{\ln(\delta)}{\ln(1-\alpha)} \rceil$ be the number of sampled configurations necessary to ensure that an ϵ-best configuration is contained in the samples with probability at least $1-\delta$ provided the proportion of ϵ-best configurations in the configuration space is α (see Assumption (A2)). In each epoch e, we consider in total $n_e = \lceil \frac{n_0}{2^e} \rceil + 1$ configurations and keep the winner of the last epoch, such that we have to sample at least $\lceil \frac{n_0}{2^e} \rceil$ new configurations in epoch e. To be precise, we sample in one run of AC-Band, the following number of configurations in total:

$$\begin{aligned}
\sum_{e=1}^{E} \left\lceil \frac{n_0}{2^e} \right\rceil &\geq \sum_{e=1}^{E} \frac{n_0}{2^e} \\
&= n_0 \sum_{e=1}^{E} \left(\frac{1}{2}\right)^e \\
&= n_0 \left(\sum_{e=0}^{E} \left(\frac{1}{2}\right)^e - 1 \right) \\
&= 2n_0 \left(1 - \frac{1}{2^{E+1}}\right) - n_0 \\
&= n_0 - \frac{n_0}{2^E}.
\end{aligned}$$

This value must be greater than $N_{\alpha,\delta}$ to guarantee that we have an ϵ-best configuration contained in the sampled configurations with probability at least $1-\delta$. Rearranging the above inequality leads to

$$\begin{aligned}
2^E &\geq \frac{n_0}{n_0 - N_{\alpha,\delta}} \\
\Rightarrow\ E &\geq \log_2\left(\frac{n_0}{n_0 - N_{\alpha,\delta}}\right),
\end{aligned}$$

where we consider that $n_0 > N_{\alpha,\delta}$. Since we want our algorithm to finish as fast as possible, $E = \left\lceil \log_2\left(\frac{n_0}{n_0 - N_{\alpha,\delta}}\right)\right\rceil$ is a suitable choice. In addition, we need to guarantee that the number of epochs is well defined, thus we must ensure that

$$\begin{aligned} E \geq \log_2\left(\frac{n_0}{n_0 - N_{\alpha,\delta}}\right) &\overset{!}{\geq} 1 \\ \Leftrightarrow \quad \frac{n_0}{n_0 - N_{\alpha,\delta}} &\geq 2 \\ \Leftrightarrow \quad n_0 &\leq 2N_{\alpha,\delta}. \end{aligned}$$

Putting everything together, we get the condition that $n_0 \in \left(N_{\alpha,\delta}, 2N_{\alpha,\delta}\right]$.

C.2 Sufficient Budget

Lemma C.1. *For $N \in \mathbb{N}$ and any $a, b, c \in \mathbb{R}$ it holds that*

$$\sum_{i=1}^{N} \frac{-ia + b + c}{2^i} = \frac{aN - (2^N - 1)(2a - b - c)}{2^N}.$$

Proof.

$$\begin{aligned} \sum_{i=1}^{N} \frac{-ia + b + c}{2^i} &= -a \cdot \sum_{i=1}^{N} \frac{i}{2^i} + (b + c) \cdot \sum_{i=1}^{N} \frac{1}{2^i} \\ &= -a \cdot \sum_{i=0}^{N} \frac{i}{2^i} + (b + c) \cdot \left(\sum_{i=0}^{N} \frac{1}{2^i} - 1\right) \\ &= -a \frac{\frac{N}{2^{N+2}} - \frac{N+1}{2^{N+1}} + \frac{1}{2}}{\left(\frac{1}{2} - 1\right)^2} + (b + c)\left(\frac{1 - \frac{1}{2^{N+1}}}{1 - \frac{1}{2}} - 1\right) \\ &= \frac{-aN}{2^N} + \frac{a(N+1)}{2^{N-1}} - 2a + (b + c)\left(1 - \frac{1}{2^N}\right) \\ &= \frac{-aN + a(2N + 2) - a2^{N+1}}{2^N} + \frac{(b + c)(2^N - 1)}{2^N} \\ &= \frac{aN - (2^N - 1)(2a - b - c)}{2^N}, \end{aligned}$$

where the closed-form sum formulas of the geometric series are used in the third equality. □

Proof of Thm. 0.1. If the total budget B for AC-Band is such that the epoch-wise budget for CSE is at least $z(\rho_e, k, n_e, \epsilon)$ for each epoch e (see Theorem B.2), AC-Band will return a configuration that is locally ϵ-best. With the help of Assumption (A3), we can then infer the claim.

Thus, the sufficient budget to guarantee that AC-Band finds a local ϵ-best configuration i^* can be computed as

$$\begin{aligned} &\sum_{e=1}^{E} z(\rho_e, k, n_e, \epsilon) \\ &= \sum_{e=1}^{E} R^{\rho_e,k,n_e} \max_{r \in [R^{\rho_e,k,n_e}]} P_r^{\rho_e,k,n_e} \cdot \max_{r \in [R^{\rho_e,k,n_e}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho_e,k,n_e}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right)\right) \\ &= \sum_{e=1}^{E} (R_1^{\rho_e,k,n_e} + R_2^{\rho_e,k}) \left\lfloor \frac{n_e}{k} \right\rfloor \cdot \max_{r \in [R^{\rho_e,k,n_e}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)}\left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho_e,k,n_e}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right)\right) \\ &=: (*) \end{aligned}$$

Note that for $\rho_e = \log_2\left(\frac{e+k-1}{e}\right)$, we have $f_{\rho_e}(x) = \left\lfloor \frac{x}{2^\rho} \right\rfloor = \left\lfloor \frac{xe}{e+k-1} \right\rfloor$. We can estimate $R_1^{\rho_e,k,n_e}$ now by

$$R_1^{\rho_e,k,n_2} \leq \left\lceil \log_{\frac{k}{f_{\rho_e}(k)}}(n_e) \right\rceil$$

$$
\begin{aligned}
&= \left\lceil \log_{\frac{k}{\lfloor \frac{ke}{e+k-1} \rfloor}} \left(\left\lceil \frac{n_0}{2^e} \right\rceil + 1 \right) \right\rceil \\
&\leq \left\lceil \log_{\frac{e+k-1}{e}} \left(\left\lceil \frac{n_0}{2^e} \right\rceil + 1 \right) \right\rceil \\
&= \left\lceil \frac{\log\left(\left\lceil \frac{n_0}{2^e} \right\rceil + 1 \right)}{\log\left(1 + \frac{k-1}{e}\right)} \right\rceil \\
&\leq \left\lceil \frac{\log\left(\frac{n_0}{2^e} + 2 \right)}{\log\left(1 + \frac{k-1}{E}\right)} \right\rceil \\
&= \left\lceil \frac{\log\left(n_0 + 2^{e+1} \right) - \log(2^e)}{\log\left(1 + \frac{k-1}{E}\right)} \right\rceil \\
&\leq \left\lceil \frac{\log\left(n_0 + 2^{E+1} \right) - e\log(2)}{\log\left(1 + \frac{k-1}{E}\right)} \right\rceil \\
&\leq \left\lceil \frac{\log\left(n_0 + 2^{\log_2\left(\frac{n_0}{n_0 - N_{\alpha,\delta}}\right)+2} \right) - e\log(2)}{\log\left(1 + \frac{k-1}{E}\right)} \right\rceil \\
&= \left\lceil \frac{\log\left(n_0 + 4\frac{n_0}{n_0 - N_{\alpha,\delta}} \right)}{\log\left(1 + \frac{k-1}{E}\right)} - e \cdot \frac{\log(2)}{\log\left(1 + \frac{k-1}{E}\right)} \right\rceil \\
&\leq \underbrace{1 + \log_{1+\frac{k-1}{E}} \left(n_0 + 4\frac{n_0}{n_0 - N_{\alpha,\delta}} \right)}_{=:C_2} - e \cdot \underbrace{\log_{1+\frac{k-1}{E}}(2)}_{=:C_1}.
\end{aligned}
$$

We can proceed analogously to get an upper bound for $R_2^{\rho_e,k}$:

$$
\begin{aligned}
R_2^{\rho_e,k} \leq \left\lceil \log_{\frac{k}{f_{\rho_e}(k)}}(k) \right\rceil &= \left\lceil \log_{\frac{k}{\lfloor \frac{ke}{e+k-1} \rfloor}}(k) \right\rceil \\
&\leq \left\lceil \log_{\frac{e+k-1}{e}}(k) \right\rceil = \left\lceil \log_{1+\frac{k-1}{e}}(k) \right\rceil \\
&\leq \left\lceil \log_{1+\frac{k-1}{E}}(k) \right\rceil =: C_3 \leq C_2.
\end{aligned}
$$

In the next step we can put the above estimations together.

$$
\begin{aligned}
(*) &\leq \sum_{e=1}^{E} \left\lfloor \left(\frac{n_0}{2^e} + 2 \right) \frac{1}{k} \right\rfloor \cdot (-eC_1 + C_2 + C_3) \\
&\quad \cdot \max_{r \in [R^{\rho_e,k,n_e}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{ \frac{\epsilon}{2}, \max_{r \in [R^{\rho_e,k,n_e}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2} \right\} \right) \right) \\
&\leq \underbrace{\max_{e \in [E]} \max_{r \in [R^{\rho_e,k,n_e}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{ \frac{\epsilon}{2}, \max_{r \in [R^{\rho_e,k,n_e}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2} \right\} \right) \right)}_{=:\bar{\gamma}^{-1}} \\
&\quad \cdot \left(\frac{n_0}{k} \sum_{e=1}^{E} \frac{-eC_1 + C_2 + C_3}{2^e} - \frac{2C_1}{k} \sum_{e=1}^{E} e + \frac{2E(C_2 + C_3)}{k} \right) \\
&= \bar{\gamma}^{-1} \cdot \left(\frac{n_0}{k} \cdot \frac{C_1 E - (2^E - 1)(2C_1 - C_2 - C_3)}{2^E} + \frac{2E(C_2 + C_3) - C_1 E(E+1)}{k} \right)
\end{aligned}
$$

according to Lemma C.1 and the Gaussian sum formula.
A cruder bound for the sufficient budget is given by

$$
\bar{\gamma}^{-1} \cdot \frac{n_0 + 2^{E+1}}{k} \cdot \frac{C_1 E - (2^E - 1)(2C_1 - C_2 - C_3)}{2^E}. \tag{4}
$$

Recall that the number of parallel runs is decreasing with the epoch e, so that in the worst case the configuration returned by AC-Band is sampled only in the last epoch E. Consequently it will be run in parallel only with configurations, which are considered in the last epoch E, and guaranteed to be a local ϵ-best configuration (see Theorem B.2). By Assumption (A3), the local ϵ-best property corresponds to a global ϵ-best property with probability of at least

$$1 - \frac{1}{\#\text{query sets containing } i^* \text{ in epoch } E} = 1 - \left(R^{\rho_E,k,n_E}\right)^{-1}$$

for this worst case. In addition we have to take into account that an ϵ-best configuration is contained only with probability at least $1-\delta$, such that we get an overall probability of at least

$$\min\{1-\delta, 1 - \left(R^{\rho_E,k,n_E}\right)^{-1}\}$$

that AC-Band returns an ϵ-best configuration. □

Note that the total budget of AC band is divided among the epoch-wise calls of CSE by means of the quotient c_e:

$$c_e = \frac{(C_1 E - (2^E - 1)(2C_1 - C_2 - C_3))2^e}{2^E(-eC_1 + C_2 + C_3)}.$$

This quotient is obtained by bounding the sufficient budget for CSE similar as in the proof of Theorem 0.1 to obtain (4):

$$\begin{aligned} z(\rho_e, k, n_e, \epsilon) &= R^{\rho_e,k,n_e} \max_{r \in [R^{\rho_e,k,n_e}]} P_r^{\rho_e,k,n_e} \\ &\quad \cdot \max_{r \in [R^{\rho_e,k,n_e}]} \left(1 + \bar{\gamma}^{-1}_{\mathbb{A}_r(i^*)} \left(\max\left\{\frac{\epsilon}{2}, \max_{r \in [R^{\rho_e,k,n_e}]} \frac{\Delta_{(f_\rho(|\mathbb{A}_r(i^*)|)+1)|\mathbb{A}_r(i^*)}}{2}\right\}\right)\right) \\ &\leq \bar{\gamma}^{-1}(R_1^{\rho_e,k,n_e} + R_2^{\rho_e,k}) \left\lfloor \frac{n_e}{k} \right\rfloor \\ &\leq \bar{\gamma}^{-1} \cdot \frac{n_0 + 2^{E+1}}{k} \cdot \frac{-eC_1 + C_2 + C_3}{2^e}. \end{aligned}$$

AC-Band thus allocates its entire budget such that any call to CSE is guaranteed to return an appropriate configuration once the total budget is sufficiently large.

C.3 Discussion of Sufficient Budget

The behavior of the sufficient budget with respect to k, α and δ is illustrated in Figure 3. Note that we ignore the γ^{-1} terms in these plots, as these depend on the underlying AC problem and occur as a multiplicative constant in the sufficient budget.

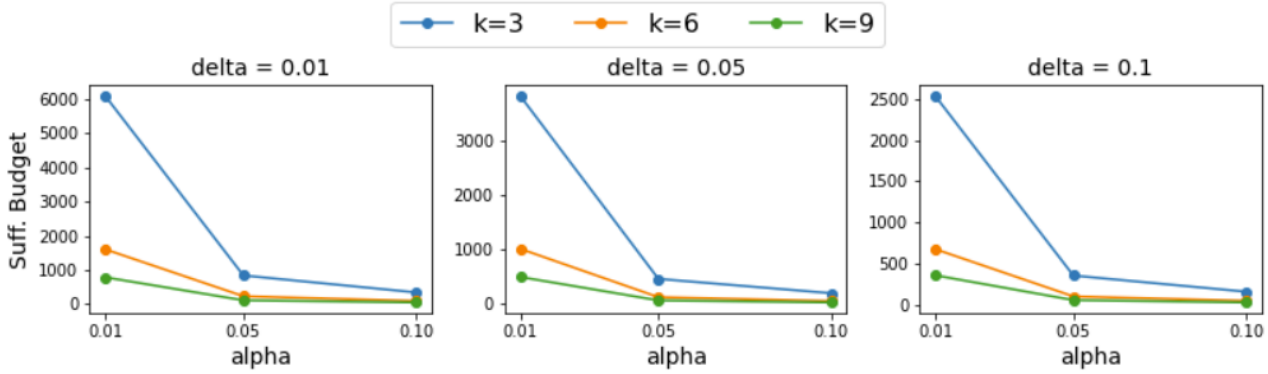


Figure 3: Sufficient budget for different values for k, α and δ.

The dependency of the sufficient budget on k, α and δ is as expected, since it decreases with increasing k, α and δ, respectively.

D Extended Experiments

We provide further details regarding the experiments in Section . In particular, the results from Figure 2 are reported in more detail in Table 1 and augmented by the results for the Regions200 dataset, for which we provide a similar illustration of the results in Figure 4 Moreover, we outline additional experimental results and provide additional metrics to evaluate the quality of the configurations found. The experimental setup used here is the same as in the main paper. We report two additional metrics to provide additional insights: the percent gap to subset-best and the R^δ metric used in previous works (Kleinberg et al. 2019; Weisz,

György, and Szepesvári 2018, 2019; Weisz et al. 2020) within the following tables. The percent gap to subset-best is a variation of the percent gap to best metric where only the configurations that were sampled by the method during a run are considered. In this way, we can see how good a method performs within the sample it selects. A value of 0 means the configuration returned is the best in the subset. A 10% cutoff is used for the δ-capped runtime. We provide results for two additional experiments: (i) varying the δ for AC-Band and (I)CAR(++) and, (ii) increasing the configuration sampling budget of AC-band.

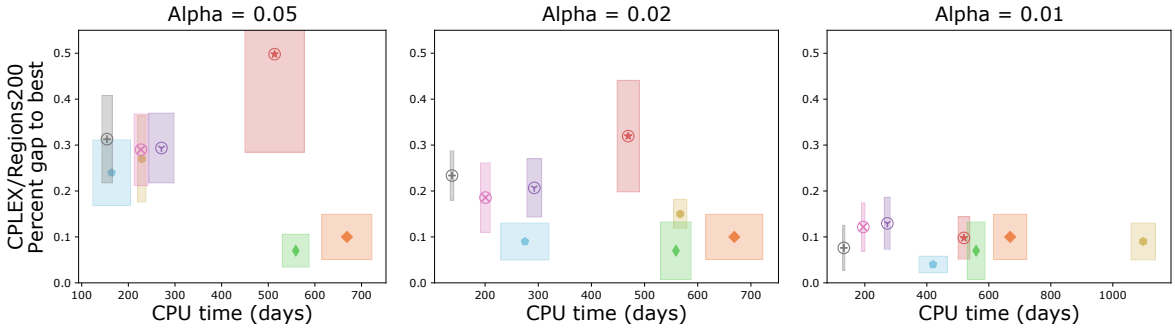


Figure 4: Mean CPU time and percent gap to best over 5 seeds for $\delta = 0.05$ and different α (columns) for AC-Band, ICAR, CAR++ and Hyperband on the Regions200 dataset. Circles indicate variants of AC-Band. Rectangles represent the standard error over the seeds. The number of configurations tried for CAR++: $\{97, 245, 492\}$, ICAR: $\{134, 351, 724\}$, AC-Band: $\{60, 153, 303\}$, Hyperband($\eta = 5$): $\{842\}$, Hyperband($\eta = 8$): $\{618\}$.

Varying δ The user must decide δ based on their preferences, thus there is no "correct" value to set δ to in our experiments. Therefore, we also experiment with $\delta = 0.01$ in addition to the results in Section where $\delta = 0.05$ is used. The results for a lower δ (see Figure 5 or Table 2) are consistent with the results reported for $\delta = 0.05$. In particular, AC-Band with $k = 2$ lies on the Pareto front of percent gap to best and CPU time, backing up the claim that a lower value of k is preferable. With $\delta = 0.01$ and $k = 2$, AC-Band is 80% percent faster than ICAR and 74% faster than Hyperband over all α and all datasets, while providing configurations that are only 7% and 6% worse in terms of the gap to the best configuration. Furthermore, AC-Band exhibits a nearly linear speedup with the number of available cores, leading to a low wallclock time for increasing k. We note that a real parallel implementation would, of course, suffer from some extra overhead.

Looking closer at the results obtained for the CNFuzzDD dataset for $\alpha = 0.01$, we note that the percent gap to best increases for $k = 8$. This increase is solely due to one seed for which a percent gap to best value of 5.74 is obtained. AC-Band uses most of its sampling budget in the first rounds, where large sets of configurations are evaluated on large sets of instances. Over the course of AC-Band, both the configuration and instance sets become smaller. For the seed in question, AC-Band samples one new configuration in the last round that is able to beat the current incumbent on 18 instances. Since the configuration wins, it is returned, even though the incumbent has seen more instances and proven worthy over the previous 8 epochs. The possibility of this happening grows with the number of configurations to sample (decreasing α) since the same amount of instances is distributed among more configurations.

Increasing the sampling budget of AC-Band AC-Band samples fewer configurations than Hyperband or (I)CAR(++) and leaves more of the configuration space unexplored. This explains, in part, why AC-Band usually has a worse percent gap to best than its competitors, but less runtime. To investigate this further, we let AC-Band sample the same number of configurations as CAR++ and Hyperband with $\eta = 8$ in Table 3 and Table 4. In particular, we set N to be equal to the number of samples of either method and set $n_0 = N + 1$. Note that we do not sample exactly the same amount of configurations due to the rounding operations within AC-Band.

On the CNFuzzDD and RCW dataset (see Table 3), a larger sampling budget for AC-Band leads to a gap to best that is nearly as good as those obtained by (I)CAR(++), while needing significantly less CPU time. The same can be seen for the Regions200dataset, however with some exceptions where the gap to best increases. Specifically, for $\alpha = 0.05$ where only 101 configurations out of 2000 are examined. For a few seeds, no good configuration is contained in the 101 samples, leading to these results.

Table 4 report the results obtained for Hyperband with different values of η as well as the AC-Band results with a sampling budget of 618 configurations. For all three datasets, AC-Band comes closer to the results of Hyperband in terms of gap to best, while needing significantly less CPU time. This is especially true for the Regions200 and RCW datasets. On the CNFuzzDD dataset, AC-Band's performance is weaker, which may be due to this dataset containing the smallest number of instances.

	CPU Time (thousand days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	100.64	12.74	242.74	14.96	467.32	25.08	0.02	0.04	0.01	0.01	0.02	0.03	0.01	0.02	0.01	0.01	0.02	0.03	5.0	0.1	4.9	0.1	4.9	0.1
CAR++	92.30	5.48	224.21	16.38	452.06	18.08	0.05	0.04	0.01	0.01	0.01	0.01	0.02	0.03	0.00	0.01	0.01	0.01	5.2	0.2	4.9	0.1	4.9	0.1
CAR	157.76	18.07	367.86	7.24	771.45	21.72	0.04	0.03	0.01	0.01	0.01	0.01	0.01	0.02	0.00	0.01	0.01	0.01	5.2	0.2	4.9	0.1	4.9	0.1
$k = 2$	13.05	2.04	11.76	0.90	11.02	0.56	0.14	0.17	0.16	0.19	0.05	0.08	0.09	0.12	0.14	0.19	0.05	0.08	5.7	0.90	5.7	1.0	5.1	0.5
$k = 4$	16.32	1.91	14.91	0.68	14.82	0.53	0.05	0.06	0.02	0.03	0.01	0.02	0.00	0.00	0.00	0.01	0.01	0.02	5.3	0.3	5.1	0.2	4.9	0.1
$k = 8$	20.09	1.92	21.58	1.24	21.21	0.91	0.12	0.14	0.01	0.01	0.01	0.01	0.06	0.11	0.00	0.01	0.01	0.01	5.6	0.8	5.0	0.1	4.9	0.1
$k = 16$	37.87	3.46	36.07	1.71	38.45	0.32	0.05	0.06	0.05	0.07	0.05	0.10	0.02	0.03	0.03	0.05	0.05	0.10	5.2	0.3	5.0	0.3	5.2	0.6

(a): CNFuzzDD dataset

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	164.30	91.05	274.84	100.72	420.15	103.24	0.24	0.16	0.09	0.09	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.00	34.8	4.3	29.8	2.2	28.5	1.8
CAR++	229.29	19.93	566.99	28.21	1097.90	88.41	0.27	0.17	0.15	0.07	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	35.3	4.3	32.0	2.2	29.8	1.8
CAR	523.69	53.34	1294.87	64.11	2549.22	199.00	0.27	0.17	0.16	0.09	0.09	0.09	0.00	0.00	0.10	0.30	0.00	0.00	35.3	4.5	31.9	1.6	29.8	2.2
$k = 2$	154.27	25.29	137.79	7.59	132.00	7.78	0.31	0.21	0.23	0.12	0.08	0.11	0.00	0.00	0.01	0.02	0.00	0.00	36.6	5.9	34.6	5.0	30.2	1.6
$k = 4$	227.07	33.18	200.72	20.76	194.88	12.07	0.29	0.17	0.19	0.17	0.12	0.12	0.00	0.00	0.00	0.00	0.01	0.02	36.2	4.4	34.3	5.4	31.8	2.7
$k = 8$	271.02	60.26	292.56	31.26	272.56	21.64	0.29	0.17	0.21	0.14	0.13	0.13	0.00	0.00	0.01	0.02	0.01	0.02	36.2	6.3	33.5	3.5	31.1	2.2
$k = 16$	513.57	142.68	469.19	46.49	519.70	42.30	0.50	0.48	0.32	0.27	0.10	0.10	0.00	0.00	0.03	0.06	0.00	0.00	40.0	9.0	36.8	6.9	30.6	1.7

(b): Regions200 dataset

	CPU Time (thousand days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	1.28	0.39	2.03	0.30	4.07	0.24	0.14	0.08	0.08	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	156.1	11.9	146.5	4.1	143.3	4.9
CAR++	1.73	0.37	3.64	0.18	7.52	0.13	0.17	0.09	0.10	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	162.1	11.9	149.1	4.1	143.3	4.9
CAR	3.30	0.50	7.59	0.19	15.65	0.25	0.17	0.09	0.10	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	160.1	13.3	149.1	4.7	143.3	4.9
$k = 2$	0.30	0.02	0.27	0.01	0.26	0.01	0.18	0.13	0.12	0.12	0.10	0.04	0.02	0.03	0.04	0.07	0.04	0.04	164.1	32.2	152.2	17.8	144.7	2.9
$k = 4$	0.52	0.04	0.47	0.03	0.45	0.02	0.21	0.08	0.19	0.22	0.07	0.08	0.03	0.03	0.08	0.16	0.02	0.03	169.8	18.5	165.2	36.5	139.8	8.4
$k = 8$	0.69	0.09	0.72	0.03	0.71	0.04	0.20	0.11	0.23	0.22	0.09	0.09	0.04	0.06	0.14	0.15	0.06	0.07	165.6	24.0	173.0	42.5	143.0	14.0
$k = 16$	1.42	0.29	1.31	0.06	1.43	0.08	0.21	0.08	0.15	0.16	0.12	0.06	0.05	0.07	0.03	0.06	0.05	0.04	167.6	15.5	158.2	30.5	146.4	4.6

(c): RCW dataset

Table 1: Mean (μ) and standard derivation (σ) for CPU time, percent gap to best, percent gap to subset-best and mean δ-capped runtime over 5 seeds for $\delta = \mathbf{0.05}$ and different α (columns) for AC-Band, ICAR, CAR++, CAR on the CNFuzzDD (top), Regions200 (middle) and RCW (bottom) dataset. The number of configurations tried for CAR(++): $\{97, 245, 492\}$, ICAR: $\{134, 351, 724\}$, AC-Band: $\{60, 153, 303\}$.

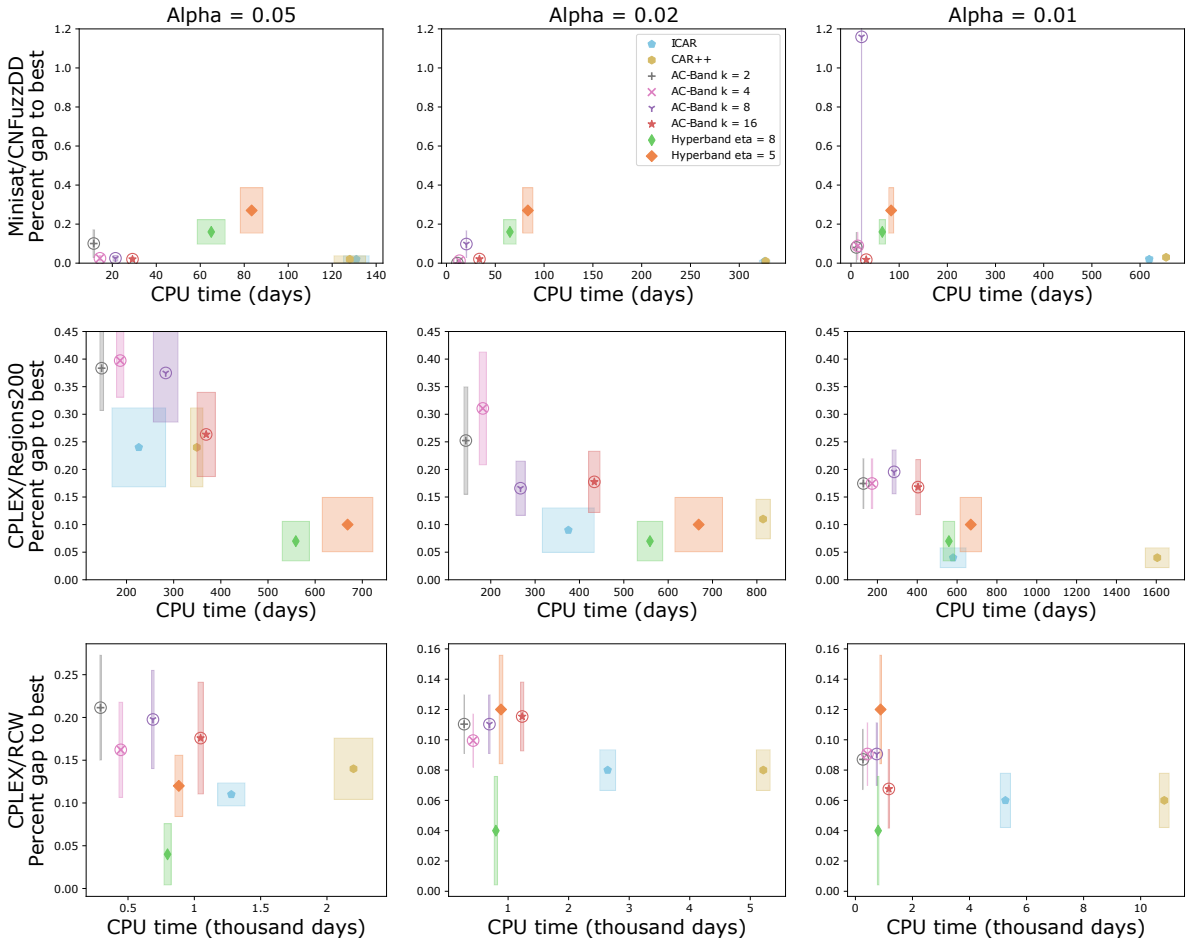


Figure 5: Mean CPU time and percent gap to best over 5 seeds for $\delta = 0.01$ and different α (columns) for AC-Band, ICAR, CAR++ and Hyperband on CNFuzzDD (top), Regions200 (middle) and RCW (bottom). Circles indicate variants of AC-Band. Rectangles represent the standard error over the seeds. The number of configurations tried for CAR++: $\{128, 325, 652\}$, ICAR: $\{166, 431, 884\}$, AC-Band: $\{93, 232, 462\}$, Hyperband($\eta = 5$): $\{842\}$, Hyperband($\eta = 8$): $\{618\}$.

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	130.57	13.14	325.61	11.80	619.16	15.58	0.02	0.04	0.01	0.01	0.02	0.03	0.01	0.02	0.01	0.01	0.02	0.03	5.0	0.2	4.9	0.1	4.9	0.1
CAR++	128.14	15.59	327.18	9.35	654.46	12.68	0.02	0.04	0.01	0.01	0.03	0.02	0.01	0.02	0.01	0.01	0.03	0.02	5.0	0.2	4.9	0.1	4.9	0.1
CAR	220.33	29.17	554.95	19.47	1169.06	12.44	0.01	0.02	0.01	0.01	0.02	0.02	0.00	0.00	0.01	0.01	0.01	0.02	5.1	0.2	4.9	0.1	4.9	0.1
$k = 2$	11.56	0.77	11.17	0.32	11.20	0.65	0.10	0.14	0.00	0.00	0.08	0.15	0.08	0.15	0.00	0.00	0.08	0.14	5.5	0.8	4.9	0.1	5.4	0.8
$k = 4$	14.38	0.63	13.29	0.50	14.11	0.36	0.02	0.03	0.01	0.02	0.09	0.14	0.00	0.00	0.01	0.01	0.09	0.14	5.1	0.1	4.9	0.0	5.3	0.8
$k = 8$	21.43	2.45	20.38	0.63	21.98	0.52	0.02	0.03	0.10	0.14	2.16	2.29	0.00	0.00	0.09	0.12	1.16	1.29	5.1	0.1	5.4	0.7	13.0	16.1
$k = 16$	29.18	1.99	33.77	0.82	31.70	0.45	0.02	0.02	0.02	0.03	0.02	0.04	0.00	0.00	0.01	0.02	0.02	0.04	5.0	0.1	5.0	0.2	5.0	0.3

(a): CNFuzzDD dataset

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	226.12	127.05	374.51	131.49	579.64	145.09	0.24	0.16	0.09	0.09	0.04	0.04	0.01	0.03	0.00	0.00	0.00	0.00	34.8	4.3	29.8	2.2	28.5	1.8
CAR++	349.44	29.05	813.73	35.96	1604.52	133.59	0.24	0.16	0.11	0.08	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.00	34.8	4.3	30.6	2.2	28.5	1.8
CAR	798.6	74.21	1885.13	83.41	3717.16	264.71	0.24	0.16	0.11	0.08	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.00	34.8	4.3	30.6	1.6	28.5	1.8
$k = 2$	147.24	8.72	143.81	9.80	129.57	5.53	0.38	0.17	0.25	0.22	0.17	0.10	0.00	0.00	0.02	0.03	0.05	0.07	38.9	4.9	34.4	5.2	31.6	3.5
$k = 4$	186.61	17.15	182.01	17.80	172.98	7.71	0.40	0.15	0.31	0.23	0.17	0.10	0.00	0.00	0.03	0.03	0.01	0.02	39.6	3.5	36.0	5.6	31.6	3.5
$k = 8$	283.07	58.47	267.06	23.24	283.98	22.41	0.37	0.20	0.17	0.11	0.20	0.09	0.02	0.04	0.02	0.02	0.03	0.02	38.1	4.9	31.5	3.3	32.0	3.0
$k = 16$	368.87	43.25	433.53	28.73	403.93	27.27	0.26	0.17	0.18	0.12	0.17	0.11	0.00	0.00	0.00	0.00	0.02	0.02	35.2	5.4	33.0	2.8	31.2	3.0

(b): Regions200 dataset

	CPU Time (thousand days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	1.27	0.22	2.64	0.29	5.26	0.40	0.11	0.03	0.08	0.03	0.06	0.04	0.00	0.00	0.00	0.00	0.00	0.00	150.1	2.3	146.2	3.7	142.1	5.9
CAR++	2.20	0.32	5.21	0.25	10.82	0.38	0.14	0.08	0.08	0.03	0.06	0.02	0.00	0.00	0.00	0.00	0.00	0.00	156.1	2.3	146.5	3.7	143.3	5.9
CAR	4.44	0.48	10.79	0.26	22.60	0.61	0.14	0.08	0.08	0.03	0.06	0.02	0.00	0.00	0.00	0.00	0.00	0.00	156.1	11.9	146.5	4.1	143.3	4.9
$k = 2$	0.29	0.01	0.27	0.09	0.26	0.01	0.21	0.14	0.11	0.04	0.09	0.04	0.07	0.08	0.04	0.04	0.03	0.03	168.7	25.9	145.1	3.6	141.9	4.7
$k = 4$	0.45	0.03	0.42	0.01	0.42	0.02	0.16	0.12	0.10	0.04	0.09	0.05	0.05	0.09	0.04	0.04	0.04	0.04	156.9	19.6	144.9	3.3	141.8	4.6
$k = 8$	0.69	0.02	0.70	0.03	0.74	0.05	0.20	0.13	0.11	0.04	0.09	0.05	0.05	0.09	0.04	0.04	0.04	0.04	164.2	21.7	145.1	3.6	141.8	4.6
$k = 16$	1.05	0.04	1.23	0.05	1.17	0.05	0.18	0.15	0.12	0.05	0.07	0.06	0.06	0.09	0.04	0.04	0.02	0.02	163.7	27.2	145.6	3.8	139.4	5.0

(c): RCW dataset

Table 2: Mean (μ) and standard derivation (σ) for CPU time, percent gap to best, percent gap to subset-best and mean δ-capped runtime over 5 seeds for $\delta =$ **0.01** and different values of α (columns) for AC-Band, ICAR, CAR++, CAR on the CNFuzzDD (top), Regions200 (middle) and RCW (bottom) datasets. The number of configurations tried for CAR(++): $\{128, 325, 652\}$, ICAR: $\{166, 431, 884\}$, AC-Band: $\{93, 232, 462\}$.

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	100.64	12.74	242.74	14.96	467.32	25.08	0.02	0.04	0.01	0.01	0.02	0.03	0.01	0.02	0.01	0.01	0.02	0.03	5.0	0.1	4.9	0.1	4.9	0.1
CAR++	92.30	5.48	224.21	16.38	452.06	18.08	0.05	0.04	0.01	0.01	0.01	0.01	0.02	0.03	0.00	0.01	0.01	0.01	5.2	0.2	4.9	0.1	4.9	0.1
CAR	157.76	18.07	367.86	7.24	771.45	21.72	0.04	0.03	0.01	0.01	0.01	0.01	0.01	0.02	0.00	0.01	0.01	0.01	5.2	0.2	4.9	0.1	4.9	0.1
$k = 2$	11.66	0.87	11.50	0.25	10.83	0.25	0.03	0.02	0.02	0.02	0.09	0.13	0.00	0.00	0.01	0.02	0.09	0.14	5.1	0.1	5.0	0.2	5.3	0.7
$k = 4$	14.18	0.49	14.53	0.42	13.51	0.35	0.03	0.03	0.10	0.16	0.03	0.04	0.00	0.00	0.09	0.14	0.03	0.05	5.1	0.1	5.4	0.8	5.0	0.2
$k = 8$	22.39	0.82	21.45	0.59	20.17	0.58	0.02	0.03	0.02	0.01	0.00	0.01	0.00	0.00	0.01	0.01	0.00	0.01	5.1	0.2	5.0	0.1	4.9	0.1
$k = 16$	29.56	1.61	30.00	1.18	32.28	0.43	0.04	0.03	0.01	0.01	0.02	0.03	0.00	0.00	0.00	0.00	0.02	0.03	5.2	0.2	5.0	0.1	5.0	0.1

(a): CNFuzzDD dataset

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	164.30	91.05	274.84	100.72	420.15	103.24	0.24	0.16	0.09	0.09	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.00	34.8	4.3	29.8	2.2	28.5	1.8
CAR++	229.29	19.93	566.99	28.21	1097.90	88.41	0.27	0.17	0.15	0.07	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	35.3	4.3	32.0	2.2	29.8	1.8
CAR	523.69	53.34	1294.87	64.11	2549.22	199.00	0.27	0.17	0.16	0.09	0.09	0.09	0.00	0.00	0.10	0.30	0.00	0.00	35.3	4.5	31.9	1.6	29.8	2.2
$k = 2$	148.29	10.60	136.75	12.34	125.29	3.91	0.35	0.20	0.15	0.13	0.04	0.04	0.01	0.03	0.01	0.02	0.00	0.01	37.8	5.6	31.8	3.0	28.9	1.8
$k = 4$	184.87	19.23	187.03	15.01	166.71	6.53	0.58	0.51	0.16	0.18	0.09	0.10	0.14	0.28	0.00	0.00	0.02	0.05	42.4	9.7	32.3	4.5	30.0	1.5
$k = 8$	310.18	25.52	263.16	19.92	241.98	13.16	0.39	0.27	0.12	0.09	0.14	0.14	0.04	0.08	0.01	0.01	0.05	0.10	38.6	6.5	30.7	1.6	31.1	2.4
$k = 16$	380.15	43.31	358.15	40.36	384.76	22.49	0.36	0.19	0.18	0.18	0.09	0.10	0.01	0.03	0.02	0.03	0.01	0.02	38.1	5.2	32.5	4.4	30.0	1.5

(b): Regions200 dataset

	CPU Time (days)						Percent gap to best						Percent gap to subset-best						R^δ					
α	0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01		0.05		0.02		0.01	
Method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ICAR	1.28	0.40	2.03	0.30	4.07	0.24	0.14	0.08	0.08	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	156.1	11.9	146.5	4.1	143.3	4.9
CAR++	1.72	0.37	3.64	0.18	7.52	0.13.	0.17	0.09	0.10	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	162.1	11.9	149.1	4.1	143.3	4.9
CAR	3.30	0.50	7.59.	0.19	15.65	0.26	0.17	0.09	0.10	0.03	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00	160.1	13.3	149.1	4.7	143.3	4.9
$k = 2$	0.30	0.02	0.27	0.01	0.27	0.01	0.14	0.09	0.11	0.04	0.12	0.06	0.01	0.03	0.04	0.04	0.07	0.05	154.5	16.1	145.1	3.6	146.4	4.6
$k = 4$	0.43	0.02	0.04	0.01	0.04	0.01	0.18	0.11	0.11	0.04	0.10	0.10	0.04	0.03	0.03	0.04	0.07	0.06	161.3	24.1	145.1	3.6	145.9	12.3
$k = 8$	0.81	0.06	0.73	0.03	0.69	0.02	0.18	0.11	0.11	0.05	0.13	0.06	0.04	0.05	0.03	0.04	0.09	0.05	159.8	18.9	146.6	6.7	148.5	6.3
$k = 16$	1.09	0.10	1.06	0.04	1.17	0.04	0.23	0.20	0.09	0.05	0.15	0.06	0.08	0.11	0.03	0.04	0.10	0.06	169.8	39.2	141.8	4.6	151.8	6.3

(c): RCW dataset

Table 3: Mean (μ) and standard derivation (σ) for CPU time, percent gap to best, percent gap to subset-best and mean δ-capped runtime over 5 seeds for $\delta =$ **0.05** and different values of α (columns) for AC-Band, ICAR, CAR++, CAR on the CNFuzzDD (top), Regions200 (middle) and RCW (bottom) datasets. For AC-Band N was set to the number of configurations sampled by CAR++ for the respective α value. The number of configurations tried for CAR(++): $\{97, 245, 492\}$, ICAR: $\{134, 351, 724\}$, AC-Band: $\{$**101, 247, 492**$\}$. Note that AC-Band samples slightly more configurations due to rounding operations.

		CPU Time (days)		Percent gap to best		Percent gap to subset-best		R^δ	
	Method	μ	σ	μ	σ	μ	σ	μ	σ
Hyperband	$\eta = 4$	80.05	18.36	0.14	0.17	0.13	0.16	5.7	0.0
	$\eta = 5$	83.37	11.50	0.27	0.26	0.27	0.26	6.3	0.1
	$\eta = 6$	77.64	23.71	0.13	0.14	0.11	0.13	5.8	0.8
	$\eta = 8$	65.43	14.21	0.16	0.14	0.16	0.14	5.7	0.8
AC-Band	$k = 2$	10.55	0.32	0.39	0.53	0.39	0.53	7.1	3.0
	$k = 4$	13.66	0.26	0.06	0.08	0.06	0.08	5.2	0.5
	$k = 8$	19.49	0.26	0.21	0.36	0.21	0.36	6.1	2.1
	$k = 16$	33.35	0.57	0.01	0.02	0.01	0.02	5.0	0.1

(a): CNFuzzDD dataset

		CPU Time (days)		Percent gap to best		Percent gap to subset-best		R^δ	
	Method	μ	σ	μ	σ	μ	σ	μ	σ
Hyperband	$\eta = 4$	798.58	64.32	0.06	0.06	0.06	0.05	29.0	2.3
	$\eta = 5$	668.78	120.01	0.10	0.11	0.04	0.05	30.7	3.1
	$\eta = 6$	662.96	94.83	0.06	0.07	0.06	0.06	28.6	2.4
	$\eta = 8$	559.19	64.26	0.07	0.08	0.00	0.00	29.9	2.5
AC-Band	$k = 2$	117.68	6.73	0.09	0.09	0.05	0.07	29.6	1.92
	$k = 4$	165.07	11.85	0.09	0.09	0.05	0.07	29.6	1.92
	$k = 8$	230.52	25.96	0.09	0.09	0.02	0.02	29.6	1.92
	$k = 16$	385.45	32.5	0.09	0.09	0.04	0.02	29.2	2.03

(b): Regions200 dataset

		CPU Time (days)		Percent gap to best		Percent gap to subset-best		R^δ	
	Method	μ	σ	μ	σ	μ	σ	μ	σ
Hyperband	$\eta = 4$	991.58	55.48	0.07	0.08	0.07	0.07	141.4	9.1
	$\eta = 5$	883.04	67.03	0.12	0.08	0.07	0.07	147.6	12.2
	$\eta = 6$	873.19	56.11	0.10	0.11	0.10	0.10	144.3	22.1
	$\eta = 8$	796.60	59.49	0.04	0.08	0.00	0.00	138.0	9.3
AC-Band	$k = 2$	250.98	6.87	0.14	0.10	0.11	0.05	151.1	14.0
	$k = 4$	411.85	11.33	0.12	0.12	0.09	0.07	148.5	15.1
	$k = 8$	657.61	25.65	0.11	0.12	0.07	0.07	146.6	17.4
	$k = 16$	1196.41	38.75	0.08	0.07	0.07	0.07	139.3	16.0

(c): RCW dataset

Table 4: Mean (μ) and standard derivation (σ) for CPU time, percent gap to best, percent gap to subset-best and mean δ-capped runtime over 5 seeds for different η for Hyperband and AC-Band on the CNFuzzDD (top), Regions200 (middle) and RCW (bottom) datasets. The number of configurations tried for Hyperband: $\{378, 842, 280, 618\}$, AC-Band: $\{618\}$. Note that we use a value of η for which no more than the available number of configurations in a dataset would be sampled.

C

Appendix to Best Arm Identification with Retroactively Increased Sampling Budget for More Resource-Efficient HPO

Organization of Appendix

A Algorithms

Variants of iSHA. While iSHA is arguably the most efficient way to continue a previous run of SHA synchronously, there are also other possible ways to do so. One way, which we call discarding Incremental-SuccessiveHalving (given in Algorithm 2), is when the start pool of hyperparameter configurations is extended by the new hyperparameter configurations, it is allowed to discard hyperparameter configurations that were promoted in the previous run and have already been evaluated on a larger budget. Another way that is more efficient and reusing previous evaluations of hyperparameter configurations, is by conserving the information about hyperparameter configurations that have already been evaluated for a specific budget but have been discarded in a previous iteration. In this way, hyperparameter configurations that were already discarded are allowed to return to the pool of promising candidates. This variant will be called preserving Incremental-SuccessiveHalving algorithms and is given in Algorithm 3.

Algorithm 2 Discarding Incremental-SuccessiveHalving (`d-iSHA`)

Input: S set of arms, r, max size R, η, $(C_k)_k$ old sequence of configurations, $(L_k)_k$ old sequence of losses
Initialize: $S_0 \leftarrow S \cup C_0$, $n = |S_0|$, $s = \min\{t \in \mathbb{N} : nR(t+1)\eta^{-t} \leq B,\ t \leq \log_\eta(\min\{R, n\})\}$
for $k \in \{0, 1, \ldots, s\}$ **do**
  $n_k = \lfloor n/\eta^k \rfloor$, $r_k = r\eta^k$
  pull each arm in $S_k \backslash C_k$ for r_k times
  $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor$ arms from S_k
end for
Output: Remaining configuration

Algorithm 3 Preserving Incremental-SuccessiveHalving (`p-iSHA`)

Input: S set of arms, r, max size R, η, $(C_k)_k$ old sequence of configurations, $(L_k)_k$ old sequence of losses
Initialize: $S_0 \leftarrow S \cup C_0$, $n = |S_0|$, $s = \min\{t \in \mathbb{N} : nR(t+1)\eta^{-t} \leq B,\ t \leq \log_\eta(\min\{R, n\})\}$
for $k \in \{0, 1, \ldots, s\}$ **do**
  $n_k = \lfloor n/\eta^k \rfloor$, $r_k = r\eta^k$
  pull each arm in $S_k \backslash C_k$ for r_k times
  $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor$ arms from $S_k \cup C_k$
end for
Output: Remaining configuration

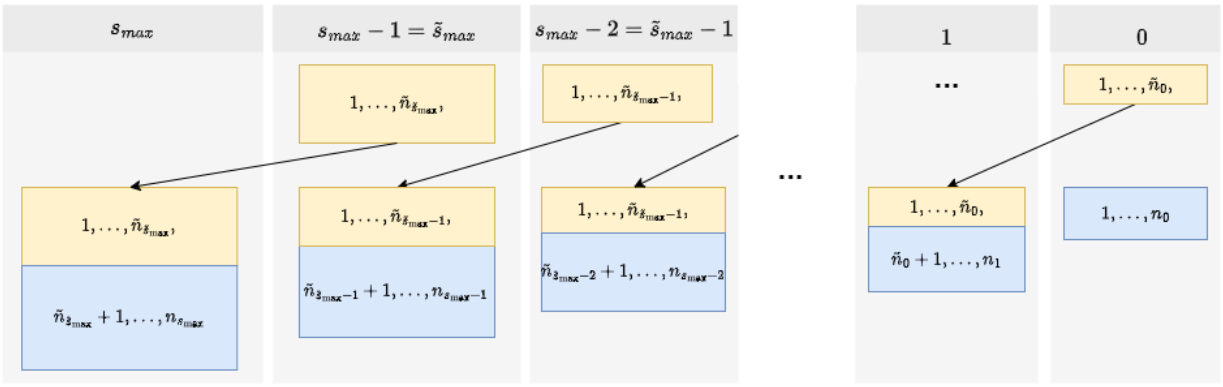


Figure 3: Illustration of how the brackets of incremental hyperband are arranged and filled up when the maximum budget R is increased.

Incremental Hyperband. The incremental Hyperband variant mentioned in Section 6.4 is given in Algorithm 4, where all differences to the original `Hyperband` algorithm by [Li *et al.*, 2018] are indicated by a blue text color.

Algorithm 4 Incremental-Hyperband (iHB)

Input: max size R, $\eta \geq 2$, old max size $\tilde{R} \in \{0, R/\eta\}$, old sequence of configuration samples $((C_{s,k})_{k\in\{0,\ldots,s\}})_{s\in\{0,\ldots,\log_\eta(\tilde{R})\}}$ and losses $((L_{s,k})_{k\in\{0,\ldots,s\}})_{s\in\{0,\ldots,\log_\eta(\tilde{R})\}}$
Initialize: $s_{max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{max}+1)R$
if $\tilde{R} > 0$ **then**
  $\tilde{s}_{max} = \lfloor \log_\eta(\tilde{R}) \rfloor = s_{max} - 1$, $\tilde{B} = (\tilde{s}_{max}+1)\tilde{R}$
end if
for $s \in \{s_{max}, s_{max}-1, \ldots, 0\}$ **do**
  $n_s = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r_s = R/\eta^s$
  if $\tilde{R} > 0$ and $s > 0$ **then**
    $\tilde{s} = s - 1$, $\tilde{n}_s = \lceil \frac{\tilde{B}}{\tilde{R}} \frac{\eta^{\tilde{s}}}{(\tilde{s}+1)} \rceil$, $\tilde{r}_s = \tilde{R}/\eta^{\tilde{s}} = r_s$
  else
    $\tilde{n}_s = 0$
  end if
  $S \leftarrow$ sample $n_s - \tilde{n}_s$ configurations
  x-iSHA$(S, r_s, R, \eta, (C_{\tilde{s},k})_{k\in\{0,\ldots,\tilde{s}\}}, (L_{\tilde{s},k})_{k\in\{0,\ldots,\tilde{s}\}})$
end for
Output: Configuration with smallest intermediate loss

B Proofs

B.1 Proof of Theorem 6.4

Proof of Theorem 6.4. This proof consists of two parts: First, we will focus on the efficient Incremental-SuccessiveHalving Algorithm given in Algorithm 1, i.e., iSHA. Second, we will show a similar lower bound on the number of necessary samples for the discarding and preserving Incremental-SuccessiveHalving algorithms given in Algorithm 2 and Algorithm 3.

Part I: iSHA analysis

Let $n_k = |S_k| + |C_k|$ and $\tilde{n}_k = |C_k|$ such that $n_0 = n$ and $\tilde{n}_o = \tilde{n}$. Without loss of generality, we assume that the limit values of the losses are ordered, such that $\nu_1 \leq \nu_2 \leq \cdots \leq \nu_n$. Note, that due to the above condition also the limit values of arms in S_k and resp. in C_k are ordered, e.g. for $\nu_i, \nu_j \in S_k$ with $i < j$ we have $\nu_i \leq \nu_j$. Let in the following be $i'_k = \min\{\lfloor n_k/\eta \rfloor + 1, \lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_k/\eta \rfloor + 1\}$. Also denote by B the total number of pulls by iSHA and assume that $B \geq z_{\mathrm{iSHA}}$, then we have for each round k

$$
\begin{aligned}
r_k &\geq \frac{B}{(n_k - \tilde{n}_k)\lceil \log_\eta(n) \rceil} - 1 \\
&\geq \frac{\eta}{n_k - \tilde{n}_k} \max_{i=2,\ldots,n} i\left(1 + \min\left\{R, \gamma^{-1}\Big(\max\left\{\frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2}\right\}\Big)\right\}\right) - 1 \\
&\geq \frac{\eta}{n_k - \tilde{n}_k}\, i'_k\left(1 + \min\left\{R, \gamma^{-1}\Big(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{i'_k} - \nu_1}{2}\right\}\Big)\right\}\right) - 1 \\
&\overset{(*)}{\geq} \frac{\eta}{n_k - \tilde{n}_k}\, (n_k - \tilde{n}_k)/\eta\left(1 + \min\left\{R, \gamma^{-1}\Big(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{i'_k} - \nu_1}{2}\right\}\Big)\right\}\right) - 1 \\
&= \left(1 + \min\left\{R, \gamma^{-1}\Big(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{i'_k} - \nu_1}{2}\right\}\Big)\right\}\right) - 1 \\
&= \min\left\{R, \gamma^{-1}\Big(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{i'_k} - \nu_1}{2}\right\}\Big)\right\},
\end{aligned}
$$

where the fourth line $(*)$ follows from:

- **Case 1:** $i'_k = \lfloor \tilde{n}_k/\eta \rfloor + 1$.
  We have
  $$i'_k \geq n_k/\eta \geq (n_k - \tilde{n}_k)/\eta.$$
- **Case 2:** $i'_k = \lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_{k-1}/\eta \rfloor + 1$.
  If $\tilde{n}_k = 0$, we have
  $$i'_k = \left\lfloor \frac{n_{k-1}}{\eta} \right\rfloor + 1 \geq \frac{n_{k-1}}{\eta} \geq \frac{n_k}{\eta} \geq \frac{n_k - \tilde{n}_k}{\eta}.$$
  If $\tilde{n}_k \geq 1$, we have
  $$
  \begin{aligned}
  i'_k &\geq \frac{\tilde{n}_k}{\eta} - 1 + \frac{n_{k-1}}{\eta} - 1 + 1 \\
  &= \frac{\tilde{n}_k - n_{k-1} - \eta}{\eta} \\
  &\geq \frac{n_k + (\eta - 1)n_k + \tilde{n}_k - \eta}{\eta} \\
  &\geq \frac{n_k}{\eta} \geq \frac{n_k - \tilde{n}_k}{\eta},
  \end{aligned}
  $$
  where line 3 follows from $n_k = \lfloor n_{k-1}/\eta \rfloor$ and line 4 from $n_k \geq \tilde{n}_k \geq 1$ and $\eta \geq 2$, so we have $\eta - 1 \geq 1$, so we can estimate $n_k \geq 1$.

Next, we show that $\ell_{i,t} - \ell_{1,t} > 0$ for all $t \geq \tau_i := \gamma^{-1}\big(\frac{\nu_i - \nu_1}{2}\big)$. Given the definition of γ, we have for all $i \in [n]$ that $|\ell_{i,t} - \nu_i| \leq \gamma(t) \leq \frac{\nu_i - \nu_1}{2}$ where the last inequality holds for $t \geq \tau_i$. Thus, for $t \geq \tau_i$ we have

$$
\begin{aligned}
\ell_{i,t} - \ell_{1,t} &= \ell_{i,t} - \nu_i + \nu_i - \nu_1 + \nu_1 - \ell_{1,t} \\
&= \ell_{i,t} - \nu_i - (\ell_{1,t} - \nu_1) + \nu_i - \nu_1 \\
&\geq -2\gamma(t) + \nu_i - \nu_1
\end{aligned}
$$

$$\geq -2\frac{\nu_i - \nu_1}{2} + \nu_i - \nu_1$$
$$= 0.$$

Under this scenario, we will eliminate arm i before arm 1 since on each round the arms are sorted by their empirical losses and the top half are discarded. Note that by the assumption the ν_i limits are non-decreasing in i so that the τ_i values are non-increasing in i.

Fix a round k and assume $1 \in S_k \cup C_k$ (note, $1 \in S_0 \cup C_0$). The above calculation shows that

$$t \geq \tau_i \implies \ell_{i,t} \geq \ell_{1,t}. \tag{1}$$

We regard two different scenarios in the following.

- **Case 1:** $k \leq s-1$.
  In this case, we keep the best $\lfloor n_k/\eta \rfloor - \lfloor \tilde{n}_k/\eta \rfloor$ arms from the set $S_k \cup C_k \backslash C_{k+1}$ and have already promoted the best $\lfloor \tilde{n}_k/\eta \rfloor$ from C_k.

$$\begin{aligned}
&\{1 \in S_k \cup C_k,\ 1 \notin S_{k+1} \cup C_{k+1}\} \\
&\quad\iff \left\{ \sum_{i \in S_k \cup C_k \backslash C_{k+1}} \mathbf{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor n_k/\eta \rfloor - \lfloor \tilde{n}_k/\eta \rfloor, \right. \\
&\qquad \left. \sum_{i \in C_k} \mathbf{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor \tilde{n}_k/\eta \rfloor \right\} \\
&\quad\implies \left\{ \sum_{i \in S_k \cup C_k \backslash C_{k+1}} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor n_k/\eta \rfloor - \lfloor \tilde{n}_k/\eta \rfloor, \right. \\
&\qquad \left. \sum_{i \in C_k} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor \tilde{n}_k/\eta \rfloor \right\} \\
&\quad\implies \left\{ \sum_{i=2}^{\lfloor n_k/\eta \rfloor - \lfloor \tilde{n}_k/\eta \rfloor + \lfloor \tilde{n}_k/\eta \rfloor + 1} \mathbf{1}\{r_k < \tau_i \wedge i \in S_k \cup C_k \backslash C_{k+1}\} \geq \lfloor n_k/\eta \rfloor \right. \\
&\qquad \left. - \lfloor \tilde{n}_k/\eta \rfloor, \sum_{i=2}^{\lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_{k-1}/\eta \rfloor + 1} \mathbf{1}\{r_k < \tau_i \wedge i \in C_k\} \geq \lfloor \tilde{n}_k/\eta \rfloor \right\} \\
&\quad\implies \left\{ r_k < \min\left\{ \tau_{\lfloor n_k/\eta \rfloor + 1}, \tau_{\lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_{k-1}/\eta \rfloor + 1} \right\} \right\} \\
&\quad\iff \left\{ r_k < \tau_{\max\{\lfloor n_k/\eta \rfloor + 1, \lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_{k-1}/\eta \rfloor + 1\}} \right\},
\end{aligned}$$

  where the first line follows by the definition of the algorithm and the second by Equation 1. In the third line we assume the worst case scenario, where the best $\lfloor n_k/\eta \rfloor - \lfloor \tilde{n}_k/\eta \rfloor$ arms in $S_k \cup C_k \backslash C_{k+1}$ are all worse than the best $\lfloor \tilde{n}_k/\eta \rfloor$ arms in C_k (which are kept in the set C_{k+1}) and vice versa that the best $\lfloor \tilde{n}_k/\eta \rfloor$ arms in C_k are worse than all arms in S_k. The fourth line follows by τ_i being non-increasing (for all $i < j$ we have $\tau_i \geq \tau_j$ and consequently, $\mathbf{1}\{r_k < \tau_i\} \geq \mathbf{1}\{r_k < \tau_j\}$ so the *first* indicators of the sum not including 1 would be on before any other i's in $S_k \subset [n]$ sprinkled throughout $[n]$).

- **Case 2:** $k = s$.
  In this case we keep the best $\lfloor n_k/\eta \rfloor$ arms from $S_k \cup C_k$ and have $C_{k+1} = \varnothing$, thus we get analogously as above

$$\begin{aligned}
\{1 \in S_k \cup C_k,\ \ 1 \notin S_{k+1}\} &\iff \left\{ \sum_{i \in S_k \cup C_k} \mathbf{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor n_k/\eta \rfloor \right\} \\
&\implies \left\{ \sum_{i \in S_k \cup C_k} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor n_k/\eta \rfloor \right\} \\
&\implies \left\{ \sum_{i=2}^{\lfloor n_k/\eta \rfloor + 1} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor n_k/\eta \rfloor \right\} \\
&\iff \left\{ r_k < \tau_{\lfloor n_k/\eta \rfloor + 1} \right\}.
\end{aligned}$$

Overall, we can conclude, that $1 \in S_k \cup C_k$ and $1 \notin S_{k+1} \cup C_{k+1}$ if $r_k < \tau_{\max\{\lfloor n_k/\eta \rfloor + 1, \lfloor \tilde{n}_k/\eta \rfloor + \lfloor n_k/\eta \rfloor + 1\}}$. This implies

$$\{1 \in S_k \cup C_k,\ \ r_k \geq \tau_{i'_k}\} \implies \{1 \in S_{k+1} \cup C_{k+1}\}. \tag{2}$$

Recalling that $r_k \geq \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{i'_k}-\nu_1}{2}\big\}\big)$ and $\tau_{i'_k} = \gamma^{-1}\big(\frac{\nu_{i'_k}-\nu_1}{2}\big)$, we examine the following three exhaustive cases:

- **Case 1:** $\frac{\nu_{i'_k}-\nu_1}{2} \geq \frac{\epsilon}{4}$ and $1 \in S_k \cup C_k$
  In this case, $r_k \geq \gamma^{-1}\big(\frac{\nu_{i'_k}-\nu_1}{2}\big) = \tau_{i'_k}$. By Equation 2 we have that $1 \in S_{k+1} \cup C_{k+1}$ since $1 \in S_k \cup C_k$.
- **Case 2:** $\frac{\nu_{i'_k}-\nu_1}{2} < \frac{\epsilon}{4}$ and $1 \in S_k \cup C_k$
  In this case $r_k \geq \gamma^{-1}\big(\frac{\epsilon}{4}\big)$ but $\gamma^{-1}\big(\frac{\epsilon}{4}\big) < \tau_{i'_k}$. Equation 2 suggests that it may be possible for $1 \in S_k \cup C_k$ but $1 \notin S_{k+1} \cup C_{k+1}$. On the good event that $1 \in S_{k+1} \cup C_{k+1}$, the algorithm continues and on the next round either case 1 or case 2 could be true. So assume $1 \notin S_{k+1} \cup C_{k+1}$. Here we show that $\{1 \in S_k \cup C_k,\ \ 1 \notin S_{k+1} \cup C_{k+1}\} \implies \max_{i \in S_{k+1} \cup C_{k+1}} \nu_i \leq \nu_1 + \epsilon/2$. Because $1 \in S_0 \cup C_0$, this guarantees that Algorithm 1 either exits with arm $\hat{i} = 1$ or some arm $\hat{i}$ satisfying $\nu_{\hat{i}} \leq \nu_1 + \epsilon/2$.
  Let $p = \min\{i \in [n] : \frac{\nu_i - \nu_1}{2} \geq \frac{\epsilon}{4}\}$. Note that $p > i'_k$ by the criterion of the case and
  $$r_k \geq \gamma^{-1}\left(\frac{\epsilon}{4}\right) \geq \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right) = \tau_i, \quad \forall i \geq p.$$
  Thus, by Equation 1 ($t \geq \tau_i \implies \ell_{i,t} \geq \ell_{1,t}$) we have that arms $i \geq p$ would always have $\ell_{i,r_k} \geq \ell_{1,r_k}$ and be eliminated before or at the same time as arm 1, presuming $1 \in S_k \cup C_k$. In conclusion, if arm 1 is eliminated so that $1 \in S_k \cup C_k$ but $1 \notin S_{k+1} \cup C_{k+1}$ then $\max_{i \in S_{k+1} \cup C_{k+1}} \nu_i \leq \max_{i<p} \nu_i < \nu_1 + \epsilon/2$ by the definition of p.
- **Case 3:** $1 \notin S_k \cup C_k$
  Since $1 \in S_0 \cup C_0$, there exists some $r < k$ such that $1 \in S_r \cup C_r$ and $1 \notin S_{r+1} \cup C_{r+1}$. For this r, only case 2 is possible since case 1 would proliferate $1 \in S_{r+1} \cup C_{r+1}$. However, under case 2, if $1 \notin S_{r+1} \cup C_{r+1}$ then $\max_{i \in S_{r+1} \cup C_{r+1}} \nu_i \leq \nu_1 + \epsilon/2$.

Because $1 \in S_0 \cup C_0$, we either have that 1 remains in $S_k \cup C_k$ (possibly alternating between cases 1 and 2) for all k until the algorithm exits with the best arm 1, or there exists some k such that case 3 is true and the algorithm exits with an arm $\hat{i}$ such that $\nu_{\hat{i}} \leq \nu_1 + \epsilon/2$.

Part II: iSHA variants analysis

Next, we proof the same guarantee for the discarding and preserving Incremental-SuccessiveHalving algorithms given in Algorithm 2 and Algorithm 3.
Therefore we proceed in two steps: First, we will reduce the `d-iSHA` algorithm to the SHA algorithm to take over its theoretical guarantees. Second, we will show where the proof of SHA has to be modified to achieve the same theoretical guarantees for our `p-iSHA` algorithm.

Step 1: We will distinguish two different cases in the following in order to reduce the discarding Incremental-SuccessiveHalving algorithm 2 to the original version of Successive Halving by [Jamieson and Talwalkar, 2016a] (or [Karnin *et al.*, 2013]).

- **Case 1:** $(C_k)_k = \varnothing$.
  If we have $(C_k)_k = \varnothing$, we have simply the Successive Halving algorithm by [Jamieson and Talwalkar, 2016a] and can keep their theoretical guarantees.
- **Case 2:** $(C_k)_k \neq \varnothing$.
  Thus the interesting case which we will consider in the following is the case $(C_k)_k \neq \varnothing$. Assume that Algorithm 2 is called as subroutine by Algorithm 4. Since $(C_k)_k \neq \varnothing$, Algorithm 2 was already called before with number of arms $\tilde{n}$ and budget $\tilde{r}_s = \tilde{R}/\eta^{\tilde{s}} = \frac{R}{\eta}/\eta^{s-1} = R/\eta^s = r_k$ for $s \in \{0, \dots, \lfloor \log_\eta(R) \rfloor\}$. Thus, the arms in $(C_k)_k$ were already pulled for r_k times and their loss values $(L_k)_k$ were observed. Combining these with the loss values we observe in each iteration k in Algorithm 2 for r_k pulls of the arms in $S_k \backslash C_k$, we can keep the best $\lfloor n/\eta^{k+1} \rfloor$ arms from S_k regarding the observed losses of the recent pulls of $S_k \backslash C_k$ and the before observed losses of C_k. Therefore, we get the same arms in S_{k+1} as starting Algorithm 3 from scratch with $(C_k)_k = \varnothing$ and $S = S \cup C_0$ and can apply Case 1.

To conclude both cases, we can keep the theoretical result that was proven by [Li *et al.*, 2018] for the original version of Successive Halving in a finite horizon setting ($R < \infty$).

Step 2: To achieve the same guarantee for the preserving Incremental-SuccessiveHalving algorithm, we can proceed analogue as in the proof of Successive Halving by [Li *et al.*, 2018]. For a fixed round k and $1 \in S_k \cup C_k$, since $1 \in S_0 \cup C_0$, we have

$$\{1 \in S_k \cup C_k, 1 \notin S_{k+1}\} \Leftrightarrow \left\{ \sum_{i \in S_k \cup C_k} \mathbf{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor n_k/\eta \rfloor \right\}$$

$$\Rightarrow \left\{ \sum_{i \in S_k \cup C_k} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor n_k/\eta \rfloor \right\}$$
$$\Rightarrow \left\{ \sum_{i=2}^{n_k + |C_k \setminus (S_k \cap C_k)| + 1} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor n_k/\eta \rfloor \right\}$$
$$\Leftrightarrow \{r_k < \tau_{\lfloor n_k/\eta \rfloor + 1}\}.$$

The rest of the proof is the same as that for Successive Halving in [Li *et al.*, 2018]. □

B.2 Comparison of SHA and `iSHA`

Proof of Theorem 6.5. Let us first regard the number of total pulls when we run SHA in comparison to a run of iSHA, where we assume that we had already run SHA for $\tilde{n}$ many arms and $r = 1$ which is equal to $r = R/\eta^s$ for $s = \log_\eta(R)$. We concentrate in the following on a lower bound on the pulls of SHA re-used on n arms.

$$\begin{aligned}
\sum_{k=0}^{s} n_k r_k &= \sum_{k=0}^{s} \left\lfloor \frac{n}{\eta^k} \right\rfloor \cdot \left\lfloor \frac{R\eta^k}{\eta^s} \right\rfloor \\
&\geq \sum_{k=0}^{s} \left(\frac{n}{\eta^k} - 1 \right) \left(\frac{R\eta^k}{\eta^s} - 1 \right) \\
&= \sum_{k=0}^{s} \frac{nR}{\eta^s} - \frac{R\eta^k}{\eta^s} - \frac{n}{\eta^k} + 1 \\
&= \frac{(s+1)(nR+\eta^s)}{\eta^s} - \frac{R}{\eta^s} \sum_{k=0}^{s} \eta^k - n \sum_{k=0}^{s} \left(\frac{1}{\eta} \right)^k \\
&= \frac{(s+1)(nR+\eta^s)}{\eta^s} - \frac{R(\eta^{s+1}-1)}{\eta^s(\eta-1)} - \underbrace{\frac{n(1-(1/\eta)^{s+1})}{1-1/\eta}}_{= \frac{n\left(\frac{\eta^{s+1}-1}{\eta^{s+1}}\right)}{\frac{\eta-1}{\eta}} = \frac{n(\eta^{s+1}-1)}{\eta^s(\eta-1)}} \\
&= \frac{(s+1)(nR+\eta^s)}{\eta^s} - \frac{(\eta^{s+1}-1)(R+n)}{\eta^s(\eta-1)} \\
&= \frac{(s+1)(nR+\eta^s)(\eta-1) - (\eta^{s+1}-1)(R+n)}{\eta^s(\eta-1)},
\end{aligned}$$

where we used the closed form for the geometric series in the fifth line and simple transformations in all other lines.

An upper bound on the total pulls of iSHA(n, r) is given by

$$\begin{aligned}
\sum_{k=0}^{s} n_k r_k &= \sum_{k=0}^{s} \left(\lfloor n/\eta^k \rfloor - \lfloor \tilde{n}/\eta^k \rfloor \right) \left\lfloor \frac{R\eta^k}{\eta^s} \right\rfloor \\
&\leq \sum_{k=0}^{s} \left(\frac{n-\tilde{n}}{\eta^k} + 1 \right) \cdot \frac{R\eta^k}{\eta^s} \\
&= \frac{(s+1)(n-\tilde{n})R}{\eta^s} + \frac{R}{\eta^s} \sum_{k=0}^{s} \eta^k \\
&= \frac{(s+1)(n-\tilde{n})R}{\eta^s} + \frac{R(\eta^{s+1}-1)}{\eta^s(\eta-1)} \\
&= \frac{(s+1)(n-\tilde{n})R(\eta-1) + R(\eta^{s+1}-1)}{\eta^s(\eta-1)}.
\end{aligned}$$

Finally, we compare both by building the quotient

$$\begin{aligned}
\frac{\#\{\text{total pulls of } \texttt{iSHA}\ (n,r)\}}{\#\{\text{total pulls of SH}(n,r)\}} &\leq \frac{(s+1)(n-\tilde{n})R(\eta-1) + R(\eta^{s+1}-1)}{(s+1)(nR+\eta^s)(\eta-1) - (\eta^{s+1}-1)(R+n)} \\
&= 1 - \frac{(s+1)(\tilde{n}R+\eta^s)(\eta-1) - (\eta^{s+1}-1)(2R+n)}{(s+1)(nR+\eta^s)(\eta-1) - (\eta^{s+1}-1)(R+n)}.
\end{aligned}$$

□

It is worth mentioning that we can do a similar analysis for the discarding and preserving Incremental-SuccessiveHalving algorithms given in Algorithm 2 and Algorithm 3:

Analogously as in the proof of Theorem 6.5, we first need an upper bound on the total pulls in a run of d-iSHA(n, r). While we only sample $n - \tilde{n}$ new arms in the first round of `d-iSHA`, the best n/η arms may be all from the newly sampled ones and thus none of the arms which are kept into the next round of `d-iSHA` was already pulled with a higher budget in the run of SH$(\tilde{n}, \tilde{r})$. In this worst case, we can estimate

$$\begin{aligned}\sum_{k=0}^{s} n_k r_k &= (n-\tilde{n})\left\lfloor \frac{R}{\eta^s} \right\rfloor + \sum_{k=1}^{s} \left\lfloor \frac{n}{\eta^k} \right\rfloor \left\lfloor \frac{R\eta^k}{\eta^s} \right\rfloor \\ &\leq \frac{(n-\tilde{n})R}{\eta^s} + \sum_{k=1}^{s} \frac{nR}{\eta^s} \\ &= \frac{(n-\tilde{n})R}{\eta^s} + \frac{snR}{\eta^s} \\ &= \frac{R((s+1)n-\tilde{n})}{\eta^s}.\end{aligned}$$

Again, we can now compute the quotient of the pulls as follows.

$$\begin{aligned}\frac{\#\{\text{total pulls of } \texttt{d-iSHA}\ (n,r)\}}{\#\{\text{total pulls of SH}(n,r)\}} &\leq \frac{(\eta-1)R((s+1)n-\tilde{n})}{(\eta-1)(s+1)(nR+\eta^s)-(\eta^{s+1}-1)(R+n)} \\ &= 1 - \frac{(\eta-1)((s+1)\eta^s+R\tilde{n})-(\eta^{s+1}-1)(R+n)}{(\eta-1)(s+1)(nR+\eta^s)-(\eta^{s+1}-1)(R+n)}.\end{aligned}$$

Note that we can apply the same for the number of pulls of `p-iSHA` since we have the same worst-case scenario where we only keep newly sampled configurations into the next round of `p-iSHA` and none of the previously promoted configurations.

To get an intuition for the improvement in the number of total pulls, we show in Figure 4 and Figure 5 the above terms for different values of rounds s, maximal budgets per round R and discarding portion η. Note that the above results assume the worst-case scenario for the `p-iSHA` resp. the `d-iSHA` algorithm in which all previously promoted configurations perform worse than all newly sampled ones. In most problem scenarios the average improvement in the number of total pulls of `p-iSHA` resp. `d-iSHA` will lie between the plotted curves of the worst case scenario in Figure 4 and the best case scenario which coincidences with `iSHA` and is shown in Figure 5. Since our proposed methods will never need a greater number of total pulls than SH, we plotted the minimum value of 1 and our derived fractions in Theorem 6.5.

Proof of Corollary 6.6. In the following, we only regard the asymptotic behavior of the number of pulls for an infinite large budget and when $r = R/\eta^s$ and $s = \log_\eta(R)$. In this case, we can ignore the flooring functions since the asymptotic behavior is not affected by those. We get for the asymptotic ratio between the number of pulls of rerunning SHA and iSHA that

$$\lim_{s\to\infty} \frac{\sum_{k=0}^{s}\left(\lfloor n/\eta^k \rfloor - \lfloor \tilde{n}/\eta^k \rfloor\right)\left\lfloor \frac{R\eta^k}{\eta^s} \right\rfloor}{\sum_{k=0}^{s}\left\lfloor \frac{n}{\eta^k} \right\rfloor \cdot \left\lfloor \frac{R\eta^k}{\eta^s} \right\rfloor} = \lim_{s\to\infty} \frac{\sum_{k=0}^{s} \frac{(n-\tilde{n})R}{\eta^s}}{\sum_{k=0}^{s} \frac{nR}{\eta^s}} = \frac{n-\tilde{n}}{n} = 1 - x^{-1},$$

where we used that in each new run of iSHA it holds that $n = |S| + |C_0| = |C_0| \cdot x$, where C_0 is the number of configurations in the previous run with cardinality $\tilde{n}$. □

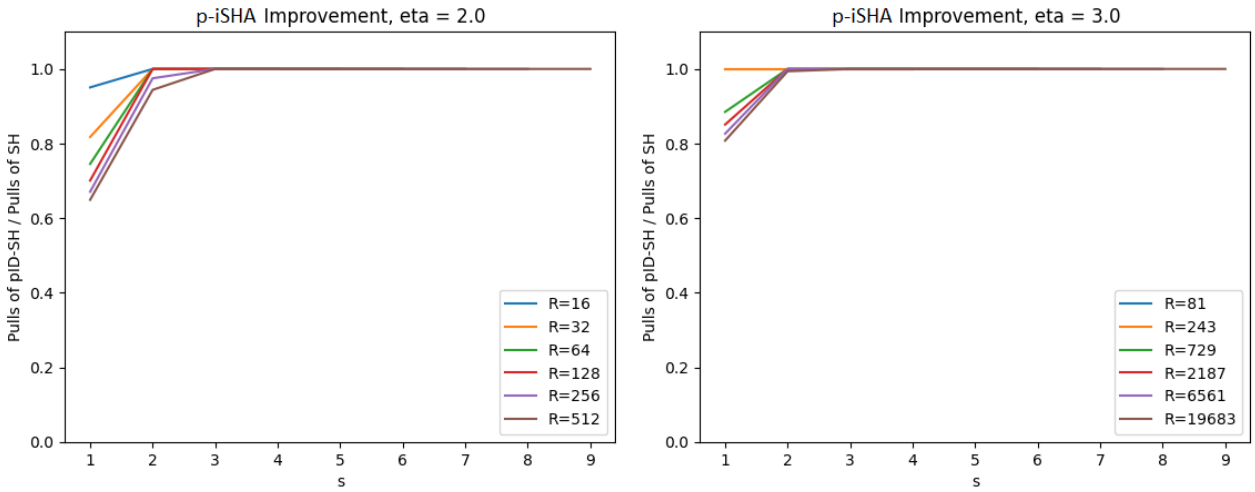


Figure 4: Fraction of the number of total pulls of `p-iSHA` resp. `d-iSHA` and SHA for different values of rounds of SHA s, maximal budgets per round R and discarding fraction η.

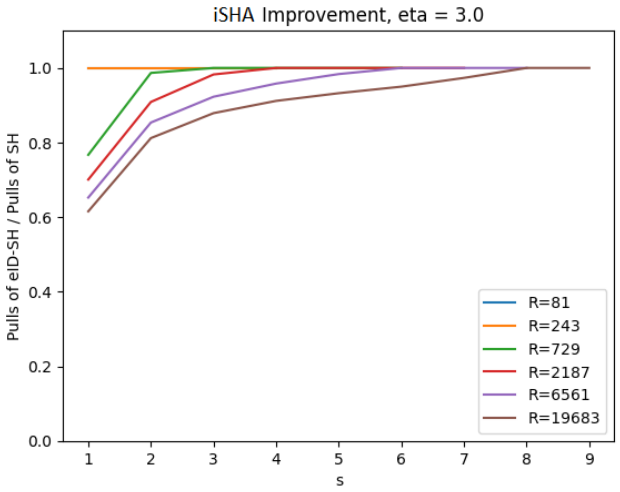


Figure 5: Fraction of number of total pulls of `iSHA` for different values of rounds of SHA s and maximal budgets per round R.

B.3 Incremental-Hyperband

Proof of Theorem 6.8. To derive the necessary budget of iHB in Algorithm 4, we simply have to sum up all necessary budgets for each call of xiSHA. Luckily, the necessary budgets for iSHA, d-iSHA and p-iSHA do not differ, thus a run of iHB uses independent of the variant of the called Successive Halving algorithm a total budget of

$$\begin{aligned}
&\sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \text{Budget_xiSHA}(S_s, r_s, R, \eta, (C_{\tilde{s},k})_{k\in\{0,\dots,\tilde{s}\}}, (L_{\tilde{s},k})_{k\in\{0,\dots,\tilde{s}\}}) \\
&\quad = \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \eta \left\lceil \log_\eta \left(\left\lceil (\lfloor \log_\eta(R) \rfloor + 1) \cdot \frac{\eta^s}{s+1} \right\rceil \right) \right\rceil \\
&\qquad \cdot \max_{i=2,\dots,n_s} i \left(1 + \min \left\{ R, \gamma^{-1} \left(\max \left\{ \frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2} \right\} \right) \right\} \right) \\
&\quad = (*).
\end{aligned}$$

Here, we denoted by S_s the set of configurations sampled in round s. Due to simple estimates and transformations, we get

$$\begin{aligned}
\eta \left\lceil \log_\eta \left(\left\lceil (\lfloor \log_\eta(R) \rfloor + 1) \cdot \frac{\eta^s}{s+1} \right\rceil \right) \right\rceil &\le \eta \left\lceil \log_\eta \left(\left\lceil (\log_\eta(R) + 1) \right\rceil \cdot \left\lceil \frac{\eta^s}{s+1} \right\rceil \right) \right\rceil \\
&= \eta \left\lceil \log_\eta \left(\left\lceil \log_\eta(R) + 1 \right\rceil \right) + \log_\eta \left(\left\lceil \frac{\eta^s}{s+1} \right\rceil \right) \right\rceil \\
&\le \eta \left\lceil \log_\eta \left(\log_\eta(R) + 2 \right) + \log_\eta \left(\frac{\eta^s}{s+1} + 1 \right) \right\rceil \\
&\le \eta \left\lceil \log_\eta \left(\log_\eta(R) \right) + 2 + \log_\eta \left(\frac{\eta^s}{s+1} \right) + 1 \right\rceil \\
&= \eta \left\lceil \log_\eta \left(\log_\eta(R) \right) + \log_\eta \left(\eta^s \right) - \log_\eta \left(s+1 \right) + 3 \right\rceil \\
&\le \eta \left(\log_\eta \left(\log_\eta(R) \right) + s - \log_\eta \left(s+1 \right) + 4 \right).
\end{aligned}$$

Note that the fourth line follows from

$$\begin{aligned}
&\log_\eta(x+1) \le \log_\eta(x) + 1 \\
&\Leftrightarrow x + 1 \le \eta \cdot x \\
&\Leftrightarrow x \ge \frac{1}{\eta - 1}.
\end{aligned}$$

In our setting, we have $\eta \ge 2$, thus $\log_\eta(x+1) \ge \log_\eta(x) + 1$ if and only if $x \ge 1$. We have $\frac{\eta^s}{s+1} \ge 1$ for $s \ge 0$ and also wlog. $\log_\eta(R) \ge 2$, otherwise the value of s_{max} and thus the run of Hyperband would be trivial.
We can continue with

$$\begin{aligned}
(*) &\le \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \eta \left(\log_\eta \left(\log_\eta(R) \right) + s - \log_\eta \left(s+1 \right) + 4 \right) \\
&\quad \underbrace{\cdot \max_{s=0,\dots,\lfloor \log_\eta(R) \rfloor} \max_{i=2,\dots,n_s} i \left(1 + \min \left\{ R, \gamma^{-1} \left(\max \left\{ \frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2} \right\} \right) \right\} \right)}_{=:\bar{\gamma}^{-1}} \\
&= \eta \left(\left(\lfloor \log_\eta(R) \rfloor + 1 \right) \left(\log_\eta(\log_\eta(R)) + 4 \right) + \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} s - \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \log_\eta(s+1) \right) \bar{\gamma}^{-1} \\
&= \eta \left(\left(\lfloor \log_\eta(R) \rfloor + 1 \right) \left(\log_\eta(\log_\eta(R)) + 4 \right) + \frac{\lfloor \log_\eta(R) \rfloor \left(\lfloor \log_\eta(R) \rfloor + 1 \right)}{2} \right. \\
&\quad \left. - \log_\eta \left(\prod_{s=0}^{\lfloor \log_\eta(R) \rfloor} (s+1) \right) \right) \bar{\gamma}^{-1} \\
&= \eta \left(\left(\lfloor \log_\eta(R) \rfloor + 1 \right) \left(\log_\eta(\log_\eta(R)) + 4 \right) + \frac{\lfloor \log_\eta(R) \rfloor \left(\lfloor \log_\eta(R) \rfloor + 1 \right)}{2} \right.
\end{aligned}$$

$$- \log_\eta \left(\left(\lfloor \log_\eta(R) \rfloor + 1 \right)! \right) \Bigg) \bar{\gamma}^{-1}.$$

Since we choose the budget B in our iHB algorithm as $B = (s_{\max} + 1)R = \left(\lfloor \log_\eta(R) \rfloor + 1 \right) R$, we can divide both by $\left(\lfloor \log_\eta(R) \rfloor + 1 \right)$ and get

$$R \geq \eta \left(\log_\eta(\log_\eta(R)) + 4 + \frac{\lfloor \log_\eta(R) \rfloor}{2} - \frac{\log_\eta \left(\left(\lfloor \log_\eta(R) \rfloor + 1 \right)! \right)}{\lfloor \log_\eta(R) \rfloor + 1} \right) \bar{\gamma}^{-1}.$$

Recall that in each call of `xiSHA` in round s of iHB we compare $n_s = \left(\lfloor \log_\eta(R) \rfloor + 1 \right) \frac{\eta^s}{s+1}$ hyperparameter configurations, thus we get an overall number of samples of

$$\begin{aligned}
& \left(\lfloor \log_\eta(R) \rfloor + 1 \right) \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \frac{\eta^s}{s+1} \\
\geq & \sum_{s=0}^{\lfloor \log_\eta(R) \rfloor} \eta^s \\
\overset{\text{Geometric Sum}}{=} & \frac{\eta^{\lfloor \log_\eta(R) \rfloor + 1} - 1}{\eta - 1} \\
\geq & \frac{R-1}{\eta - 1}.
\end{aligned}$$

By assumption 6.7 we have an ϵ-optimal hyperparameter configuration in our sample set with probability at least $1 - \delta$ if and only if

$$\begin{aligned}
& \frac{R-1}{\eta - 1} \geq \lceil \log_{1-\alpha}(\delta) \rceil \\
\Leftrightarrow \; & R \geq \lceil \log_{1-\alpha}(\delta) \rceil (\eta - 1) + 1.
\end{aligned}$$

□

C Theoretical Analysis of ASHA

Proof of Theorem 6.2. For sake of simplicity, we denote the configurations by their indices and assume without loss of generality that the configurations are ordered, such that the optimal configuration has index 1, the second best 2 etc. In the worst case, we observe the configurations in such an order that we have never two consistent rankings in two successive rungs, thus we have to enlarge the rung each time we have at least η configurations in the recent top rung K. With this, we get in each rung k at least $\lfloor n/\eta^k \rfloor$ configurations for which the budget $b_k = r\eta^k$ is used by algorithm design. In addition, we can bound the index of the top rung K by $\lfloor \log_\eta(n) \rfloor$, because we have at least η times many configurations in rung $k+1$ in comparison to rung k and the first rung has index 0. We can compute the following upper bound for the budget of ASHA:

$$\begin{aligned} B &= \sum_{k=0}^{K} |\text{rung}_k| \cdot b_k \\ &\leq \sum_{k=0}^{K} \left\lfloor \frac{n}{\eta^k} \right\rfloor \cdot r\eta^k \\ &\leq \sum_{k=0}^{K} r \cdot n \\ &= (K+1) \cdot rn \\ &\leq (\lfloor \log_\eta(n) \rfloor + 1) rn. \end{aligned}$$

By simple transformations we get

$$r \geq \frac{B}{(\lfloor \log_\eta(n) \rfloor + 1)n}.$$

We prove the correctness of ASHA indirectly, so we assume in the following that ASHA does not return the near-optimal configuration. For sake of convenience, we write $[K-1]_0 = \{0, 1, 2, \ldots, K-1\}$. We can regard two different cases for this scenario.

- **Case 1:** Configuration 1 does not even reach the top rung K.

$$\begin{aligned} 1 \notin \text{rung}_K \;&\Leftrightarrow\; \exists k \in [K-1]_0 :\; 1 \in \text{rung}_k \wedge 1 \notin \text{rung}_{k+1} \\ &\Leftrightarrow \exists k \in [K-1]_0 : \sum_{i \in \text{rung}_k \setminus \{1\}} \mathbf{1}\{\ell_{1,b_k} > \ell_{i,b_k}\} > \frac{|\text{rung}_k|}{\eta} \\ &\Leftrightarrow \exists k \in [K-1]_0 : \sum_{i \in \text{rung}_k \setminus \{1\}} \mathbf{1}\{\nu_i - \nu_1 < \nu_i - \ell_{i,b_k} - \nu_1 + \ell_{1,b_k}\} > \frac{|\text{rung}_k|}{\eta} \\ &\Rightarrow \exists k \in [K-1]_0 : \sum_{i \in \text{rung}_k \setminus \{1\}} \mathbf{1}\{\nu_i - \nu_1 < |\nu_i - \ell_{i,b_k}| + |\nu_1 - \ell_{1,b_k}|\} > \frac{|\text{rung}_k|}{\eta} \\ &\Rightarrow \exists k \in [K-1]_0 : \sum_{i \in \text{rung}_k \setminus \{1\}} \mathbf{1}\{\nu_i - \nu_1 < 2\gamma(b_k)\} > \frac{|\text{rung}_k|}{\eta} \\ &\Rightarrow \exists k \in [K-1]_0 :\; \nu_{\lfloor |\text{rung}_k|/\eta \rfloor + 1} - \nu_1 < 2\gamma(b_k) \\ &\Rightarrow \exists k \in [K-1]_0 :\; r\eta^k = b_k < \gamma^{-1}\left(\frac{\nu_{\lfloor |\text{rung}_k|/\eta \rfloor + 1} - \nu_1}{2}\right) \\ &\Leftrightarrow \exists k \in [K-1]_0 :\; r < \eta^{-k}\gamma^{-1}\left(\frac{\nu_{\lfloor |\text{rung}_k|/\eta \rfloor + 1} - \nu_1}{2}\right) \\ &\Rightarrow \frac{B}{(\lfloor \log_\eta(n) \rfloor + 1)n} \leq r < \max_{k \in [K]} \eta^{-k}\gamma^{-1}\left(\frac{\nu_{\lfloor |\text{rung}_{k-1}|/\eta \rfloor + 1} - \nu_1}{2}\right) \\ &\Rightarrow B < (\lfloor \log_\eta(n) \rfloor + 1)n \max_{k \in [K]} \eta^{-k}\gamma^{-1}\left(\frac{\nu_{\lfloor |\text{rung}_{k-1}|/\eta \rfloor + 1} - \nu_1}{2}\right). \end{aligned}$$

  By contradiction, we get the following condition for the budget of ASHA to ensure that configuration 1 gets promoted into the top rung K:

$$B \geq (\lfloor \log_\eta(n) \rfloor + 1)n \max_{k \in [K]} \eta^{-k}\gamma^{-1}\left(\frac{\nu_{\lfloor |\text{rung}_{k-1}|/\eta \rfloor + 1} - \nu_1}{2}\right).$$

- **Case 2:** Configuration 1 is contained in the top rung K, but is not returned by ASHA.

$$\begin{aligned}
&\text{return(ASHA)} \neq \{1\} \wedge 1 \in \text{rung}_K \\
&\quad \Leftrightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ \ell_{1,b_K} > \ell_{i,b_K} \\
&\quad \Leftrightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ \nu_i - \nu_1 < \nu_i - \ell_{i,b_K} - \nu_1 + \ell_{1,b_K} \\
&\quad \Rightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ \nu_i - \nu_1 < |\nu_i - \ell_{i,b_K}| + |\nu_1 - \ell_{1,b_K}| \\
&\quad \Rightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ \nu_i - \nu_1 < 2\gamma(b_K) \\
&\quad \Rightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ r\eta^K = b_K < \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right) \\
&\quad \Rightarrow \exists i \in \text{rung}_K \backslash \{1\} \ : \ \frac{B}{(\lfloor \log_\eta(n) \rfloor + 1)n} \leq r < \eta^{-K}\gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right) \\
&\quad \Rightarrow B < (\lfloor \log_\eta(n) \rfloor + 1)n\eta^{-K} \max_{i \in \text{rung}_K \backslash \{1\}} \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right).
\end{aligned}$$

By contradiction, ASHA returns a near-optimal solution if it is contained in the top rung K and if

$$B \geq (\lfloor \log_\eta(n) \rfloor + 1)n\eta^{-K} \max_{i \in \text{rung}_K \backslash \{1\}} \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right).$$

To summarize both cases, ASHA needs a budget of

$$B \geq (\lfloor \log_\eta(n) \rfloor + 1)n \cdot \max\left\{ \max_{k \in [K]} \eta^{-k}\gamma^{-1}\left(\tfrac{\nu_{\lfloor |\text{rung}_{k-1}|/\eta \rfloor + 1} - \nu_1}{2}\right), \eta^{-K} \max_{i \in \text{rung}_K \backslash \{1\}} \gamma^{-1}\left(\tfrac{\nu_i - \nu_1}{2}\right) \right\}$$

to ensure that the optimal configuration will be returned. □

D Lower Bound for $\frac{\epsilon}{2}$-optimal Arm Identification

For a given limit vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n) \in \mathbb{R}^n$ and a convergence rate sequence $\boldsymbol{\gamma} := (\gamma_i(t))_{i \in [n], t \in \mathbb{N}}$, we write $\mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$ for the set of all loss sequences $\ell = (\ell_i(t))_{i \in [n], t \in \mathbb{N}}$ that fulfill

(i) $\forall i \in [n]$: $\nu_i' = \lim_{t \to \infty} \ell_i(t)$ exists,

(ii) $\forall i \in [n], t \in \mathbb{N}$: $|\ell_i(t) - \nu_i'| \leq \gamma_i(t)$,

(iii) $\exists \pi : [n] \to [n]$ bijective such that $\nu_i' = \nu_{\pi(i)}$ for all $i \in [n]$.

For sake of convenience, we will simply write $\ell_{i,t}$ for $\ell_i(t)$ and likewise $\gamma_{i,t}$ for $\gamma_i(t)$.

For a given limit vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n) \in \mathbb{R}^n$ and some $\epsilon > 0$, we write $\text{OPT}_\epsilon(\boldsymbol{\nu})$ for the set of all optimal arms:

$$\text{OPT}_\epsilon(\boldsymbol{\nu}) := \left\{ i \in [n] \,\middle|\, \nu_i - \arg\min_{k \in [n]} \nu_k \leq \epsilon \right\}.$$

If Alg is a (possibly probabilistic) sequential algorithm, we denote by $B(\text{Alg}, \ell)$ the number of total pulls made by Alg before termination when used for the loss sequence $\ell \in \mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$. Similarly, $B_i(\text{Alg}, \ell)$ denotes the number of pulls of arm i, when we run Alg on the loss function ℓ. We write $\text{Alg}(\ell)$ for the output of Alg executed on the instance ℓ. In the following, we provide lower bounds on $\mathbb{E}[B(\text{Alg}, \ell)]$ for algorithms Alg, which identify, for any sequence of losses $\ell \in \mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$, almost surely an $\epsilon/2$-optimal arm, i.e., $P(\text{Alg}(\ell) \in \text{OPT}_{\epsilon/2}(\boldsymbol{\nu})) = 1$.

The proof of the lower bound for $\epsilon/2$-optimal arm identification is prepared with the next lemma.

Lemma D.1. *Let Alg be a deterministic solution for the $\epsilon/2$-optimal arm identification problem and $\ell, \ell' \in \mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$ be two loss sequences.*

(i) If $Alg(\ell) \neq Alg(\ell')$, then

$$\exists i \in [n],\ t \in \{1, \ldots, \min\{B_i(Alg, \ell), B_i(Alg, \ell')\}\} \ : \ \ell_{i,t} \neq \ell'_{i,t}.$$

(ii) If ℓ and ℓ' coincide on $\{t < B'\}$ and on $I \subset [n]$ in the sense that

$$\forall i \in [n],\ \forall t < B' \ : \ \ell_{i,t} = \ell'_{i,t} \tag{3}$$

and

$$\forall i \in I,\ \forall t \in \mathbb{N} \ : \ \ell_{i,t} = \ell'_{i,t} \tag{4}$$

then $Alg(\ell) \neq Alg(\ell')$ implies

$$\exists i \in [n] \backslash I \ : \ \min\{B_i(Alg, \ell), B_i(Alg, \ell')\} \geq B'.$$

Proof. (i) To prove the contraposition, suppose that

$$\forall i \in [n],\, t \in \{1, \dots, \min\{B_i(\text{Alg}, \ell), B_i(\text{Alg}, \ell')\}\} : \ell_{i,t} = \ell'_{i,t} \tag{5}$$

holds.
Claim 1: $B_i(\text{Alg}, \ell) = B_i(\text{Alg}, \ell')$ for any $i \in [n]$.
Proof: Assume this was not the case. Let $i \in [n]$ be the first set, for which Alg exceeds its budget on one of ℓ, ℓ' but does not reach it on the other instance, and suppose w.l.o.g. $B_i(\text{Alg}, \ell) > B_i(\text{Alg}, \ell')$. Since Alg has observed until this point exactly the same feedback on ℓ as on ℓ', this is a contradiction as Alg is deterministic. ■
Combining Claim 1 and equation 5 yields that Alg observes on ℓ exactly the same feedback as on ℓ' until its termination. Since Alg is deterministic, this implies $\text{Alg}(\ell) = \text{Alg}(\ell')$.

(ii) If $\text{Alg}(\ell) \neq \text{Alg}(\ell')$, then (i) together with equation 4 yields

$$\exists i \in [n] \backslash I, t \leq \min\{B_i(\text{Alg}, \ell), B_i(\text{Alg}, \ell')\} : \ell_{i,t} \neq \ell'_{i,t},$$

and thus equation 3 implies

$$\exists i \in [n] \backslash I : \min\{B_i(\text{Alg}, \ell), B_i(\text{Alg}, \ell')\} \geq B'.$$

□

Lemma D.1 is the main ingredient for the proof of the lower bound in Theorem 6.1, since we first derive the lower bound for deterministic algorithms and then apply Yao's minimax principle [Yao, 1977] to infer the lower bound for any randomized algorithm.

Given our notation, we state Theorem 6.1 in the following more formal way:

Theorem D.2 (Lower bound for $\epsilon/2$-optimal Arm Identification)**.** *Let $\boldsymbol{\nu}$ be such that $|\text{OPT}_0(\boldsymbol{\nu})| = 1$, i.e., $\arg\min_i \nu_i$ is unique. If* Alg *is a (possibly probabilistic) sequential algorithm that correctly identifies almost surely an $\epsilon/2$-optimal arm for any loss sequence in $\mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$, then there exists $\ell \in \mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$ such that*

$$\mathbb{E}[B(Alg, \ell)] \geq n \cdot \gamma^{-1}\left(\max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\}\right).$$

Proof of Theorem D.2. We split the proof into two parts.

Part 1: The statement holds in case Alg is a deterministic algorithm.
Abbreviate $B' \coloneqq \gamma^{-1}\left(\max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\}\right)$ and define $\gamma(t) = \max_{i \in [n]} \gamma_{i,t}$. Without loss of generality assume that $\nu_1 < \nu_2 \leq \ldots \leq \nu_n$, as otherwise we can find a permutation $\pi_n : [n] \mapsto [n]$ of the indicies such that this holds. Now, define $\ell = (\ell_{i,t})_{i \in [n], t \in \mathbb{N}}$ via

$$\ell_{i,t} \coloneqq \begin{cases} \frac{\nu_1 + \nu_i}{2}, & \text{if } t < B', \\ \nu_i + \frac{\epsilon}{2}, & \text{if } t \geq B' \text{ and } \nu_i - \nu_1 \leq \frac{\epsilon}{2}, \\ \nu_i, & \text{else.} \end{cases}$$

Since $\nu_1 < \nu_2 \leq \cdots \leq \nu_n$ we obtain that $\text{OPT}_{\epsilon/2}(\boldsymbol{\nu}) = \{1\}$. For $t < B' \leq \gamma^{-1}\left(\max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{2}\right\}\right)$, which implies $\gamma(t) \geq \max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{2}\right\}$, we have due to $\nu_1 \leq \nu_i \leq \nu_n$ the following inequalities for a fixed but unknown arm $i \in [n]$:

In the case i is an $\frac{\epsilon}{2}$-optimal arm, we get

$$\begin{aligned} |\ell_{i,t} - \lim_{t \to \infty} \ell_{i,t}| &= \left|\frac{\nu_1 + \nu_i}{2} - \nu_i - \frac{\epsilon}{2}\right| \\ &\leq \left|\frac{\nu_1 + \nu_i}{2} - \nu_1 - \frac{\epsilon}{2}\right| \\ &\leq \left|\frac{\nu_i - \nu_1}{2} - \frac{\epsilon}{2}\right| \leq \left|\frac{\epsilon}{4} - \frac{\epsilon}{2}\right| = \frac{\epsilon}{4} \\ &\leq \max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\} \leq \gamma(t). \end{aligned}$$

In the other case, i.e., in which i is not $\frac{\epsilon}{2}$-optimal, we have

$$\begin{aligned} |\ell_{i,t} - \lim_{t \to \infty} \ell_{i,t}| &= \left|\frac{\nu_1 + \nu_i}{2} - \nu_i\right| \\ &= \left|\frac{\nu_1 - \nu_i}{2}\right| = \frac{\nu_i - \nu_1}{2} \leq \frac{\nu_n - \nu_1}{2} \\ &\leq \max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\} \leq \gamma(t). \end{aligned}$$

This shows that $\ell_{i,t}$ is a valid loss sequence, i.e., an element of $\mathfrak{L}(\boldsymbol{\nu}, \boldsymbol{\gamma})$ with the limit values $\left\{\nu_i + \frac{\epsilon}{2} \cdot 1\{\nu_i - \nu_1 \leq \frac{\epsilon}{2}\}\right\}_{i\in[n]}$.

For any $j \in \{2, \dots, n\}$ define an instance $\ell^j = (\ell^j_{i,t})_{i\in[n],t\in\mathbb{N}}$ such that

$$\ell^j_{i,t} \coloneqq \begin{cases} \frac{\nu_1+\nu_i}{2}, & \text{if } t < B', \\ \nu_1, & \text{if } t \geq B' \text{ and } i = j, \\ \nu_j, & \text{if } t \geq B' \text{ and } i = 1 \text{ and } \nu_j - \nu_1 > \frac{\epsilon}{2} \\ \nu_j + \frac{\epsilon}{2}, & \text{if } t \geq B' \text{ and } i = 1 \text{ and } \nu_j - \nu_1 \leq \frac{\epsilon}{2} \\ \nu_i + \frac{\epsilon}{2}, & \text{if } t \geq B' \text{ and } i \notin \{1, j\} \text{ and } \nu_i - \nu_1 \leq \frac{\epsilon}{2} \\ \ell_{i,t}, & \text{else,} \end{cases}$$

for all $i \in [n]$ and $t \in \mathbb{N}$. By construction, we have that ℓ^j is a valid loss sequence with limit value $\boldsymbol{\nu}^j$ such that $\mathrm{OPT}_{\epsilon/2}(\boldsymbol{\nu}^j) = \{j\}$.

Since Alg solves the problem to find an $(\frac{\epsilon}{2})$-optimal arm, it satisfies $\mathrm{Alg}(\ell^1) = 1 \neq i_1 \coloneqq 2 = \mathrm{Alg}(\ell^2)$. Regarding that ℓ^1 and ℓ^2 coincide on $\{t < B'\}$ and on all $k \in [n]\backslash\{1, 2\}$ in the sense of equation 3 and equation 4, Lemma D.1 (ii) assures the existence of some $i \in \{1, 2\}$ such that $B_i(\mathrm{Alg}, \ell^1) \geq \min\{B_i(\mathrm{Alg}, \ell^1), B_i(\mathrm{Alg}, \ell^2)\} \geq B'$. Let $F_1 \coloneqq [n]\backslash\{1, i_1\}$ and fix an arbitrary $i_2 \in F_1$. Then, $\mathrm{Alg}(\ell^1) = 1 \neq i_2 = \mathrm{Alg}(\ell^{i_2})$ and since ℓ^1 and ℓ^{i_2} coincide on $\{t < B'\}$, Lemma D.1 (ii) yields the existence of some $i \in \{1, i_2\}$ such that $B_i(\mathrm{Alg}, \ell^1) \geq \min\{B_i(\mathrm{Alg}, \ell^1), B_i(\mathrm{Alg}, \ell^{i_2})\} \geq B'$. From $i_2 \in F_1 = [n]\backslash\{1, i_1\}$ we infer $i_1 \neq i_2$. With this, we define $F_2 \coloneqq F_1\backslash\{i_2\} = [n]\backslash\{1, i_1, i_2\}$.

Inductively, whenever $F_l \neq \varnothing$, we may select an element $i_{l+1} \in F_l$ and infer from Lemma D.1 (ii), due to $\mathrm{Alg}(\ell^1) = 1 \neq i_{l+1} = \mathrm{Alg}(\ell^{i_{l+1}})$ and the similarity of ℓ^1 and $\ell^{i_{l+1}}$ on $\{t < B'\}$ the existence of an element $i \in \{1, i_{l+1}\}$ such that $B_i(\mathrm{Alg}, \ell^1) \geq B'$, and define $F_{l+1} \coloneqq F_l\backslash\{i_{l+1}\}$. Then, $i_{l+1} \in F_l = [n]\backslash\{1, i_1, i_2, \dots, i_l\}$ assures $i_{l+1} \notin \{1, i_1, i_2, \dots, i_l\}$. This procedure terminates at the smallest l' such that $F_{l'} = \varnothing$, and $i_1, \dots, i_{l'}$ are distinct. Regarding that $|F_{l+1}| - |F_l| = 1$ for all $l \in \{1, \dots, l' - 1\}$, we have $l' = n$. Consequently,

$$B(\mathrm{Alg}, \ell^1) \geq \sum_{l=1}^{l'} B_{i_l}(\mathrm{Alg}, \ell^1) \geq n \cdot B'$$

holds, which shows the claim for deterministic algorithms with regard to the definition of B'.

Part 2: The statement holds for arbitrary Alg.
Let $\mathfrak{A}$ be the set of all deterministic algorithms[2] and ℓ^1 be the instance from the first part. Write δ_{ℓ^1} for the probability distribution on $\{\ell^1\}$, which assigns ℓ^1 probability one, i.e., the Dirac measure on ℓ^1. Note that for any randomized algorithm Alg there exists a probability distribution P on $\mathfrak{A}$ such that $\mathrm{Alg} \sim P$. By applying Yao's minimax principle [Yao, 1977] and using part one we conclude

$$\begin{aligned} \mathbb{E}[B(\mathrm{Alg}, \ell)] = \mathbb{E}_{\mathrm{Alg}'\sim P}[B(\mathrm{Alg}', \ell)] &\geq \inf_{\mathrm{Alg}\in\mathfrak{A}} \mathbb{E}_{\ell'\sim\delta_{\ell^1}}[B(\mathrm{Alg}, \ell')] \\ &= \inf_{\mathrm{Alg}\in\mathfrak{A}} B(\mathrm{Alg}, \ell^1) \geq n \cdot B', \end{aligned}$$

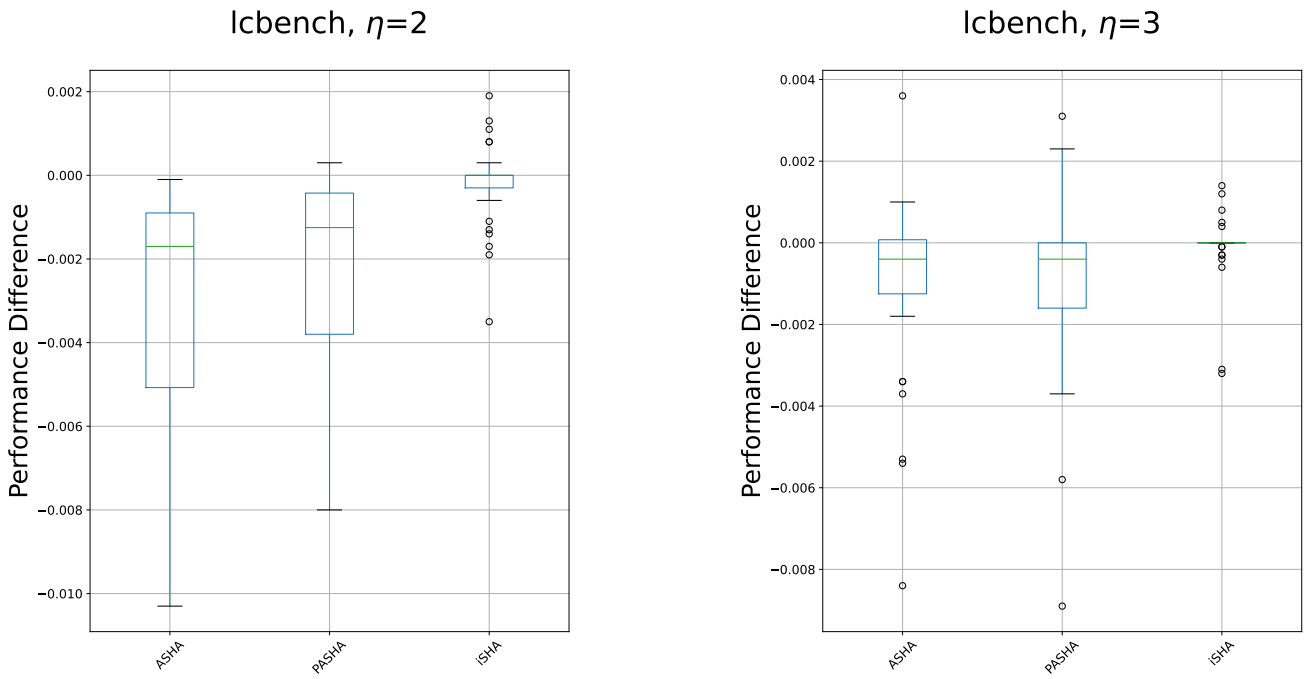
where B' is as in part one. □

[2] At any time $t \in \mathbb{N}$, a deterministic algorithm $\mathrm{Alg} \in \mathfrak{A}$ may either make a query $i \in \{1, \dots, n\}$ or terminate with a decision $\mathrm{X} \in \{1, \dots, n\}$. Thus, $\mathfrak{A}$ is a countable set.

E Detailed Empirical Results

In this section we provide the results of the empirical study in more detail, providing individual figures for every benchmark set and value for η.

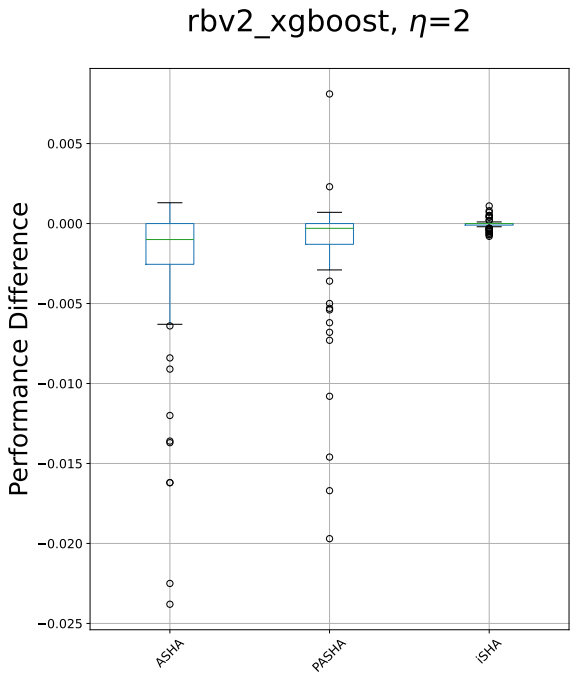
E.1 Results for lcbench

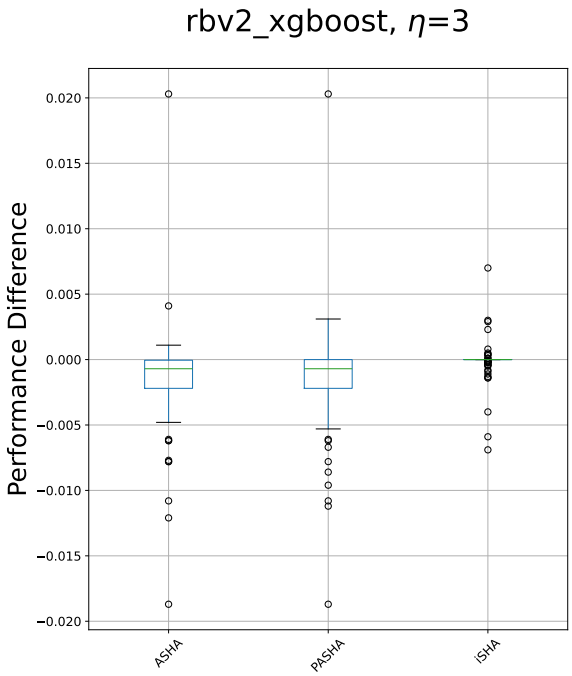


η	ASHA	PASHA	iSHA
2	0.3235	0.4412	0.7353
3	0.6765	0.5588	0.8824

Table 5: Consistency between ASHA, PASHA and iSHA with respect to SHA on lcbench.

E.2 Results for rbv2_xgboost benchmark

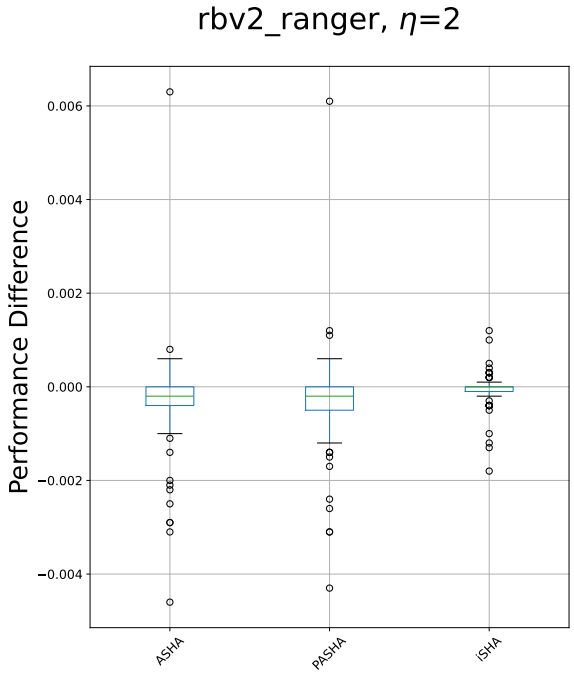


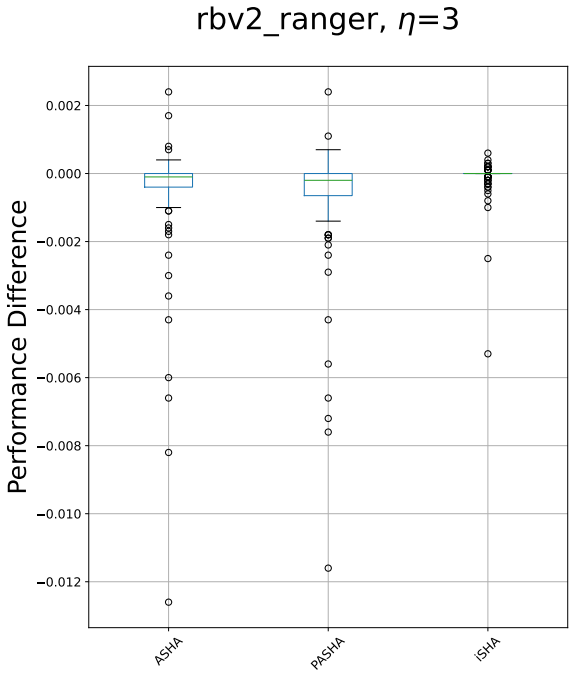


η	ASHA	PASHA	iSHA
2	0.5126	0.6975	0.9916
3	0.5126	0.5462	0.9076

Table 6: Concistency between ASHA, PASHA and iSHA with respect to SHA on rbv2_xgboost.

E.3 Results for rbv2_ranger benchmark

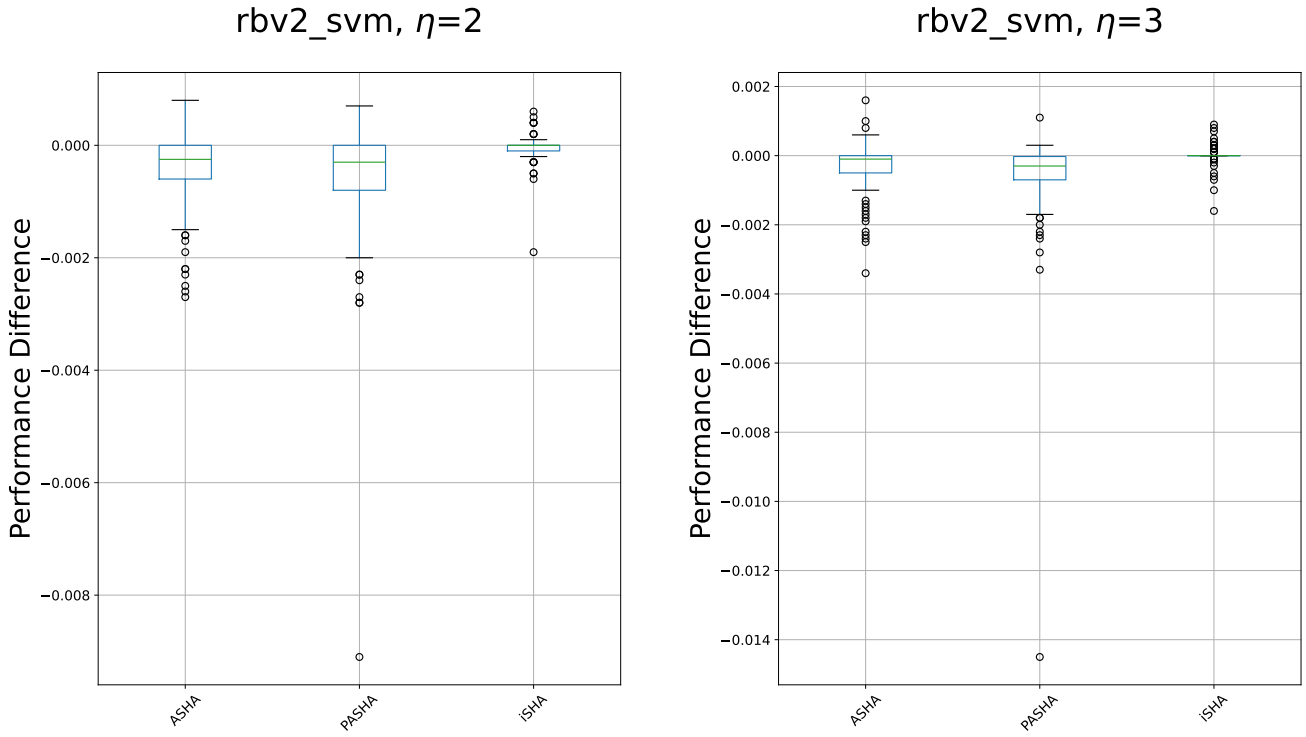




η	ASHA	PASHA	iSHA
2	0.9076	0.8655	0.9664
3	0.8655	0.8067	0.9832

Table 7: Concistency between ASHA, PASHA and iSHA with respect to SHA on rbv2_ranger.

E.4 Results for rbv2_svm benchmark

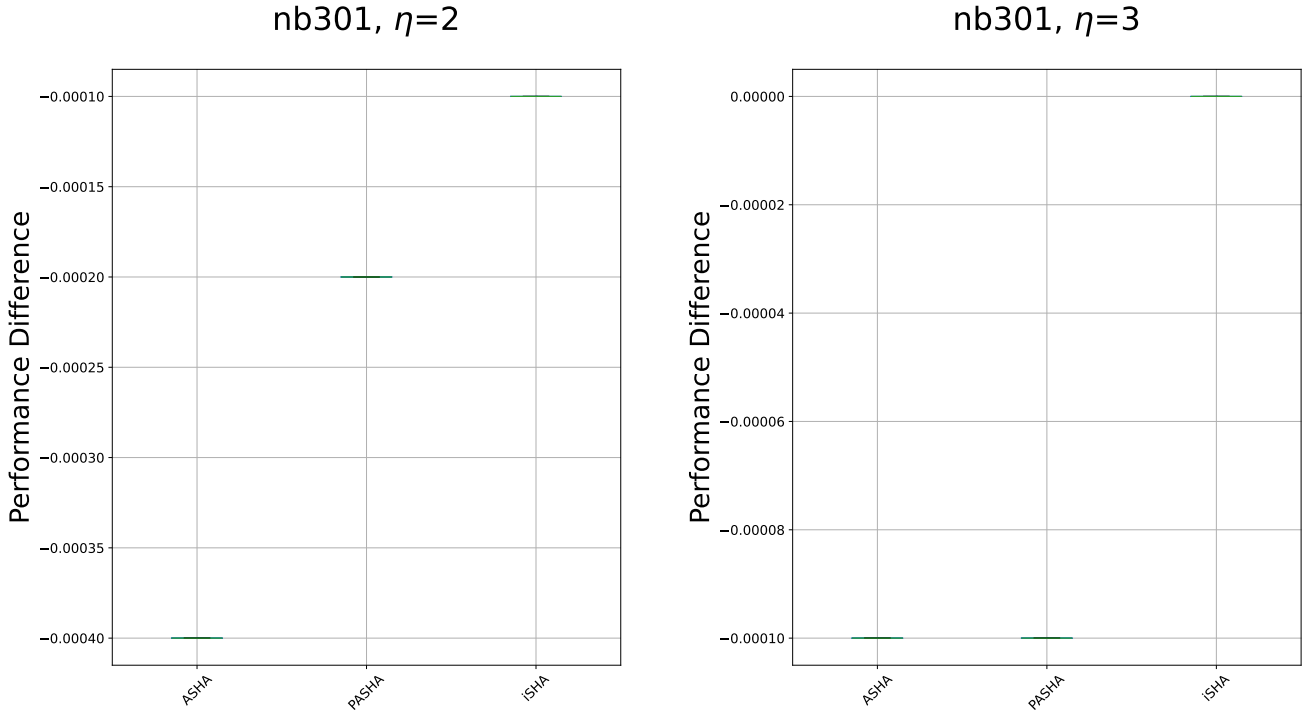


η	ASHA	PASHA	iSHA
2	0.8396	0.8396	0.9906
3	0.8774	0.8302	0.9906

Table 8: Concistency between ASHA, PASHA and iSHA with respect to SHA on rbv2_svm.

E.5 Results for nb301 benchmark

Remark: nb301 only comprises a single instance, i.e., CIFAR-10. Therefore, the boxplots are flat as only results on a single benchmark instance are reported.



η	ASHA	PASHA	iSHA
2	1.0	1.0	1.0
3	1.0	1.0	1.0

Table 9: Concistency between ASHA, PASHA and iSHA with respect to SHA on nb301.

List of Figures

List of Symbols

AC	Algorithm Configuration
$\mathcal{A}$	target alorithm in AC problem
$\mathcal{I}$	problem instance space
$\mathcal{P}$	probability distribution over $\mathcal{I}$
$\Theta = \Theta_1 \times \cdots \times \Theta_n$	parameter configuration space of n parameters
$c : \mathcal{I} \times \Theta \to \mathbb{R}$	cost function for running $\mathcal{A}$ with a specific problem instance and parameter configuration
θ^*	best parameter configuration over distribution of instances $\theta^* \in \text{argmin}_{\theta \in \Theta} \int_{\mathcal{I}} c(i, \theta) d\mathcal{P}(i)$
OPT	minimal expected cost over all configurations OPT $= \inf_{\theta \in \Theta} \mathbb{E}_{i \sim \mathcal{P}}[c(i, \theta)]$
κ	capping timeout > 0
δ	failure probability
ϵ	suboptimality
PAC	probably approximately correct
OPT_γ	minimal expected cost excluding the best γ-fraction $\text{OPT}_\gamma = \inf_{x \in \mathbb{R}}\{x \mid \mathbb{P}_{\theta \sim \text{Unif}(\Theta)}(\mathbb{E}_{i \sim \mathcal{P}}[c(i, \theta)] \leq x) \geq \gamma\}$
$\mathcal{I}_{\text{train}}$	training set of instances $\subseteq \mathcal{I}$
HPO	Hyperparameter Optimization
HPC	Hyperparameter Configuration
$l : \Theta \to \mathbb{R}$	loss function for running $\mathcal{A}$ with a specific HPC
MAB	Multi-Armed Bandits
n	number of considered configurations / arms
$[n]$	set of arm indices $[n] := \{1, \ldots, n\}$
$\mathfrak{A}$	action set
T	time horizon $\in \mathbb{N} \cup \{\infty\}$
$r_{i,t}$	reward of arm $i \in [n]$ at time $t \in [T]$
$\mathcal{D}_i$	reward distribution for arm $i \in [n]$
μ_i	mean reward for arm $i \in [n]$
i^*	best arm, e.g. for stochastic MABs $i^* \in \text{argmax}_{i \in \mathfrak{A}} \mu_i$
μ^*	maximal reward $\mu^* = \max_{i \in \mathfrak{A}} \mu_i$
R_T	cumulative regret up to time T: $R_T = \max_{i \in \mathfrak{A}} \sum_{t=1}^{T} r_{i,t} - \sum_{t=1}^{T} r_{i,t}$
Δ_i	suboptimality gabs $\Delta_i = \mu^* - \mu_i$
$p_{i,j}$	winning probability of arm $i \in [n]$ over arm $j \in [n]$, $pi, j \in [0, 1]$
$\Delta_{i,j}$	calibrated pairwise probabilities $\Delta_{i,j} = p_{i,j} - \frac{1}{2}$

i^*_{Cope}	Copeland Winner $i^*_{Cope} \in \{i \in [n] \mid d_i \geq d_j \ \forall j \in [n]\}$, where $d_i = \lvert\{j \in [n] \mid p_{i,j} > 0.5\}\rvert$
i^*_{Borda}	Borda Winner $i^*_{Borda} \in \text{argmax}_{i \in [n]} \frac{1}{n-1} \sum_{j \in [n], j \neq i} p_{i,j}$
R_T^{strong}	strong regret $R_T^{strong} = \sum_{t=1}^{T} \max\{\Delta_{i^*,i_t}, \Delta_{i^*,j_t}\}$
R_T^{avg}	average regret $R_T^{avg} = \sum_{t=1}^{T} \frac{1}{2} (\Delta_{i^*,i_t} + \Delta_{i^*,j_t})$
R_T^{weak}	weak regret $R_T^{weak} = \sum_{t=1}^{T} \min\{\Delta_{i^*,i_t}, \Delta_{i^*,j_t}\}$
S_t	superarm at time step $t \in [T]$: $S_t = \{i_{t,1}, \ldots, i_{t,K}\} \subseteq [n]$
K	subset size $\in \mathbb{N}$, $K \leq n$
ν_i	limit value to which averaged rewards converge $\lim_{t \to \infty} \frac{1}{t} \sum_{s=1}^{t} r_{i,s} \to \nu_i \in \mathbb{R} \ \ \forall i \in \mathfrak{A}$
$\gamma : \mathbb{N} \to \mathbb{R}$	convergence speed $\left\lvert \frac{1}{t} \sum_{s=1}^{t} r_{i,s} - \nu_i \right\rvert \leq \gamma(t) \ \ \forall i \in \mathfrak{A}$

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I hereby declare that this dissertation is my original work, composed independently, and without the use of any unauthorized materials and additional, non-indicated help. All sources and references utilized are properly acknowledged and cited. This dissertation has not been previously submitted to any other faculty or institution. Furthermore, I confirm that I have not undergone an unsuccessful doctoral examination, not have I been stripped of any previously earned doctoral degrees.

Paderborn, January 13, 2025

Jasmin Brandt