
Article:
Hall, G.T., Oliveto, P.S. and Sudholt, D. (2022) On the impact of the performance metric on efficient algorithm configuration. Artificial Intelligence, 303. 103629. ISSN: 0004-3702

https://doi.org/10.1016/j.artint.2021.103629

On the Impact of the Performance Metric on Efficient Algorithm Configuration

George T. Hall[a,b], Pietro S. Oliveto[a], Dirk Sudholt[a,c]

[a] *Department of Computer Science, University of Sheffield, United Kingdom*
[b] *Genetics and Genomic Medicine, UCL Great Ormond Street Institute of Child Health, University College London, United Kingdom*
[c] *Chair of Algorithms for Intelligent Systems, University of Passau, Passau, Germany*

Abstract

Algorithm configurators are automated methods to optimise the parameters of an algorithm for a class of problems. We analyse the impact of the cutoff time κ (the time spent evaluating a configuration for a problem instance) on the expected number of configuration comparisons required to find the optimal parameter value for the performance metrics (the measure used to judge the performance of a configuration) that compare configurations using either the best-found fitness values or optimisation times. We first prove that the configurators that use optimisation time as performance metric are not able to tune any unary unbiased algorithm for any function with up to an exponential number of optima using $\kappa \leq (n \ln n)/2$. Afterwards, we show that for simple algorithm configuration scenarios the required cutoff time for the optimisation time metric may be considerably larger while using the best fitness metric allows the tuners to configure the target algorithm in linear time in the number of parameters.

Keywords: Algorithm configurators, parameter tuning, runtime analysis, performance metrics, cutoff time

1. Introduction

General purpose heuristics, such as evolutionary algorithms, have the advantage that they can generate high quality solutions to optimisation problems without requiring much knowledge about the problem at hand. All that is required to apply a general purpose heuristic is a suitable representation for candidate solutions and a measure (the fitness function) that enables the comparison of the quality of different solutions against each other. However, it is well understood that different design choices and settings of their numerous parameters (e.g., mutation rate, crossover rate, selective pressure and population

Email addresses: george.hall@ucl.ac.uk (George T. Hall), p.oliveto@sheffield.ac.uk (Pietro S. Oliveto), dirk.sudholt@uni-passau.de (Dirk Sudholt)

size for generational genetic algorithms (GAs)) may considerably affect their performance and in turn the quality of the identified solutions. In particular, the capability of heuristics to identify high quality solutions in a short time depends crucially on the use of suitable parameter settings [1].

Traditionally, the design and parameter tuning of the algorithm for the problem at hand has mainly been done *manually*. Typically, the developer chooses some algorithmic designs and values for the associated parameters and executes them on instances of the problem. Refinements to the previous choices are then made according to how well each algorithm/parameter configuration has performed. However, such a procedure (or a similar one) is a time-consuming and error-prone process. From a scientific research point of view, it is also biased by personal experience hence difficult to reproduce. Consequently it has become increasingly common to use automated and principled methodologies for algorithm development [2, 3, 4, 5, 6].

Many automated algorithm configurators have gained widespread usage since they have often identified better parameter values compared to carefully chosen default configurations [5, 7, 8, 9, 10]. While varying in several algorithmic details, all algorithm configurators generally aim to evolve increasingly good parameter values by evaluating the performance of candidate configurations on a training set of instances and using some perturbation mechanism (e.g., iterated local search in ParamILS [5] or updating the sampling distributions in irace [7, 11]) to generate new ones based on the better-performing ones in the previous generation. The overall aim is that the ultimately identified parameter values perform well (*generalise*) on unseen instances of the tackled problem.

Despite their popularity, there is a lack of theoretical understanding of such configurators. For instance, it is unclear how good the identified parameter values are compared to optimal ones and how long it takes for these to be identified. Furthermore, there are no theoretical indications on how to set the algorithm configurator's own parameters such as the cutoff time (how long each algorithm configuration should be run) or which performance metric to use to estimate the performance of different configurations.

One exception is a recent worst-case analysis of using optimisation time as performance metric [12]. It was proven that all algorithm configurators using this performance metric and a cutoff time that is either static or bounded by the runtime of a previously-run configuration have poor performance in the worst case. On the other hand, the authors design a worst-case-tailored algorithm called Structured Procrastination (SP) that adaptively increases the cutoff time if it is too small to allow the configuration to reach the optimum and which provably performs better in the worst case. However, worst-case analyses do not explain the success that popular algorithm configurators have in practice. Indeed, experimental evidence has recently been provided that many algorithm configuration landscapes exhibit unimodal (even convex) characteristics, and hence are more benign for gradient-based algorithm configurators than the worst-case scenario [13].

In this paper, we take a first step towards the time complexity analyses of specific configurators for different scenarios to explain their behaviour and

performance. Our aim is to highlight the impact of the chosen performance metrics both in cases when the parameter values that minimise the optimisation time are sought and when those that maximise solution quality within some time budget are preferred. In particular, we argue that the most natural performance metric for a given application is not necessarily the optimal choice. For instance, it may be natural to assume that if the set of parameter values that minimises the optimisation time of the target algorithm for a problem class is sought, then using the optimisation time itself as performance metric is preferable. Our analyses reveal that this is not necessarily the case even in simple algorithm configuration scenarios: using the best-identified fitness within a cutoff time as performance measure allows algorithm configurators to identify more quickly the parameter values that minimise the target algorithm's optimisation time in the considered scenarios.

More precisely, we prove that any algorithm configurator that uses optimisation time as performance metric requires a cutoff time of $\Omega(n \log n)$ with overwhelming probability (w. o. p.)[1] to tune any unary (i.e., using only mutation operators) unbiased target algorithm with a single parameter for any target function containing up to an exponential number of optima in the problem size n. For smaller cutoff times such configurators behave as if all parameter values have the same performance. Afterwards, for specific, yet simple, scenarios we show that the required cutoff time may be considerably larger, while using the fitness-based performance metric allows the tuner to identify the parameter values that minimise the optimisation time in linear time in the number of parameter values.

For our puproses we will consider a simple stochastic hillclimbing tuner, called ParamRLS as well as the popular and established ParamILS configurator which uses iterated local search. We will analyse the number of iterations required by algorithm configurators to identify the optimal parameter values w. o. p. for the randomised local search (RLS_k) algorithm, where k, the only parameter, is the local search neighbourhood size (i.e., k bits are flipped without replacement in each iteration) for two well-known black-box benchmark function classes: Ridge [14] and OneMax [15]. The choice of performance metric affects which configuration is the target that the configurator seeks. An *F-optimal* configuration achieves the highest expected solution quality within the cutoff time and is therefore the target when using the fitness-based performance metric, whereas a *T-optimal* configuration has the smallest expected optimisation time and therefore is the target when using optimisation time as performance metric.

The two target function classes have considerably different characteristics. For Ridge, each parameter value has the same improvement probability independent of the position of the candidate solution in the search space. For OneMax, however, it is better to flip fewer bits the closer the candidate so-

[1] We say that a probability is overwhelming if it is at least $1 - 2^{-\Omega(n^\varepsilon)}$ for some constant $\varepsilon > 0$. We frequently use that by a union bound, any polynomial number of events that all occur w. o. p. occur together w. o. p. as $n^{O(1)} \cdot 2^{-\Omega(n^\varepsilon)} = 2^{-\Omega(n^\varepsilon)+O(\log n)} = 2^{-\Omega(n^\varepsilon)}$.

lution is to the optimum. This implies that for RIDGE the optimal parameter value (i.e., $k = 1$) is independent of the chosen performance measure: $k = 1$ will have better performance independent of for how long the algorithm is run i.e., even for very small cutoff times as long as sufficiently many runs are performed in each configuration comparison. Roughly speaking, for ONEMAX, larger values of k find better solutions in short runs of RLS_k, whereas smaller values of k perform better in longer runs. Hence, for this problem class, which parameter value k is optimal depends on the chosen performance measure.

Our results are summarised in Table 1, where we use 'Opt-Time' to refer to all configurators that use optimisation time as performance metric and 'ParamRLS-F' and 'ParamRLS-T' to indicate respectively whether ParamRLS uses the best-identified fitness or the optimisation time as performance metric. ParamILS uses its traditional optimisation time metric with *penalised average runtime* (PAR).

Our analysis shows that ParamRLS-F can efficiently identify that $k = 1$ is the optimal parameter value for RIDGE independent of the cutoff time as long as the performance of each parameter configuration is evaluated a sufficient number of times (recall that $k = 1$ is both F-optimal and T-optimal). On the other hand, configurators that use optimisation time as performance metric require at least quadratic cutoff times in the problem size. This implies that ParamRLS-F can identify the T-optimal configuration with any cutoff time, whereas configurators using a performance metric explicitly designed for this task cannot. For ONEMAX, ParamRLS-F identifies that $k = 1$ is the optimal parameter value for any cutoff time greater than $0.975n$ for which it is both F-optimal and T-optimal while, for cutoff times in the range $[0.02n, 0.72n]$ it will identify that $k = 5$ is an F-optimal configuration. In contrast, configurators using optimisation time as performance metric require at least superlinear cutoff times to identify the T-optimal parameter value for ONEMAX or they will return one chosen uniformly at random. Therefore, ParamRLS-F is once again able to identify the T-optimal configuration using cutoff times that are asymptotically smaller than those required by configurators that use optimisation time as performance metric.

Compared to the extended abstract of this paper [16], this version includes all previously omitted proofs and a tightened analysis of the configuration of RLS_k for ONEMAX. Furthermore, the mutation operator of ParamRLS has been generalised to allow for arbitrarily large step sizes, and the results concerning ParamRLS-T have been extended to also hold for all configurators that use optimisation time as performance metric. Finally, we include positive and negative results for the popular ParamILS configurator exactly as it is used in practice. This is remarkable for a first runtime analysis paper on algorithm configurators since, concerning function optimisation, for many years simplified evolutionary algorithms had to be analysed before it became possible to perform runtime analyses of standard realistic ones [17, 18, 19, 20, 21, 22, 23].

This paper is split into four main sections. In Section 2 we provide formal descriptions of the algorithm configuration problem, the algorithm configurators, the target algorithms and the optimisation function classes considered in

Configurator	Cutoff Time (κ)	Runs per Evaluation (r)	Result
Configuring RLS_k if all training instances have $\leq \exp(\sqrt{n}/\log^2 n)$ optima			
All Opt-Time	$\kappa \leq (n \ln n)/2$	$\forall r \in \mathrm{poly}(n)$	blind
Any algorithm with single parameter θ			
ParamILS	large enough to optimise all training instances w. o. p.	$\forall r \in \mathrm{poly}(n)$	optimal θ
Configuring RLS_k for RIDGE*			
ParamRLS-F	$\kappa = 1$	$\forall r \geq n^{3/2}$	optimal k ($k = 1$)
ParamRLS-F	$\kappa = \omega(n)$	$\forall r \geq 1$	optimal k ($k = 1$)
All Opt-Time	$\kappa \leq (1-\varepsilon)n^2$	$\forall r \in \mathrm{poly}(n)$	blind
ParamRLS-T	$\kappa \geq (1+\varepsilon)n^2$	$\forall r \geq 1$	optimal k ($k = 1$)
ParamILS	$\kappa \geq (1+\varepsilon)n^2$	$\forall r \geq 1$	optimal k ($k = 1$)
Configuring RLS_k for ONEMAX*			
ParamRLS-F	$0.02n \leq \kappa \leq 0.72n$	$\forall r \geq 1$	optimal k ($k = 5$)
ParamRLS-F	$\kappa \geq 0.975n$	$\forall r \geq 1$	optimal k ($k = 1$)
All Opt-Time	$\kappa \leq (n \ln n)/2$	$\forall r \in \mathrm{poly}(n)$	blind
ParamRLS-T	$\kappa \geq n^{1+\varepsilon}$	$\forall r \geq 1$	optimal k ($k = 1$)
ParamILS	$\kappa \geq n^{1+\varepsilon}$	$\forall r \geq 1$	optimal k ($k = 1$)

Table 1: A summary of our results. ParamRLS-F is able to identify the configuration that attains the highest solution quality within the cutoff time, whereas configurators using the optimisation time as performance metric are blind (i.e. behave as though all configurations have the same performance) for cutoff times considerably smaller than the expected optimisation time. RIDGE* and ONEMAX* (defined in Section 2.4) are modifications of RIDGE and ONEMAX that allow the optimum to be reached by any configuration of RLS_k.

this paper as well as the mathematical tools used in the analyses. In Section 3 we provide general upper and lower bounds on cutoff time required for configurators that use the optimisation time metric to be efficient. In Sections 4 and 5 we respectively present our analysis of ParamRLS and ParamILS configuring RLS_k for RIDGE and ONEMAX.

2. Preliminaries

2.1. The Algorithm Configuration Problem

In this section we briefly outline the Algorithm Configuration Problem, following the definition given in [2]. Informally, given an algorithm $\mathcal{A}$, its set of parameters $\{p_1, \ldots, p_{N_P}\}$ and an optimisation problem Π with instance distribution $\mathcal{I}$, the *algorithm configuration problem* is that of identifying a parameter configuration (i.e. a value for each parameter) that optimises the performance of $\mathcal{A}$ for Π with instances distributed following $\mathcal{I}$. We call the algorithm solving the configuration problem the *configurator* (or *tuner*) and the algorithm to be tuned ($\mathcal{A}$) the *target algorithm*.

More formally, we say that, if each parameter p_i has a set of feasible values Θ_i, then $\mathbf{\Theta} \subseteq \Theta_1 \times \ldots \times \Theta_{N_P}$ is the *parameter space* of $\mathcal{A}$ (i.e., the search space of all feasible parameter configurations). We denote a specific configuration by $\theta \in \mathbf{\Theta}$ and the instantiation of $\mathcal{A}$ using the parameter values in configuration θ as $\mathcal{A}(\theta)$. Finally, let $cost_{\mathcal{I}}(\theta)$ be a measure of the performance of running $\mathcal{A}(\theta)$ on instances distributed according to $\mathcal{I}$. Then the algorithm configuration problem is that of finding

$$\theta^* \in \operatorname*{argmin}_{\theta \in \mathbf{\Theta}} cost_{\mathcal{I}}(\theta)$$

In practice, the value of $cost_{\mathcal{I}}(\theta)$ is often estimated using a training set of problem instances $\Pi' \subseteq \Pi$. The training set should be chosen such that configurations that perform well on it also perform well on the problem class itself.

In order to evaluate the performance of configurations, the following decisions need to be made:

- The definition of the training set of problem instances $\Pi' \subseteq \Pi$.
- The *cutoff time* κ (i.e. the time for which a configuration is executed in a single run in an evaluation).
- The number of *runs* r per evaluation (i.e. the number of times an instance is drawn using the distribution $\mathcal{I}$ and the configuration run on it).
- The *performance metric* (i.e. which quality measure to use to assess the performance of different configurations for an instance, e.g. "the best-found solution within a time budget" or "the time required to reach a solution of at least a given quality"). It is important to note that different performance metrics may yield different optimal configurations.
- The method used to *aggregate* performance measures over multiple runs of a configuration.

Since for the two problem classes considered in this paper (see Section 2.4) one random instance suffices to identify the optimal configuration, we need not worry about the choice of the training set. In this work, results showing that configurators will be blind hold for any aggregation function. For positive results, the performance of configurations is aggregated over multiple runs on the same problem instance as follows. In ParamRLS-F, the configuration that wins the most runs wins the comparison; in ParamRLS-T and ParamILS, the configuration with the smallest mean penalised runtime wins the comparison. We will consider two different *performance metrics*:

1. **Optimisation time**: the time required for $\mathcal{A}(\theta)$ to find the optimal solution of an instance π_i. If the optimum is not found before the cutoff time κ, then $C \cdot \kappa$ is taken as the time to reach the optimum, where C is a penalty constant (i.e. PAR-C). This metric is commonly used in ParamILS where usually $C = 10$ [5].

2. **Best fitness**: the fitness of the best solution found within the cutoff time, paired with the time at which progress was last made (this second component of the performance metric is only used in the event of a tie, in which case we favour the configuration which made progress least recently).

Let T be the number of configuration comparisons performed by the configurator before the optimal configuration θ^* is identified. Then for ParamRLS and ParamILS the total tuning time (i.e., the number of fitness function evaluations) will be at most $\mathcal{B} = T \cdot \kappa \cdot r$. Our aim in this paper is to estimate, for each performance metric, how the cutoff time κ and the number of runs r impact T and $\mathcal{B}$ for the ParamRLS and ParamILS configurators. In particular, we will estimate the value of T, from which $\mathcal{B}$ can also be easily derived.

2.2. The Configurators: ParamRLS and ParamILS

In this section we present the tuners which we analyse in this paper. We first give details of ParamRLS which is the simplest possible stochastic local search tuner. We then present the popular ParamILS tuner.

2.2.1. ParamRLS

We design a simple configurator following the framework laid out for ParamILS [5]:

1. Initialise the configurator with some initial configuration θ;
2. Mutate θ by modifying a single parameter and accept the new configuration θ' if it results in improved performance;
3. Repeat Step 2 until no single parameter change yields an improvement.

Following the above scheme, we initialise the configurator by choosing a configuration uniformly at random from Θ and, in an alteration to the above framework, accept the new configuration if it performs at least as well as its parent. The target algorithms considered in this paper only have one parameter. We refer to the current value of θ in our tuner as the *active parameter*. Concerning Step 2, ParamILS applies an iterated local search procedure. We instead consider the more simple random local search operator $\pm\{1, \ldots, \ell\}$, and, thus call our algorithm ParamRLS. Under the local search operator $\pm\{1, \ldots, \ell\}$, the active parameter value is increased or decreased by any integer between 1 and ℓ inclusive, where both this integer and whether to increase or decrease are decided uniformly at random each time the value of the active parameter is mutated. The operator $\pm\{1\}$ has previously been analysed for the optimisation of functions defined over search spaces with larger alphabets than those that can be represented using bit strings [24]. We assume that any mutation that oversteps a boundary is considered infeasible by ParamRLS and would automatically lose the comparison against a feasible parameter value[2].

[2]We make this assumption to simplify the analysis. In practice, infeasible configurations could clearly be rejected before being run in a comparison. This assumption is thus pessimistic.

The resulting configurator is described in Algorithm 1. The termination condition may be either a predetermined number of iterations without a change in configuration (i.e., the solution is likely a local or global optimum) or a fixed number of iterations. In this paper, we calculate both (1) the expected number of comparisons until the configurator first sets the active parameter to the optimal parameter configuration and (2) the configurator settings required such that it returns the optimal value w. o. p. We thus provide bounds on the necessary termination criterion for the configurator.

If the configurator uses the fitness-based performance metric for performance comparison described in the previous section, then we call the algorithm ParamRLS-F while if it uses the time-based metric, then we refer to it as ParamRLS-T. The two configurators are described in Algorithm 2 and Algorithm 3, respectively. In Algorithm 3, we denote the capped optimisation time for $\mathcal{A}(\theta)$ on instance π using cutoff time κ and penalty constant C as $\texttt{CapOptTime}(\mathcal{A}(\theta'), \kappa, \pi, C)$. Since the behaviour of a parameter configuration is identical on all instances of the problem classes considered in this paper, for simplicity we only use one instance for each comparison. Using larger training sets would not affect our results (i.e. the number of required comparisons to identify the optimal configuration would remain the same). In practice, it may be desirable to cache the performances of configurations. However, in order to simplify the analysis, we assume throughout this paper that no caching takes place. Re-evaluating the performance of configurations may also be useful since it allows the configurator to overcome local optima introduced by the noisy evaluation of configurations.

Algorithm 1: ParamRLS $(\mathcal{A}, \Theta, \Pi', \kappa, r, \ell)$

Input: target algorithm $(\mathcal{A})$, parameter space (Θ), training instances (Π'), cutoff time (κ), number of runs per evaluation (r), step size (ℓ).

```
1 θ ←initial configuration chosen uniformly at random
2 while termination condition not satisfied do
3 |   θ' ← ±{1,...,ℓ}(θ)
4 |   θ ← better(A, θ, θ', Π', κ, r) // Comparison between θ and θ'
5 return θ
```

2.2.2. ParamILS

We now give an outline of the popular ParamILS tuner [5]. The main difference with ParamRLS is that iterated local search is performed by ParamILS, instead of simple random local search i.e., just comparing the current configuration against a random neighbour in each iteration (the neighbourhood of a configuration θ is defined as the set of configurations that differ from θ by exactly one parameter value).

We reproduce the pseudocode given in [5] as Algorithm 4. Essentially, after

Algorithm 2: ParamRLS-F $(\mathcal{A}, \Theta, \Pi, \kappa, r, \ell)$

Input: target algorithm ($\mathcal{A}$), parameter space (Θ), training instances (Π), cutoff time (κ), number of runs per evaluation (r), step size (ℓ).

$\theta \leftarrow$initial configuration chosen uniformly at random
while *termination condition not satisfied* **do**
  $\theta' \leftarrow \pm\{1, \ldots, \ell\}(\theta)$
  repeat r **times**
    $\pi \leftarrow$ problem instance drawn from Π according to $\mathcal{I}$
    Fit $\leftarrow \mathcal{A}(\theta)$ fitness after κ iterations on π
    Fit$'$ $\leftarrow \mathcal{A}(\theta')$ fitness after κ iterations on π
    ImprTime $\leftarrow$time of last improvement of $\mathcal{A}(\theta)$ on π
    ImprTime$'$ $\leftarrow$time of last improvement of $\mathcal{A}(\theta')$ on π
    if Fit $>$ Fit$'$ **then** W $\leftarrow$ W $+ 1$
    else if Fit$'$ $>$ Fit **then** W$'$ $\leftarrow$ W$'$ $+ 1$
    else
      if ImprTime $<$ ImprTime$'$ **then** W $\leftarrow$ W $+ 1$
      else if ImprTime$'$ $<$ ImprTime **then** W$'$ $\leftarrow$ W$'$ $+ 1$
  if W$'$ $>$ W **then** $\theta \leftarrow \theta'$
  else if W $==$ W$'$ **then with probability** 0.5 **do** $\theta \leftarrow \theta'$
return θ

Algorithm 3: ParamRLS-T $(\mathcal{A}, \Theta, \Pi', \kappa, r, \ell)$

Input: target algorithm ($\mathcal{A}$), parameter space (Θ), training instances (Π'), cutoff time (κ), number of runs per evaluation (r), penalisation constant (C), step size (ℓ).

$\theta \leftarrow$initial configuration chosen uniformly at random
while *termination condition not satisfied* **do**
  $\theta' \leftarrow \pm\{1, \ldots, \ell\}(\theta)$
  OptTimes $\leftarrow 0$ `// runtime counter for` $\mathcal{A}(\theta)$
  OptTimes$'$ $\leftarrow 0$ `// runtime counter for` $\mathcal{A}(\theta')$
  repeat r **times**
    $\pi \leftarrow$ problem instance drawn from Π' according to $\mathcal{I}$
    OptTimes $\leftarrow$ OptTimes $+$ `CapOptTime`$(\mathcal{A}(\theta), \kappa, \pi, C)$
    OptTimes$'$ $\leftarrow$ OptTimes$'$ $+$ `CapOptTime`$(\mathcal{A}(\theta'), \kappa, \pi, C)$
  if OptTimes$'$ $<$ OptTimes **then** $\theta \leftarrow \theta'$
  else if OptTimes $=$ OptTimes$'$ **then**
    with probability 0.5 **do** $\theta \leftarrow \theta'$
return θ

having selected the best initial configuration out of $\rho+1$ random ones, ParamILS repeats the following main loop:

1. It applies an iterated local search procedure to the current solution called *IterativeFirstImprovement*;
2. It perturbs the identified local optimum by performing a random walk of length s through the parameter space, i.e. it performs s random moves where in each move it chooses a parameter u.a.r. and randomly selects a new value for it (*perturbation phase*);
3. It re-initialises the search procedure with probability p_{restart}.

The selection criterion used by ParamILS is to accept parameter configurations that have performed better or equally well over the training set.

The *IterativeFirstImprovement* procedure is described in Algorithm 5. Given an input configuration θ, the procedure visits its undiscovered[3] neighbours (i.e., all the neighbours which have not been tested in the current call of *IterativeFirstImprovement*) $UndiscNbh(\theta)$ in a randomised order and then accepts the first one it finds which is at least as good as θ. It then performs the same procedure on this newly-discovered configuration. This process is repeated until it finds a configuration with no undiscovered neighbours that are at least as good.

Two different approaches are commonly applied to choose which instances of the training set (and how many) should be used to compare different configurations. The basic approach, referred to as BasicILS, compares the mean runtime of each configuration to optimise a training set of N instances chosen uniformly at random. A more sophisticated approach, referred to as FocusedILS, avoids wasting configuration budget on the evaluation of suboptimal configurations. As a result not necessarily all of the training instances need to be used for the winner of a comparison to be decided. For results where we prove that ParamILS is blind, it is irrelevant whether BasicILS or FocusedILS is used since the cutoff time is simply too small to allow the optimal configuration to be identified. For results where we show that ParamILS is able to identify the optimal configuration, we assume that it uses BasicILS in order to simplify the analysis. This is a pessimistic assumption: using FocusedILS may allow fewer runs to be used. However, for our results where only a single run per evaluation is used, BasicILS and FocusedILS will behave identically. Thus, for the *better* procedure of ParamILS we simply use the comparison procedure used in ParamRLS-T as defined in Algorithm 3, with the only difference that ParamILS always accepts the newest configuration amongst two equally performing ones (i.e. when OptTimes = OptTimes$'$ in line 11 of Algorithm 3, ParamILS sets $\theta \leftarrow \theta'$ with probability 1 instead of 0.5).

[3]The consideration of *undiscovered* neighbours fixes a typo in the pseudocode given in [5] which could lead to an infinite loop if equally good configurations belong to the same neighbourhood. Our pseudocode follows the implementation of ParamILS available at `http://www.cs.ubc.ca/labs/beta/Projects/ParamILS`.

Algorithm 4: ParamILS pseudocode, recreated from [5].

Input: Initial configuration $\theta_0 \in \Theta$, parameters r, p_{restart}, and s.
Output: Best parameter configuration θ found.

for $i = 1, \ldots, \rho$ **do**
    $\theta \leftarrow$ random $\theta \in \Theta$
    if *better*(θ, θ_0) **then** $\theta_0 \leftarrow \theta$
$\theta_{\text{inc}} \leftarrow \theta_{\text{ils}} \leftarrow$ *IterativeFirstImprovement*(θ_0)
while *not TerminationCriterion()* **do**
    $\theta \leftarrow \theta_{\text{ils}}$
    for $i = 1, \ldots, s$ **do**
        `// Random perturbation step of size` s
        `// Nbh contains all neighbours of a configuration`
        $\theta \leftarrow$ random $\theta' \in Nbh(\theta)$
    $\theta \leftarrow$ *IterativeFirstImprovement*(θ)
    if *better*$(\theta, \theta_{\text{ils}})$ **then** $\theta_{\text{ils}} \leftarrow \theta$
    if *better*$(\theta_{\text{ils}}, \theta_{\text{inc}})$ **then** $\theta_{\text{inc}} \leftarrow \theta_{\text{ils}}$
    with probability p_{restart} **do** $\theta_{\text{ils}} \leftarrow$ random $\theta \in \Theta$
return θ_{inc}

2.2.3. Blind tuners

We prove several times in this work that a tuner is effectively unable to configure a target algorithm for a problem class because in the vast majority of runs, the tuner will simply return a parameter value chosen uniformly at random. We formalise this notion as follows.

Definition 1. We call a tuner *blind* if there is an event A that occurs w. o. p. and, conditional on A, the tuner returns a configuration chosen according to a distribution which would be generated if all configurations had the same performance.

Here the event A characterises a "typical" run of a tuner; it is necessary since in an "atypical" run, we may not know the output of the tuner. Our notion of blindness implies that if ParamRLS or ParamILS are blind then their output is virtually indistinguishable from a uniform random distribution of parameter values for any polynomial time period.

2.3. The Target Algorithm: RLS$_{\boldsymbol{k}}$

In this paper we will analyse the performance of the presented algorithm configurators for tuning the RLS$_k$ target algorithm which has only one parameter, k. RLS$_k$ differs from conventional RLS in that the latter flips exactly one bit per iteration whereas RLS$_k$ flips exactly k distinct bits per iteration, chosen uniformly at random. Our aim is to identify the time required by the tuners to identify the best value for the parameter k. We provide the pseudocode for RLS$_k$ in Algorithm 6. We define the permitted values for k as the set $\{1, \ldots, \phi\}$.

Algorithm 5: IterativeFirstImprovement(θ) procedure, recreated from [5].

```
repeat
    θ' ← θ
    forall θ'' ∈ UndiscNbh(θ') in randomised order do
        // UndiscNbh contains all undiscovered neighbours of
           a configuration
        if better(θ'', θ') then θ ← θ''; break
until θ' = θ
return θ
```

Algorithm 6: RLS_k for the maximisation of a function f

```
initialise x // according to initialisation scheme
while termination criterion not met do
    x' ← x with k distinct bits flipped, chosen uniformly at random
    if f(x') ≥ f(x) then x ← x'
```

2.4. The Function Classes RIDGE* *and* ONEMAX*

Apart from providing some general results in the following section, in the rest of the paper we will analyse the performance of the configurators for tuning RLS_k for two optimisation problems with considerably different characteristics. One where the performance of each parameter configuration does not change throughout the search space and another where, according to the cutoff times, different configurations will perform better.

We first consider the standard RIDGE benchmark problem [14]. This function consists of a gradient of increasing fitness with the increase of the number of 0-bits in the bit string that leads towards the 0^n bit string (i.e., ZEROMAX). From there a path of n points consisting of consecutive 1-bits followed by consecutive 0-bits, called the *ridge*, leads to the global optimum (i.e. the 1^n bit string). RIDGE is defined as:

$$\text{RIDGE}(x) = \begin{cases} n + |x|_{\text{ONES}}, & \text{if } x \text{ is in the form } 1^i 0^{n-i} \\ n - |x|_{\text{ONES}}, & \text{otherwise} \end{cases}$$

where $|x|_{\text{ONES}}$ is the number of 1-bits in the bit string. Note that the approach to the ridge is very easy for common optimisers, whereas following the ridge is significantly more challenging. Since our focus is on tuning algorithms for optimising the ridge, we follow the approach from [25] and assume in the following that RLS_k is initialised at the start of the ridge, i.e., in 0^n. Since RLS_k always flips exactly k bits, it will not always be possible to reach the optimum (i.e. 1^n). The optimal value of RIDGE which we are able to reach when using RLS_k, starting from 0^n, is in fact $\lfloor \frac{n}{k} \rfloor k$. In order to avoid an infinite

expected optimisation time (i.e. where the expected time for RLS_k to identify the optimum is infinite), in this work we will consider reaching a fitness of at least $2n - \sqrt{n} + 1$ (i.e. the final $\sqrt{n}$ points on the ridge) as having optimised the function, as this allows all configurations with $1 \leq k \leq \sqrt{n}$ to reach the optimum. We do not consider $k > \sqrt{n}$ since these values lead to an almost random search. We ensure that all optima have the same fitness by defining $\textsc{Ridge}^*(x) := \min\{\textsc{Ridge}(x), 2n - \sqrt{n} + 1\}$ and instead configure for this function.

The black box optimisation version of $\textsc{Ridge}^*$ consists of 2^n functions. For each $a \in \{0,1\}^n$ the fitness of a solution x for the corresponding function can be calculated using the following XOR transformation: $\textsc{Ridge}^*{}_a(x) := \textsc{Ridge}^*(x_1 \oplus a_1 \dots x_n \oplus a_n)$ [26]. Under this transformation, we assume that RLS_k is initialised with the bit string a. For convenience of analysis we will use the $\textsc{Ridge}^*{}_{0^n}$ function given above where the path starts at the 0^n bit string and terminates at the 1^n bit string. The best parameter value for RLS_k for a random instance will naturally be optimal also for any other instance of the black box class.

The second optimisation problem we will consider is the well-studied OneMax benchmark function. Its black box class consists of 2^n functions each of which has a different bit string as global optimum and the fitness of a bit string decreases with the Hamming distance to the optimum [26]. We tune RLS_k for only one instance since the identified optimal parameter value will naturally also be the best one for any of the other 2^n instances of the black box class. In particular, we will use the instance:

$$\textsc{OneMax}(x) = \sum_{i=1}^{n} x_i.$$

Since it is not possible for RLS_k to make progress if the distance to the optimum is at most $\lfloor k/2 \rfloor$, we treat any search point with a distance to the optimum of at most $\lfloor \phi/2 \rfloor$ as an optimum (recall that ϕ is the largest permitted value of k). This allows us to avoid an infinite expected optimisation time. It is relevant for later analyses to note that this implies that the number of optima is therefore, using $\binom{n}{m} \leq (en/m)^m$,

$$\sum_{i=0}^{\lfloor \phi/2 \rfloor} \binom{n}{i} \leq \left(\left\lfloor \frac{\phi}{2} \right\rfloor + 1 \right) \binom{n}{\lfloor \phi/2 \rfloor} \leq \left(\left\lfloor \frac{\phi}{2} \right\rfloor + 1 \right) \cdot \left(\frac{e}{\lfloor \phi/2 \rfloor} \right)^{\lfloor \phi/2 \rfloor} \cdot n^{\lfloor \phi/2 \rfloor},$$

which is polynomial for $\phi = O(1)$. As with $\textsc{Ridge}^*$, we ensure that all optima have the same fitness by defining $\textsc{OneMax}^*(x) := \min\{\textsc{OneMax}(x), n - \lfloor \phi/2 \rfloor\}$ and configure for this function instead.

The fact that the performance of the configurators does not change with the size and contents of the training set, allows us to analyse the impact of the cutoff times and number of runs used for configuration comparisons in the ideal situation where the training set perfectly reflects the problem class for which

we aim to tune the target algorithm i.e., it is unlikely that configurators that fail to perform effectively in our settings will perform well in more sophisticated algorithm configuration scenarios. The problem of identifying a training set that allows proper generalisation is a different one that we do not aim to tackle in this paper.

2.5. Standard Mathematical Tools Used in this Work

We now introduce some standard tools that we use in this work. Theorems 1 and 2 are Chernoff bounds, a family of techniques used to bound the probability that a random variable deviates from its expected value by a given amount.

Theorem 1 (Theorems 1.10.1 and 1.10.5 in [27])**.** Let $X_1, \ldots, X_m$ be independent random variables taking values in $\{0, 1\}$. Let $X = \sum_{i=1}^{m} X_i$. Then for $0 \leq \delta \leq 1$,

$$\Pr(X \leq (1-\delta)\,\mathrm{E}[X]) \leq \exp\left(-\frac{\delta^2\,\mathrm{E}[X]}{2}\right)$$

and for $\delta \geq 0$

$$\Pr(X \geq (1+\delta)\,\mathrm{E}[X]) \leq \exp\left(-\frac{\min\{\delta, \delta^2\}\,\mathrm{E}[X]}{3}\right).$$

Theorem 2 (Theorem 1.10.7 in [27])**.** Under the same assumptions as in Theorem 1, for all $\delta \geq 0$

$$\Pr(X \geq \mathrm{E}[X] + \delta) \leq \exp\left(-\frac{2\delta^2}{m}\right)$$

$$\Pr(X \leq \mathrm{E}[X] - \delta) \leq \exp\left(-\frac{2\delta^2}{m}\right).$$

The *method of bounded martingale differences* (Theorem 3) is a powerful tool for bounding the probability that a function deviates from its expected value. It requires that the impact of each variable X_i on the expected function value is bounded by some value c_i.

Theorem 3 (Method of Bounded Martingale Differences (Theorem 3.67 in [28]))**.** Let $X_1, \ldots, X_m$ be an arbitrary set of random variables and let f be a function satisfying the property that for each $i \in \{1, \ldots, m\}$ there is a $c_i \geq 0$ such that

$$|\,\mathrm{E}[f \mid X_1, \ldots, X_i] - E[f \mid X_1, \ldots, X_{i-1}]| \leq c_i.$$

Then

$$\Pr(f \geq \mathrm{E}[f] + \delta) \leq \exp\left(-\frac{\delta^2}{2\sum_{i=1}^{m} c_i^2}\right)$$

and

$$\Pr(f \leq \mathrm{E}[f] - \delta) \leq \exp\left(-\frac{\delta^2}{2\sum_{i=1}^{m} c_i^2}\right).$$

Note that the X_i terms are *not* required to be independent.

3. General Upper and Lower Bounds for the Optimisation Time Metric

In this section we provide general upper and lower bounds on the cutoff time required by configurators that use optimisation time as performance metric. We start with a lower bound on the cutoff time which is necessary to prevent the configurator from being blind. We show that if the cutoff time is small enough such that the target algorithm does not find the optimum of any member of the training set, then any configurator that uses optimisation time as performance metric, including both ParamRLS-T and ParamILS, is blind. As a corollary it will follow that if the cutoff time is set to $\kappa \leq (n \ln n)/2$ then any such configurator is blind when tuning any unary unbiased algorithm (a class that includes RLS_k) for any function with up to $\exp(\sqrt{n}/\log^2 n)$ optima.

Theorem 4. Consider a configurator that uses optimisation time as performance metric tuning an algorithm $\mathcal{A}$ for a training set Π (with problem size n and $|\Pi| = \mathrm{poly}(n)$) with any choice of instances per comparison and any means of aggregating cost measures. Assume that, w. o. p., no configuration of the target algorithm $\mathcal{A}$ reaches the optimum of any member of the training set. Then for any polynomial number of runs per evaluation r the configurator is blind.

Proof. Since there are polynomially many comparisons, by the union bound no configuration will reach the optimum of any problem instance within the cutoff time, w. o. p. Therefore every configuration will have the same fitness: the cutoff time multiplied by the penalisation constant. This satisfies our definition of blindness. □

The theorem allows us to prove that any configurator that uses the optimisation time as performance metric with a cutoff time of at most $(n \ln n)/2$ is blind when tuning any member of a large class of target algorithms and the training set consists of problem instances each with up to $\exp(\sqrt{n}/\log^2 n)$ optima. The proof refers to a black-box complexity result [29] that shows that a large class of algorithms needs at least $(n \ln n)/2$ fitness evaluations to optimise any function with up to $\exp(\sqrt{n}/\log^2 n)$ optima. More specifically, the class of algorithms is that of *unary unbiased* black-box algorithms: algorithms that query new search points generated by unary operators i.e., those that take one search point as input and produce another search point as output. These operators must be *unbiased* as defined in [30]; in a nutshell, this means that operators treat all bit values and all bit positions in the same way. RLS_k belongs to the class of unary unbiased algorithms.

Corollary 5. Consider any configurator that uses optimisation time as performance metric tuning any unary unbiased black-box algorithm with a single parameter using a training set Π (with problem size n and $|\Pi| = \mathrm{poly}(n)$) where each instance has up to $\exp(\sqrt{n}/\log^2 n)$ optima. If the cutoff time is

$\kappa \leq (n \ln n)/2$ and the number of runs per comparison is polynomial, then the configurator is blind.

Proof. Applying [29, Theorem 20] with $\delta := 1/2$ tells us that all unary unbiased black-box algorithms require at least $(n \ln n)/2$ iterations to reach the optimum of any function with up to $\exp(\sqrt{n}/\log^2 n)$ optima, with probability $1 - \exp(-\Omega(\sqrt{n}/\log n))$. By the union bound, the probability that none of the polynomially many runs of the algorithm reach the optimum within $(n \ln n)/2$ iterations is still overwhelming. Then Theorem 4 implies that the tuner is blind. □

We now derive an upper bound on the cutoff time that is sufficient for configurators that use the optimisation time metric to identify the T-optimal configuration if this configuration wins any comparison against any other configuration w. o. p.

Theorem 6. Consider a configurator that uses optimisation time as performance metric tuning an algorithm $\mathcal{A}$ using a training set Π' with instance sizes n. Assume that in each iteration the configurator samples a new configuration and compares it against the best-found so far. Assume also that there are $|\Theta|$ configurations of $\mathcal{A}$, that the configurator uses at most r runs per configuration evaluation, that the cutoff time is large enough such that the configuration with the smallest expected optimisation time, θ^*, reaches the optimum of every instance $\pi \in \Pi'$, w. o. p., and that θ^* reaches the optimum of every instance $\pi \in \Pi'$ in less time than all other configurations, w. o. p.

- If the configurator uses a mutation operator that samples configurations uniformly at random, then the expected number of comparisons sufficient to sample θ^* is $|\Theta|$. After running for t comparisons, the probability that the best-found configuration is θ^* is at least

$$(1 - \exp(-\Omega(n^\varepsilon)))^{rt} \cdot \left(1 - \left(1 - \frac{1}{|\Theta|}\right)^t\right),$$

  for some constant $\varepsilon > 0$.

- Assume that the configurator uses a global mutation operator: at every step it samples every configuration with probability at least $p_{\min} > 0$. Then, for every constant $\varepsilon > 0$, after $t = \Omega(n^\varepsilon / p_{\min})$ comparisons, assuming $rt \in \text{poly}(n)$, the configurator has sampled θ^* and it remains unbeaten w. o. p.

Proof. By the waiting time argument (i.e. that a geometric random variable with success probability p requires, in expectation, $1/p$ trials before a success occurs), the expected number of configuration samples required before sampling θ^* is $|\Theta|$.

Since by assumption the cutoff time is large enough to allow θ^* to reach the optimum of each training instance w. o. p. and, also by assumption, it reaches the optimum of each training instance in less time so in less time than all other

configurations, also w. o. p., the probability that both of these events occur in all of the at most rt comparisons (and thus a lower bound on the probability that it will win all comparisons in which it is involved) is the product of two overwhelming probabilities raised to the power rt, which is equal to

$$\left(1 - \exp\left(-\Omega(n^{\varepsilon})\right)\right)^{rt},$$

for some constant $\varepsilon > 0$. The second term in the stated probability is the probability that θ^* is sampled at least once in t comparisons.

For the second claim, the probability that θ^* is *not* sampled within $t = \Omega(n^{\varepsilon}/p_{\min})$ steps is at most $(1 - p_{\min})^t \leq e^{-tp_{\min}} \leq e^{-\Omega(n^{\varepsilon})}$. Hence, w. o. p. θ^* is sampled within t steps. Once θ^* is sampled, it is not beaten in any remaining comparison w. o. p.: as $rt \in \text{poly}(n)$, a union bound over polynomially many events that all occur w. o. p. (as shown in the proof of the first claim) proves that the claimed events occur w. o. p. □

Note that the condition that the T-optimal configuration reaches the optimum of *all* training instances in the least time is likely to be far stronger than is required in practice (where it is only necessary that this configuration has a smaller mean penalised optimisation time than its competitors over the sampled training instances).

For configurators that sample configurations uniformly at random without replacement, such as ParamILS when configuring single-parameter algorithms, it is possible to derive stronger guarantees on the time required to identify the optimal configuration.

Theorem 7. Consider ParamILS for the configuration of an algorithm $\mathcal{A}$ with a single parameter θ for a training set Π' with instance sizes n, arbitrary polynomial values for r, s, ρ and arbitrary p_{restart}. Assume that the domain of θ has a size ϕ that is at most polynomial in n. Assume that the cutoff time is large enough such that the configuration with the smallest expected optimisation time, θ^*, reaches the optimum of every instance $\pi \in \Pi'$, w. o. p. Assume also that θ^* reaches the optimum of every instance $\pi \in \Pi'$ faster than all other configurations w. o. p. Then $\rho+\phi$ comparisons suffice for ParamILS to return θ^*, w. o. p.

Proof. After the first $\rho + 1$ comparisons, ParamILS starts the *IterativeFirstImprovement* procedure. In this procedure, since the target algorithm has only a single parameter, the set of unvisited neighbours of a configuration consists of *all* configurations which have not yet been sampled during that call to the procedure. Since there are only ϕ configurations in the scenario which we consider, this implies that θ^* is discovered within $\phi - 1$ comparisons with probability 1.

Now, θ^* will be returned by *IterativeFirstImprovement* if, once sampled, it wins every subsequent comparison. By assumption, the cutoff time is large enough such that this happens w. o. p. in any given comparison. Taking a union bound over $\rho+\phi$ comparisons, the probability that, once sampled, θ^* never loses a comparison is overwhelming. □

4. Tuning RLS_k for Ridge*

From the general case analysed in Section 3 we now turn our attention to the ability of ParamRLS and ParamILS to configure RLS_k for the Ridge* benchmark problem class. This allows us to characterise the behaviour of these configurators for ranges of cutoff times. We prove that ParamRLS-F is able to return the T-optimal and F-optimal configuration $k = 1$ for RLS_k and Ridge* for any cutoff time. The range of permitted parameter values goes up to $\sqrt{n}$ (i.e. $\phi = \sqrt{n}$) as larger values of k degrade to an almost random search. If the cutoff time is large enough (i.e. $\kappa = \omega(n)$), then even just one run per configuration evaluation suffices. For smaller cutoff times, ParamRLS-F requires more runs per configuration evaluation to identify that RLS_1 is optimal. We will show this for the extreme case $\kappa = 1$, where $n^{3/2}$ runs per comparison suffice for ParamRLS-F to return $k = 1$ w. o. p. On the other hand, ParamRLS-T and ParamILS will be blind for any $\kappa < (1 - \varepsilon)n^2$, for every constant $\varepsilon > 0$. Note that this negative result for Ridge* is stronger than the general negative results provided by Corollary 5 (although that result holds for a wide class of functions including Ridge*).

4.1. Analysis of RLS_k Optimising Ridge*

Before we can analyse the performance of the tuners when configuring for Ridge*, we must first gain an understanding of how the value of k affects the performance of RLS_k on this function class. We therefore derive the expected optimisation time with respect to k, and then show that this is minimised for $k = 1$ (i.e. $k = 1$ is T-optimal).

Lemma 8. For $k \leq n/2$, when initialised at 0^n the expected optimisation time of RLS_k on Ridge* is $\lceil \frac{n-\sqrt{n}+1}{k} \rceil \binom{n}{k}$.

Proof. During a single iteration, it is only possible to increase the fitness of an individual by exactly k since we must flip exactly the first k zeroes in the bit string (any other combination of flips will mean that the string is no longer in the form $1^i 0^{n-i}$ and will be rejected). We call an iteration in which we flip exactly the first k zeroes in the bit string an *improvement*. There are $\binom{n}{k}$ possible ways in which we can flip k bits and exactly one of these combinations flips the first k zeroes. Therefore the probability of making an improvement at any given time is $1/\binom{n}{k}$.

By the waiting time argument, we wait $\binom{n}{k}$ iterations in expectation to make a single improvement. Since the algorithm is initialised at 0^n, we need to make $\lceil (n-\sqrt{n}+1)/k \rceil$ improvements in order to reach the optimum. We therefore wait $\lceil (n - \sqrt{n} + 1)/k \rceil \binom{n}{k}$ iterations in expectation until we reach the optimum. □

Corollary 9. A value of $k = 1$ leads to the shortest expected optimisation time for RLS_k on Ridge* for any $k \leq n/2$.

The optimisation time is highly concentrated around the expectation, with deviations from $\binom{n}{k}(n/k)$ by a factor of $(1 \pm \varepsilon)$, for every constant $\varepsilon > 0$, having an exponentially small probability.

Lemma 10. With probability at least $1 - \exp(-\Omega(n/k))$, when initialised at 0^n RLS_k requires at least $(1-\varepsilon)\binom{n}{k}(n/k)$ and at most $(1+\varepsilon)\binom{n}{k}(n/k)$ iterations to optimise $\textsc{Ridge}^*$, for every constant $\varepsilon > 0$ and large enough n.

Proof. Let X^i equal 1 if RLS_k makes an improvement in iteration i and equal 0 otherwise. Let X_t denote the number of improvements made by RLS_k within the first t iterations. That is, $X_t = \sum_{i=1}^{t} X^i$. Recall that RLS_k can only make improvements by exactly k when optimising $\textsc{Ridge}^*$ and therefore needs to make $\lceil (n - \sqrt{n} + 1)/k \rceil$ improvements to reach the optimum. Since the probability of making an improvement is $1/\binom{n}{k}$ when at any non-optimal point, we have $\mathrm{E}[X_t] \leq t/\binom{n}{k}$, by the linearity of expectation (this is not an equality since progress stops once we reach an optimum).

Let $t = (1-\varepsilon)\binom{n}{k}\frac{n}{k}$, for a constant ε satisfying $\varepsilon > 0$. For $\varepsilon \geq 1$ the claim that the runtime is at least t holds automatically since $t \leq 0$. To prove the claim for $\varepsilon < 1$, we first observe that $\mathrm{E}[X_t] \leq (1-\varepsilon)\frac{n}{k}$. We now apply Chernoff bounds (Theorem 1), treating this upper bound on $\mathrm{E}[X_t]$ as an equality (this Chernoff bound remains correct despite an upper bound on the expectation being used [27]):

$$\begin{aligned}\Pr\left(X_t \geq \left\lceil \frac{n-\sqrt{n}+1}{k} \right\rceil\right) &= \Pr\left(X_t \geq \left(1 + \left(\frac{\left\lceil \frac{n-\sqrt{n}+1}{k} \right\rceil}{(1-\varepsilon)\frac{n}{k}} - 1\right)\right)\mathrm{E}[X_t]\right) \\ &\leq \exp\left(-\frac{\Theta(1)\cdot\Theta(n)}{3\cdot\Theta(k)}\right) = \exp(-\Omega(n/k)).\end{aligned}$$

The above Chernoff bound holds as $\lceil (n-\sqrt{n}+1)/k \rceil / ((1-\varepsilon)(n/k)) - 1 = \Theta(1)$ is positive for large enough n.

We proceed similarly to obtain an upper bound on the runtime. However, since we cannot use the upper bound on $\mathrm{E}[X_t]$ in the Chernoff bound for the lower tail (as we did above), we instead assume that the algorithm is operating on an infinite bit string and thus can make progress at any time, still with probability $1/\binom{n}{k}$. The time required by this modified process to make $\lceil (n - \sqrt{n}+1)/k \rceil$ improvements is identical to that required to do so by RLS_k on $\textsc{Ridge}^*$. This time, we let $t = (1+\varepsilon)\binom{n}{k}\frac{n}{k}$, which yields $\mathrm{E}[X_t] = (1+\varepsilon)\frac{n}{k}$. Hence, again by Chernoff bounds:

$$\begin{aligned}&\Pr\left(X_t < \left\lceil \frac{n-\sqrt{n}+1}{k} \right\rceil\right) \leq \Pr\left(X_t \leq \left\lceil \frac{n-\sqrt{n}+1}{k} \right\rceil\right) \\ &= \Pr\left(X_t \leq \left(1 - \left(1 - \frac{\left\lceil \frac{n-\sqrt{n}+1}{k} \right\rceil}{(1+\varepsilon)\frac{n}{k}}\right)\right)\mathrm{E}[X_t]\right) \\ &\leq \exp\left(-\frac{\Theta(1)\cdot\Theta(n)}{2\cdot\Theta(k)}\right) = \exp(-\Omega(n/k)).\end{aligned}$$

The above Chernoff bound holds as $1 - \lceil (n-\sqrt{n}+1)/k \rceil / ((1+\varepsilon)(n/k)) = \Theta(1)$

converges to $1 - 1/(1+\varepsilon) < 1$ for large enough n. $\square$

The tight runtime bounds derived in Lemma 10 mean that we can now consider the relative performance of RLS_a and RLS_b on RIDGE*, for some $a < b$. We first derive a general bound which can be applied to any two random processes with probabilities of improving which stay the same throughout the process. We derive a lower bound on the probability that the process with the higher probability of improving is ahead at some time t. We then apply this result to RLS_a and RLS_b optimising RIDGE*.

Lemma 11. Let $\mathcal{A}$ and $\mathcal{B}$ be two random processes which both take values from the non-negative real numbers, and both start with value 0. At each time step, $\mathcal{A}$ increases by some real number $\alpha \geq 0$ with probability p_a, and otherwise stays put. At each time step, $\mathcal{B}$ increases by some real number $\beta \geq 0$ with probability p_b, and otherwise stays put. Let Δ_t^a and Δ_t^b denote the total progress of $\mathcal{A}$ and $\mathcal{B}$ in t steps, respectively. Let $q := p_a(1-p_b) + (1-p_a)p_b$, $q_a := p_a(1-p_b)/q$, and $q_b := p_b(1-p_a)/q$. Then, for all $0 \leq p_b \leq p_a$ and $\alpha, \beta \geq 0$

$$\Pr(\Delta_t^b \geq \Delta_t^a) \leq \exp\left(-qt\left(1 - 2q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)\right)$$

Proof. Let $q := p_a(1-p_b) + (1-p_a)p_b$ be the probability that exactly one process makes progress in a single time step. Let $q_a := p_a(1-p_b)/q$ be the conditional probability of $\mathcal{A}$ making progress, given that one process makes progress, and define q_b likewise. Assume that in t steps we have ℓ progressing steps. Then the probability that $\mathcal{B}$ makes at least as much progress as $\mathcal{A}$ is $\Pr(\text{Bin}(\ell, q_b) \geq \lceil \ell\alpha/(\alpha+\beta) \rceil)$. Then,

$$\Pr(\Delta_t^b \geq \Delta_t^a) = \sum_{\ell=0}^{t} \Pr(\text{Bin}(t,q) = \ell) \cdot \Pr(\text{Bin}(\ell, q_b) \geq \lceil \ell\alpha/(\alpha+\beta) \rceil) \tag{1}$$

Note that $p_b \leq p_a$ is equivalent to $q_b \leq q_a$. Thus, $q_b/q_a \leq 1$. Hence

$$\begin{aligned}
&\Pr(\text{Bin}(\ell, q_b) \geq \lceil \ell\alpha/(\alpha+\beta) \rceil) = \sum_{i=\lceil \ell\alpha/(\alpha+\beta) \rceil}^{\ell} \binom{\ell}{i} q_b^i q_a^{\ell-i} \\
&= \sum_{i=\lceil \ell\alpha/(\alpha+\beta) \rceil}^{\ell} \binom{\ell}{i} q_b^{\ell\alpha/(\alpha+\beta)} q_a^{\ell-(\ell\alpha/(\alpha+\beta))} (q_b/q_a)^{i-(\ell\alpha/(\alpha+\beta))} \\
&\leq 2^\ell q_b^{\ell\alpha/(\alpha+\beta)} q_a^{\ell-(\ell\alpha/(\alpha+\beta))} = \left(2q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)^\ell.
\end{aligned}$$

Using the above in (1) and $\Pr(\text{Bin}(t,q) = \ell) = \binom{t}{\ell} q^\ell (1-q)^{t-\ell}$ yields

$$\begin{aligned}\Pr(\Delta_t^b \geq \Delta_t^a) &\leq \sum_{\ell=0}^{t} \binom{t}{\ell} q^\ell (1-q)^{t-\ell} \cdot \left(2q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)^\ell \\ &= \sum_{\ell=0}^{t} \binom{t}{\ell} (1-q)^{t-\ell} \cdot \left(2q \cdot q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)^\ell\end{aligned}$$

(using the Binomial Theorem)

$$\begin{aligned}&= \left(1 - q + 2q \cdot q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)^t \\ &= \left(1 - q\left(1 - 2q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)\right)^t \\ &\leq \exp\left(-qt\left(1 - 2q_b^{\alpha/(\alpha+\beta)} q_a^{\beta/(\alpha+\beta)}\right)\right). \qquad \square\end{aligned}$$

Applying this lemma allows us to derive a lower bound on the probability that RLS_a wins a comparison against RLS_b (with $a < b$) with a cutoff time of κ. Additional arguments for small $\kappa/\binom{n}{a}$ show that the probability that RLS_a wins is always at least $1/2$.

Lemma 12. For every $1 \leq a < b = o(n)$, in a comparison with a single run on RIDGE* with cutoff time κ, RLS_a wins a comparison against RLS_b with probability at least

$$\max\left\{\frac{1}{2},\ 1 - \exp\left(-\kappa/\binom{n}{a} \cdot (1 - o(1))\right) - \exp(-\Omega(n/b))\right\}$$

Proof. Using the notation from Lemma 11, we have $p_a = 1/\binom{n}{a}$ and $p_b = 1/\binom{n}{b}$, which implies $p_b = o(p_a)$ since $b = o(n)$. Further, $q \geq 1/\binom{n}{a} \cdot (1 - o(1))$, $q_a = 1 - o(1)$ and $q_b = p_b(1-p_a)/q \leq p_b(1-p_a)/(p_a(1-p_b) \leq p_b/p_a = \frac{b!(n-b)!}{a!(n-a)!} \leq (b/(n-b))^{b-a}$. This implies $q_b^{a/(a+b)} \leq (b/(n-b))^{a(b-a)/(a+b)}$. Using $b/(n-b) = o(n)/n = o(1)$ and $a(b-a)/(a+b) \geq a/(2a+1) \geq 1/3$, we obtain $q_b^{a/(a+b)} = o(1)$. Lemma 11 tells us that RLS_a is ahead of RLS_b with probability at least

$$1 - \exp\left(-\kappa/\binom{n}{a} \cdot (1 - o(1))\right).$$

The above argument ignores that progress stops once a global optimum is reached. If RLS_a reaches a global optimum and RLS_b does not, RLS_a still wins. We use the union bound to include a term reflecting the possibility that RLS_b finds the global optimum. By Lemma 10, if $\kappa \leq (1-\varepsilon)\binom{n}{b}\frac{n}{b}$, for constant ε satisfying $0 < \varepsilon < 1$, the probability that RLS_b does find the optimum is at most $\exp(-\Omega(n/b))$. For $\kappa \leq (1-\varepsilon)\binom{n}{b}\frac{n}{b}$ this proves a lower bound of

$$1 - \exp\left(-\kappa/\binom{n}{a} \cdot (1 - o(1))\right) - \exp(-\Omega(n/b)). \tag{2}$$

For larger κ we argue that by Lemma 10, the probability that RLS_a finishes within the first $(1-\varepsilon)\binom{n}{b}\frac{n}{b} \geq (1+\varepsilon)\binom{n}{a}\frac{n}{a}$ steps is $1-\exp(-\Omega(n/a)) \geq 1-\exp(-\Omega(n/b))$. Along with the fact that RLS_b with probability $1-\exp(-\Omega(n/b))$ needs more than $(1-\varepsilon)\binom{n}{b}\frac{n}{b}$ steps, this proves that RLS_a wins with probability at least $1-\exp(-\Omega(n/b))$ for $\kappa > (1-\varepsilon)\binom{n}{b}\frac{n}{b}$.

We have proved the claim for all $\kappa \geq \binom{n}{a}$, assuming n is large enough to make (2) at least as large as $1/2$. For $\kappa < \binom{n}{a}$ we additionally have to show that the probability of RLS_a winning a comparison against RLS_b is at least $1/2$. To this end, we argue that RLS_b can only win if it makes progress in κ steps. The probability for this is at most $\kappa/\binom{n}{b}$, by the union bound. RLS_a wins for sure if it does make progress in κ steps and RLS_b does not make progress. The probabilities for these events are at least $1-\left(1-1/\binom{n}{a}\right)^{\kappa} \geq \kappa/(\kappa+\binom{n}{a})$ (using $1-(1-p)^{\lambda} \geq p\lambda/(1+p\lambda)$ [31, Lemma 6]) and $1-\kappa/\binom{n}{b} = 1-o(1)$, respectively. So the probability that they both occur is at least

$$\frac{\kappa}{\kappa+\binom{n}{a}} \cdot (1-o(1)) \geq \frac{\kappa}{2\binom{n}{a}} \cdot (1-o(1)) > \frac{\kappa}{\binom{n}{b}}$$

for large enough n. Hence, in all cases where at least one algorithm makes progress, RLS_a is more likely to win than RLS_b. In all other cases there is a tie and the probability that RLS_a is declared winner is $1/2$. This proves a lower bound of $1/2$ for the probability that RLS_a wins. □

4.2. ParamRLS-F Performance Analysis

Using the lemmata derived in the previous section, we now analyse the expected number of comparisons required before the active parameter in ParamRLS-F has been set to $k = 1$. We also bound the probability that the active parameter has not been set to $k = 1$ after a given number of comparisons. The following theorem shows that, for large enough cutoff times, one run per configuration evaluation suffices for ParamRLS-F to return the F-optimal and T-optimal configuration $k = 1$ for RLS_k optimising RIDGE*. Note that it is not sufficient for the active parameter merely to be set to $k = 1$, since it is still possible for it to then change again to a different value. We therefore require that the active parameter remains at 1 for the remainder of the tuning time. We deal with this requirement in the same theorem. For both of the following two theorems, a $\pm\{1\}$ mutation operator suffices for ParamRLS-F to be efficient.

Theorem 13. Consider ParamRLS-F for the configuration of RLS_k for RIDGE* with $\phi \leq \sqrt{n}$. Assume that ParamRLS-F uses the local search operator $\pm\{1\}$ and each evaluation consists of only a single run. Then, in expectation, for any initial active parameter value, ParamRLS-F requires at most $2\phi(\phi-1)$ comparisons before it has set the active parameter to $k = 1$ for the first time. After $t \geq 4\phi(\phi-1)$ comparisons it returns the parameter value $k = 1$ with probability at least

$$1 - 2^{-\Omega(t/\phi^2)} - t \cdot (2^{-\Omega(\kappa/n)} + 2^{-\Omega(n)}).$$

Note that this is overwhelming for $t = \Omega(\phi^2 n^\varepsilon)$, polynomial t and $\kappa = \Omega(n^{1+\varepsilon})$, for a positive constant ε.

Proof. By Lemma 12, the probability that RLS_a beats RLS_b in a comparison with any cutoff time is at least $1/2$. We can therefore model the tuning process as the value of the active parameter performing a lazy random walk[4] over the states $1, \ldots, \phi$. We pessimistically assume that the active parameter decreases and increases by 1 with respective probabilities $1/4$ and that it stays the same with probability $1/2$.

Using standard random walk arguments [32, 33], the expected first hitting time of state 1 is at most $2\phi(\phi - 1)$. By Markov's inequality, the probability that state 1 has not been reached in $4\phi(\phi - 1)$ steps is at most $1/2$. Hence the probability that state 1 is not reached during $\lfloor t/4\phi(\phi - 1) \rfloor$ periods each consisting of $4\phi(\phi - 1)$ steps is $2^{-\lfloor t/4\phi(\phi-1) \rfloor} = 2^{-\Omega(t/\phi^2)}$.

Once state 1 is reached, we remain there unless RLS_2 beats RLS_1 in a run. By Lemma 12, this event happens in a specific comparison with probability at most $2^{-\Omega(\kappa/n)} + 2^{-\Omega(n)}$. By a union bound over at most t comparisons, the probability that this ever happens is at most $t \cdot (2^{-\Omega(\kappa/n)} + 2^{-\Omega(n)})$. □

We remark that the probability bound from Theorem 13, and similarly for later results, can be refined for a superpolynomial number of comparisons t by considering only the last n^c steps, for some polynomial $n^c \leq t$. This yields a probability bound of $1 - 2^{-\Omega(\min\{t, n^c\}/\phi^2)} - \min\{t, n^c\} \cdot (2^{-\Omega(\kappa/n)} + 2^{-\Omega(n)})$.

We now show that even for the smallest possible cutoff time of $\kappa = 1$ (i.e. where each configuration is only run for a *single* iteration), ParamRLS-F can identify the optimal configuration as long as a sufficient number of runs are executed per configuration evaluation. Recall that, when comparing two configurations in ParamRLS-F using multiple runs per problem instance, the configuration that wins the most runs wins the overall comparison.

Theorem 14. Consider ParamRLS-F for the configuration of RLS_k for RIDGE* with $\phi \leq \sqrt{n}$ and local search operator $\pm\{1\}$. Assume that the number of runs is $r = n^{3/2}$ each with cutoff time $\kappa = 1$. Then in expectation the tuner requires at most $2\phi(\phi - 1)$ comparisons to set the active parameter to $k = 1$ for the first time. After $t \geq 4\phi(\phi - 1)$ comparisons it returns the value $k = 1$ with probability at least

$$1 - 2^{-\Omega(t/\phi^2)} - t \cdot \exp(-\Omega(\sqrt{n})).$$

Note that this is exponentially small for $t = \Omega(\phi^2 n^\varepsilon)$, for a positive constant ε, and polynomial t.

Proof. We begin by showing that the active parameter value remains at $k = 1$ w. o. p. once it has been set to this value for the first time. Define X as the

[4] A *lazy random walk*, at each time step, does not move with some positive probability and otherwise moves to a neighbouring state selected uniformly at random.

number of runs out of $n^{3/2}$ runs, each with cutoff time $\kappa = 1$, in which RLS_1 makes progress. Define Y as the corresponding variable for RLS_2.

Let $X_i = 1$ if RLS_1 makes progress in run i and let it equal 0 otherwise. Then $X = \sum_{i=1}^{n^{3/2}} X_i$ and by the linearity of expectation $\mathrm{E}[X] = \sum_{i=1}^{n^{3/2}} 1/n = \sqrt{n}$. By a Chernoff bound (Theorem 1),

$$\begin{aligned}\Pr\left(X \leq \sqrt{n}/2\right) = \Pr\left(X \leq \left(1 - \frac{1}{2}\right)\mathrm{E}[X]\right) &\leq \exp\left(-\frac{\left(\frac{1}{2}\right)^2\sqrt{n}}{2}\right) \\ &= \exp(-\Omega(\sqrt{n})).\end{aligned}$$

Similarly, let $Y_i = 1$ if RLS_2 makes progress in run i and let it equal 0 otherwise. Then $Y = \sum_{i=1}^{n^{3/2}} Y_i$ and by the linearity of expectation $\mathrm{E}[Y] = 2\sqrt{n}/(n-1)$. By applying Chernoff bounds again,

$$\begin{aligned}\Pr\left(Y \geq \frac{\sqrt{n}}{2}\right) = \Pr\left(Y \geq \left(1 + \frac{n-5}{4}\right)\frac{2\sqrt{n}}{n-1}\right) &\leq \exp\left(-\frac{\frac{n-5}{4}\cdot\frac{2\sqrt{n}}{n-1}}{3}\right) \\ &= \exp(-\Omega(\sqrt{n})).\end{aligned}$$

Therefore, w. o. p., RLS_1 has made progress in more of these $n^{3/2}$ runs than RLS_2. That is, w. o. p., RLS_1 wins the comparison.

We can analyse this tuning process as a whole in the same way in which we analyse the tuning process in the proof of Theorem 13. We first observe that, in order for RLS_a to beat RLS_b (with $a < b$) in a run with cutoff time $\kappa = 1$, it is sufficient for it to have made an improvement and for RLS_b to have failed to do so. Letting A be the event that RLS_a beats RLS_b in a run with cutoff time $\kappa = 1$, we have

$$\Pr(A) \geq \frac{1}{\binom{n}{a}}\left(1 - \frac{1}{\binom{n}{b}}\right)$$

Let B denote the event that RLS_b beats RLS_a in a run with cutoff time $\kappa = 1$. Since RLS_b making progress is a necessary condition for event B to take place, we have $\Pr(B) \leq 1/\binom{n}{b}$. For large enough n, we have that

$$\frac{1}{\binom{n}{a}}\left(1 - \frac{1}{\binom{n}{b}}\right) \geq 1/\binom{n}{b}$$

which implies that $\Pr(A) \geq \Pr(B)$. This means that, for any $1 \leq x \leq r$ the probability that RLS_a wins x runs in a comparison is at least the probability that RLS_b wins x runs. Observing that if a comparison does not end in a draw then the winner must have won more runs than its competitor, we see that, since, $\Pr(A) \geq \Pr(B)$, the winner must be RLS_a with probability at least 1/2. This means that we can make the same pessimistic assumption as we do in the proof of Theorem 13: i.e. that the value of the active parameter decreases and increases by 1 with respective probabilities 1/4 and that it stays the same with

probability 1/2. By the same arguments as in the proof of Theorem 13, we have that the expected number of comparisons required to reach state $k = 1$ is at most $2\phi^2$. We also have that the probability that state $k = 1$ has not been reached after t comparisons is $2^{-\Omega(t/\phi^2)}$. Thus the probability that the tuner returns $k = 1$ after t comparisons is $1 - 2^{-\Omega(t/\phi^2)} - t \cdot \exp(-\Omega(\sqrt{n}))$. □

4.3. ParamRLS-T Performance Analysis

The analysis conducted in Section 4.2 shows that ParamRLS-F performs well regardless of the cutoff time. In this section, we show that this is not the case for ParamRLS-T. The tuner is unable to tune RLS_k for RIDGE* with any mutation operator for small cutoff times but is successful for larger cutoff times.

Theorem 15. Consider ParamRLS-T for the configuration of RLS_k for RIDGE* with $\phi \leq \sqrt{n}$, any local search operator $\pm\{1, \ldots, \ell\}$ and cutoff time $\kappa \leq (1-\varepsilon)n^2$ for any constant $\varepsilon > 0$. Then, for any polynomial choice of r, the tuner is blind.

Proof. By Lemma 10, applied for all $k \leq \sqrt{n}$, we have that for $\kappa \leq (1 - \varepsilon)n^2$, for constant ε satisfying $0 < \varepsilon < 1$, RLS_k will not have reached the optimum of RIDGE* within κ iterations, with probability at least $1 - \exp(-\Omega(n/k))$. Thus, with probability at least $1 - r \cdot t \cdot \exp(-\Omega(n))$, no configuration reached the optimum of RIDGE* in any of the r runs in any of the t comparisons. We can therefore use Theorem 4 to see that ParamRLS-T is blind. □

Whilst, as shown above, ParamRLS-T fails for smaller cutoff times, we now prove that if the cutoff time is set large enough then, w. o. p., ParamRLS-T can correctly tune RLS_k for RIDGE* even with the simple $\pm\{1\}$ mutation operator.

Theorem 16. Consider ParamRLS-T for the configuration of RLS_k for RIDGE* with $\phi \leq \sqrt{n}$. Assume that ParamRLS-T has cutoff time $\kappa \geq (1 + \varepsilon)n^2$ for any constant $\varepsilon > 0$, r runs per evaluation, where r is assumed to be polynomial, and uses the local search operator $\pm\{1\}$. Then in expectation ParamRLS-T requires at most $2\phi(\phi - 1)$ comparisons before it has set the active parameter to $k = 1$ for the first time. After $t \geq 4\phi(\phi - 1)$ comparisons it returns the parameter value $k = 1$ with probability at least

$$1 - 2^{-\Omega(t/\phi^2)} - t \cdot r \cdot \exp(-\Omega(n/\phi)).$$

Note that this is exponentially small for $t = \Omega(\phi^2 n^\varepsilon)$, for a positive constant ε and polynomial t.

Proof. According to Lemma 10, RLS_1 has reached the optimum of RIDGE* within $(1 + \varepsilon)n^2$ iterations (for constant ε satisfying $0 < \varepsilon < 1$), with probability $1 - \exp(-\Omega(n))$. Lemma 10 also implies that, with probability $1 - \exp(-\Omega(n/b)) \geq 1 - \exp(-\Omega(n/\phi))$, RLS_a reaches the optimum of RIDGE* before RLS_b, for $a < b$. This implies that, in ParamRLS-T with r runs per evaluation and t comparisons tuning RLS_k for RIDGE*, RLS_b never beats RLS_a (with $a < b$) in a comparison, with probability at least $1 - t \cdot r \cdot \exp(-\Omega(n/\phi))$.

Let us assume that the value of the active parameter performs a lazy random walk over the possible parameter values, with the parameter value 1 corresponding to an absorbing state. That is, the value of the active parameter either increases or decreases by 1 (assuming that these new value is permitted) with probability $1/4$ each, and remains the same with probability $1/2$. This is a pessimistic assumption since it is the case which arises in the scenario in which neither configuration reaches the optimum within the cutoff time. Since, as shown above, RLS_b does not beat RLS_a in a comparison w. o. p. if the cutoff time is large enough such that both configurations reach the optimum, it holds that this random walk assumption is therefore a worst-case scenario, and, provided that the cutoff time is large enough, progress towards the state $k = 1$ will in fact be faster. Note that we cannot assume that RLS_a beats RLS_b w. o. p. for all a and $b > a$ since, for some values of a, the cutoff time may not be large enough to ensure that RLS_a has reached the optimum w. o. p. The resulting tuning scenario is therefore the same as that encountered in the proof of Theorem 13. The remainder of the proof follows using the same arguments as in that proof. □

4.4. ParamILS Performance Analysis

Section 4.3 showed that the success of ParamRLS-T is heavily dependent on the cutoff time. In this section, we demonstrate that the same is true for ParamILS.

Theorem 17. Consider ParamILS for the configuration of RLS_k for RIDGE*. Assume that ParamILS has initial parameter value θ_0 chosen uniformly at random, and arbitrary polynomial values for r, s, ρ and arbitrary p_{restart}. If the cutoff time is $\kappa \leq (1 - \varepsilon)n^2$, for any constant $\varepsilon > 0$, then the tuner is blind.

Proof. Theorem 4 tells us that if, w. o. p., the target algorithm does not reach the optimum of any instance of the training set within the cutoff time then, when run for a polynomial number of comparisons, ParamILS is blind. Lemma 10 tells us that, w. o. p., no configuration of RLS_k reaches the optimum of RIDGE* within $(1 - \varepsilon)n^2$ iterations, for a positive constant ε. Combining these two results we can therefore observe that, in this context, ParamILS is blind. □

We now prove a positive result with respect to ParamILS tuning RLS_k for RIDGE*, provided that the cutoff time is large enough. In particular, we show that the first call to *IterativeFirstImprovement* results in the configuration $k = 1$ being returned w. o. p.

Theorem 18. Consider ParamILS for the configuration of RLS_k for RIDGE*, with $k \in \{1, \ldots, \phi\}$, $\phi \leq \sqrt{n}$, cutoff time $\kappa \geq (1 + \varepsilon)n^2$ for any constant $\varepsilon > 0$, arbitrary polynomial values for r, s, ρ and arbitrary p_{restart}. Then $\rho + \phi$ comparisons are sufficient for ParamILS to return the configuration $k = 1$ w. o. p.

Proof. By the same arguments as in the proof of Theorem 16, in a single run RLS_1 will reach the optimum of RIDGE* before any other configuration with

probability at least $1 - \exp(-\Omega(n/\phi))$, which is overwhelming since $\phi \leq \sqrt{n}$. Hence, RLS_1 wins a comparison against RLS with any other parameter, w. o. p. The result then follows from the general upper bound in Theorem 7. □

5. Tuning RLS_k for OneMax*

The Ridge* problem class analysed in the previous section had the same optimal parameter value regardless of the amount of time for which RLS_k was run (i.e. the F-optimal parameter value was always $k = 1$). This will not be the case for every problem class. In this section we analyse the performance of ParamRLS when configuring RLS_k for OneMax*. If RLS_k is only allowed to run for few fitness function evaluations, then the algorithm with larger values of k achieves solutions of a higher quality than the version with smaller values of k. On the other hand, if more fitness evaluations are allowed, then RLS_1 will be the fastest at identifying an optimum [34]. Our aim is to show that ParamRLS-F can identify whether $k = 1$ is the optimal parameter choice or whether a larger value for k performs better according to whether the cutoff time is small or large.

To prove our point, it suffices to consider the setting where k is permitted to take values $k \in \{1, 2, 3, 4, 5\}$. This also simplifies the analysis. We will prove that, even for a single run per configuration evaluation, ParamRLS-F identifies that $k = 1$ is optimal for any $\kappa \geq 0.975n$. This implies that ParamRLS-F is able to identify the T-optimal configuration for any such cutoff time. This time is shorter than the expected time required by any configuration to optimise OneMax* (i.e., $\Theta(n \ln n)$ [30]) and shorter than the $(n \ln n)/2$ cutoff time required by any configurator using optimisation time as performance metric. If, instead, the cutoff time is in the range $0.02n \leq \kappa \leq 0.72n$, then ParamRLS-F will identify that $k = 5$ is a better choice, as desired. Recall that we define any point with a distance to the optimal bit string of at most $\lfloor \phi/2 \rfloor = 2$ as optimal and therefore any point with a fitness of at least $n - 2$ is considered optimal. However, note that, for brevity, we refer to the bit string 1^n as "the optimum" throughout this section and therefore the "distance to the optimum" of a bit string is the Hamming distance to 1^n.

The first step in the analysis is to bound the expected progress towards the optimum in one iteration of RLS_k. We do so in the following lemma.

Lemma 19. The expected progress $\Delta_k(s)$ of RLS_k with current distance s to the optimum is

$$\Delta_k(s) = \sum_{i=\lfloor k/2 \rfloor + 1}^{k} (2i - k) \cdot \binom{s}{i} \binom{n-s}{k-i} \Big/ \binom{n}{k}$$

In particular, for $s \geq k$,

$$\begin{aligned}
\Delta_1(s) &= \frac{s}{n} \\
\Delta_2(s) &= \frac{2s(s-1)}{n(n-1)} \leq 2\left(\frac{s}{n}\right)^2 \\
\Delta_3(s) &= \frac{3s(s-1)}{n(n-1)} \leq 3\left(\frac{s}{n}\right)^2 \\
\Delta_4(s) &= \frac{8s(s-1)(s-2)(n-s/2-3/2)}{n(n-1)(n-2)(n-3)} \leq 8\left(\frac{s}{n}\right)^3 \\
\Delta_5(s) &= \frac{10s(s-1)(s-2)(n-s/2-3/2)}{n(n-1)(n-2)(n-3)} \leq 10\left(\frac{s}{n}\right)^3.
\end{aligned}$$

Proof. We first compute the probability of flipping a certain number of bits in a bit string using RLS_k. If the bit string currently has Hamming distance s to the optimum, then the probability that a k-bit mutation flips exactly i bits that disagree with the optimum and $k-i$ bits that agree with the optimum is

$$\binom{s}{i}\binom{n-s}{k-i}/\binom{n}{k} \tag{3}$$

This corresponds to a hypergeometric distribution with parameters s and n.

If a k-bit mutation flips i disagreeing bits and $k-i$ agreeing bits, the distance to the optimum decreases by $i-(k-i) = 2i-k$. This is only accepted if $2i-k \geq 0$, and progress is only made if $2i-k > 0$ or, equivalently, $i > \lfloor k/2 \rfloor$. The claim then follows from (3) and the definition of the expectation.

By [34, Lemma 28] we have $\Delta_2(s) = 2\Delta_3(s)/3$ and $\Delta_4(s) = 4\Delta_5(s)/5$, hence we only need to show the claims for $\Delta_1(s), \Delta_3(s)$, and $\Delta_5(s)$. The formula $\Delta_1(s) = s/n$ follows immediately. For $\Delta_3(s)$ we have

$$\begin{aligned}
\Delta_3(s) &= \left(\binom{s}{2}\binom{n-s}{1} + 3\binom{s}{3}\binom{n-s}{0}\right)/\binom{n}{3} \\
&= \left(\frac{s(s-1)(n-s)}{2} + \frac{3s(s-1)(s-2)}{6}\right)/\binom{n}{3} \\
&= \left(\frac{s(s-1)(n-2)}{2}\right)/\binom{n}{3} = \frac{3s(s-1)}{n(n-1)}
\end{aligned}$$

In order to bound $\Delta_3(s)$ we calculate that

$$\begin{aligned}
\frac{3s(s-1)}{n(n-1)} \leq 3\left(\frac{s}{n}\right)^2 &\iff \frac{s-1}{n-1} \leq \frac{s}{n} \iff \frac{s-1}{s} \leq \frac{n-1}{n} \\
&\iff 1 - \frac{1}{s} \leq 1 - \frac{1}{n} \iff s \leq n
\end{aligned}$$

which is trivially true. We can use a nearly identical argument to bound $\Delta_2(s)$.

For $\Delta_5(s)$ we have

$$\begin{aligned}
\Delta_5(s) = & \left(\binom{s}{3}\binom{n-s}{2} + 3\binom{s}{4}\binom{n-s}{1} + 5\binom{s}{5}\binom{n-s}{0}\right) / \binom{n}{5} \\
= & \left(\left(\frac{s(s-1)(s-2)}{6}\right)\left(\frac{(n-s)(n-s-1)}{2}\right)\right. \\
& + \frac{3s(s-1)(s-2)(s-3)(n-s)}{24} \\
& \left. + \frac{5s(s-1)(s-2)(s-3)(s-4)}{120}\right) \\
& / \left(\frac{n(n-1)(n-2)(n-3)(n-4)}{120}\right) \\
= & \frac{s(s-1)(s-2)(10n^2 - 5ns - 55n + 20s + 60)}{n(n-1)(n-2)(n-3)(n-4)} \\
= & \frac{5s(s-1)(s-2)(2n-s-3)(n-4)}{n(n-1)(n-2)(n-3)(n-4)} \\
= & \frac{10s(s-1)(s-2)(n-s/2-3/2)}{n(n-1)(n-2)(n-3)}
\end{aligned}$$

By similar arguments to those used to bound $\Delta_3(s)$ we can see that

$$\frac{10s(s-1)(s-2)(n-s/2-3/2)}{n(n-1)(n-2)(n-3)} \leq 10\left(\frac{s}{n}\right)^3 \frac{n-s/2-3/2}{n-3}$$

We therefore need to show that

$$n - \frac{s}{2} - \frac{3}{2} \leq n - 3$$

which holds if and only if $s \geq 3$. The stated bound holds if $s < 3$ since $\Delta_5(s)$ will equal 0. These two facts therefore prove the claim for all s. As above, we can use a nearly identical argument to prove the bound on $\Delta_4(s)$. □

It is well known that RLS_1 has the lowest expected optimisation time on OneMax (and thus OneMax*) for all RLS_k. It runs in expected time $n \ln n \pm O(n)$, which is best possible for all unary unbiased black-box algorithms [35, 34] up to terms of $\pm O(n)$. It is also known [35, 34] that, regardless of the fitness of the individual, flipping $2c$ bits never gives higher expected drift than flipping $2c+1$ bits (for any positive integer c)[5]. For this reason, it is necessary to be able to move from $k = 2c+1$ to $k = 2c-1$ since $k = 2c+1$ often (but not always: see Table 2, page 37) constitutes a local optimum. To enable the configurator to escape such local optima, we use the local search operator $\pm\{1, 2\}$.

[5] Although it is not necessarily optimal to maximise the drift in order to minimise the optimisation time for OneMax [36].

5.1. ParamRLS-F Identifies the Optimal Neighbourhood Size for the Cutoff Time

For large cutoff times, ParamRLS-F is able to identify the optimal parameter value $k = 1$. The analysis is surprisingly challenging as most existing methods in the runtime analysis of evolutionary algorithms are geared towards first hitting times. Results on the expected fitness after a given cutoff time (fixed-budget results) are rare [34, 25, 37, 38, 39] and do not cover RLS_k for $k > 1$.

Our analysis shows that RLS_k follows a "typical" behaviour on OneMax*. We divide a run into periods of a fixed length and show that w. o. p. at the end of each period the fitness is contained within a narrow interval of fitness values. The location of these intervals depends on k. When the tuning time is large enough, these intervals become non-overlapping, proving that one algorithm is ahead of the other w. o. p.

In the proof of Lemma 21, we make an assumption about the current distance to the optimum at the start of a new period to simplify the analysis. In brief, we assume that we start the period at the smallest distance contained in the interval. In [16] this was justified by saying that "This assumption is pessimistic here since starting period i closer to the optimum can only decrease the distance to the optimum at the end of period i.", where the assumption was "pessimistic" because it minimises the drift.

We now give a rigorous argument to show that this assumption, whilst minimising the drift, also generally decreases the distance to the optimum at the end of the period[6].

The following lemma says that, for any two distances $i < j$, the distance after t generations is generally smaller when starting close to distance i (according to a specific probability distribution), compared to when starting at distance j.

Lemma 20. Consider RLS_k on OneMax*, for an arbitrary value of $k \geq 1$, during t generations, for an arbitrary value of t. For every two integers $i \leq j$ there is a probability distribution $\alpha_{i,j}$ over distances to the optimum in $[i-k+1, i]$ such that the distance to the optimum of RLS_k after t generations, when initialised according to $\alpha_{i,j}$, is stochastically dominated by that of RLS_k after t generations, when initialised with distance j.

Proof. If $j = i$, the statement is trivial if we take $\alpha_{i,j}$ as the point distribution at $i = j$.

Assume $i < j$ and note that RLS_k has to pass through the distance interval of $I = [i-k+1, i]$ in order to progress from an initial distance j to a distance smaller than $i - k + 1$. For $\ell \in I$, let p_ℓ be the probability that the first fitness reached in I is ℓ. The distance of RLS_k after t generations, when initialised at distance ℓ,

[6] A universal statement such as "starting closer to the optimum is always better" is not true. For instance, when considering RLS_2 running on the unmodified OneMax function, starting at distance 1 to the optimum means that the algorithm cannot reach the optimum as it will always flip at least one bit incorrectly. However, RLS_2 starting at distance 2 will eventually reach the optimum, given enough time. Hence, given enough time, the algorithm achieves a better final distance when starting further away from the optimum.

is stochastically dominated by the distance of RLS_k after t generations, when initialised at distance j if we condition on traversing distance ℓ as the first distance in I; this is because the former algorithm runs for all t generations from distance ℓ and the latter algorithm runs for fewer than t generations from distance ℓ.

Note that we may have $\sum_{\ell \in I} p_\ell < 1$ since there may be a positive probability that RLS_k does not reach the interval I at all. Conditional on not reaching I, the distance when starting anywhere in I will be smaller than the distance of RLS_k starting at distance j, with probability 1. We therefore may construct $\alpha_{i,j}$ by first assigning $\alpha_{i,j}(\ell) := p_\ell$ and then distributing the remaining probability mass $1 - \sum_{\ell \in I} p_\ell$ arbitrarily to states in I. □

The following lemma splits the run of RLS_k into periods of linear length and then establishes intervals $[\ell_i, u_i]$ such that, at the end of period i, the distance to the optimum is contained in these intervals w. o. p.

Lemma 21. Consider RLS_k on OneMax* with $k = O(1)$ and a cutoff time $\kappa \geq 3.225n$. Divide the first $3.225n$ generations into 645 periods of length $n/200$ each. Define $\ell_0 = n/2 - n^{3/4}$ and $u_0 = n/2 + n^{3/4}$ and, for all $1 \leq i \leq 645$,

$$\ell_i = \ell_{i-1} - \frac{n}{200}\Delta_k(\ell_{i-1}) - o(n) \text{ and } u_i = u_{i-1} - \frac{n}{200}\Delta_k(\ell_i) + o(n).$$

Then, w. o. p. at the end of period i for $0 \leq i \leq 645$, the current distance to the optimum is in the interval $[\ell_i, u_i]$ and throughout period i, $1 \leq i \leq 645$, it is in the interval $[\ell_{i-1}, u_i]$.

Proof. We prove the statement by induction. We first show that, at time 0, the current distance to the optimum is in $[n/2 - n^{3/4}, n/2 + n^{3/4}]$ w. o. p. Let $X_i =$ if bit i is equal to 1, and let it equal 0 otherwise. Then let $X = \sum_{i=1}^{n} = X_i$ be the number of 1-bits at time 0. Since the bit string is initialised uniformly at random, $\mathrm{E}[X] = n/2$. By applying additive Chernoff bounds (Theorem 2) with $\delta = n^{3/4}$ we calculate that the number of 1-bits at initialisation is in $[n/2 - n^{3/4}, n/2 + n^{3/4}]$ with probability at least $1 - \exp(-\Omega(\sqrt{n}))$. The same applies to the distance to the optimum at this time as this is given by n minus the number of 1-bits.

Assume that at the end of period $i-1$ the current distance to the optimum is $d^*_{i-1} \in [\ell_{i-1}, u_{i-1}]$. We now derive a lower bound on the distance to the optimum at the end of period i (i.e. ℓ_i). We do so by first arguing that the assumption that $d^*_{i-1} \in [\ell_{i-1}, u_{i-1}]$ can be replaced by another assumption under which the current distance to the optimum is in the interval $[\ell_{i-1} - k + 1, \ell_{i-1}]$. According to Lemma 20, there is a probability distribution $\alpha_{\ell_{i-1}, d^*_{i-1}}$ over distances in $[\ell_{i-1} - k + 1, \ell_{i-1}]$ such that the distance to the optimum after period i when starting from $\alpha_{\ell_{i-1}, d^*_{i-1}}$ is stochastically dominated by the distance after period i when starting this period from distance d^*_{i-1}. In other words, starting in the region $[\ell_{i-1} - k + 1, \ell_{i-1}]$ is generally preferable to starting at distance d^*_{i-1}. Hence, in order to determine the next lower bound ℓ_i on the distance, we may optimistically assume that at the end of period $i - 1$, we are at some distance in

the interval $[\ell_{i-1}-k+1, \ell_{i-1}]$ and the probability that we are at each distance in this interval is given by the distribution $\alpha_{\ell_{i-1}, d^*_{i-1}}$. The precise distribution will be immaterial hereinafter; we will only use the fact that the distance at the end of period $i-1$ the distance to the optimum is in the interval $[\ell_{i-1}-k+1, \ell_{i-1}]$. When bounding ℓ_i, this replaces the original assumption that the distance to the optimum is in $[\ell_{i-1}, u_{i-1}]$.

During period i, since the current distance can only decrease and the expected progress is increasing in the distance, under this new assumption the expected progress in each step is at most $\Delta_k(\ell_{i-1})$. We now use the method of bounded martingale differences (Theorem 3) to bound the total progress in $n/200$ steps. Let us optimistically assume that the expected progress is always $\Delta_k(\ell_{i-1})$ throughout this period. Let $f(X_1, \dots, X_{\frac{n}{200}})$ be a function yielding the progress over the period given the amount of progress X_j at each iteration j. Then $f = \sum_{j=1}^{n/200} X_j$ and by the linearity of expectation $\mathrm{E}[f] = n\Delta_k(\ell_{i-1})/200$. Also, $\mathrm{E}[f \mid X_1, \dots, X_j] = \sum_{m=1}^{j} X_m + \sum_{m=j+1}^{n/200} \Delta_k(\ell_{i-1})$. Notice that, for any i, once the variable X_i is observed it subtracts a term of $\Delta_k(\ell_{i-1})$ from the expectation of f and can contribute a term of at most k. If no progress is made at time j then $X_j = 0$ and the change in the expectation of f is $-\Delta_k(\ell_{i-1})$. If progress of k is made (i.e. $X_j = k$), then the change in the expectation of f is $k - \Delta_k(\ell_{i-1})$. Thus

$$\begin{aligned} |\mathrm{E}[f \mid X_1, \dots, X_j] - \mathrm{E}[f \mid X_1, \dots, X_{j-1}]| &\leq \max\{|-\Delta_k(\ell_{i-1})|, |k - \Delta_k(\ell_{i-1})|\} \\ &\leq k =: c_j, \end{aligned}$$

since $0 \leq \Delta_k(\ell_{i-1}) \leq k$.

We are now ready to apply the method of bounded martingale differences. Applying said method with $\delta = (n/200)^{3/4} - k + 1$ yields

$$\begin{aligned} \Pr(f \geq n/200 \cdot \Delta_k(\ell_{i-1}) + (n/200)^{3/4} - k + 1) &\leq \exp\left(-\frac{((n/200)^{3/4} - k + 1)^2}{2nk^2/200}\right) \\ &= \exp(-\Omega(\sqrt{n})). \end{aligned}$$

That is, the total progress in $n/200$ steps is thus at most $n/200 \cdot \Delta_k(\ell_{i-1}) + (n/200)^{3/4} - k + 1 = n/200 \cdot \Delta_k(\ell_{i-1}) + o(n)$ w. o. p. Hence we obtain $\ell_i = \ell_{i-1} - k + 1 - \frac{n}{200}\Delta_k(\ell_{i-1}) - o(n) = \ell_{i-1} - \frac{n}{200}\Delta_k(\ell_{i-1}) - o(n)$ as a lower bound on the distance at the end of period i, w. o. p.

Since the distance in period i is at least ℓ_i, the expected progress in every step is at least $\Delta_k(\ell_i)$. Again using the method of bounded martingale differences, by the same calculations as above, the progress is at least $n/200 \cdot \Delta_k(\ell_i) - o(n)$ w. o. p. This establishes $u_i = u_{i-1} - n/200 \cdot \Delta_k(\ell_i) + o(n)$ as an upper bound on the distance at the end of period i. Taking the union bound over all failure probabilities proves the claim. □

Iterating the recurrent formulas from Lemma 21 tells us that, w. o. p., the fitnesses of different configurations differ by a linear amount after $3.225n$ itera-

tions.

Lemma 22. Define $\ell_{i,k}$ as ℓ_i Lemma 21 with respect to RLS_k. Define $u_{i,k}$ similarly. After $3.225n$ steps, w. o. p. RLS_1 is ahead of RLS_2 and RLS_3 by a linear distance: $u_{645,1} \leq \ell_{645,2} - \Omega(n)$ and $u_{645,1} \leq \ell_{645,3} - \Omega(n)$ respectively. Furthermore, w. o. p. RLS_3 is ahead of RLS_4 and RLS_5 by a linear distance: $u_{645,3} \leq \ell_{645,4} - \Omega(n)$ and $u_{645,3} \leq \ell_{645,5} - \Omega(n)$ respectively. And w. o. p. the distance to the optimum is at most $0.13n$ for RLS_1, RLS_3 and RLS_5.

In order to prove Lemma 22, however, we must first show the following result.

Lemma 23. Define $\ell_{i,k}$ and $u_{i,k}$ as in Lemma 22. Then $\ell_{i,2} \geq \ell_{i,3}$ as well as $\ell_{i,4} \geq \ell_{i,5}$ and

$$\begin{aligned}
u_{i,1} &= u_{i-1,1} - \frac{\ell_{i,1}}{200} + o(n) \\
\ell_{i,3} &\geq \ell_{i-1,3} - \frac{3\ell_{i-1,3}^2}{200n} - o(n) \\
u_{i,3} &\leq u_{i-1,3} - \frac{3\ell_{i,3}^2}{200n} + o(n) \\
\ell_{i,5} &\geq \ell_{i-1,5} - \frac{10\ell_{i-1,5}^3}{200n^2} - o(n).
\end{aligned}$$

Proof. The inequalities $\ell_{i,2} \geq \ell_{i,3}$ and $\ell_{i,4} \geq \ell_{i,5}$ follow from the fact that for even k, $\Delta_k(s) \leq \Delta_{k+1}(s)$ for all distances s [34, Lemma 28].

The other results essentially follow from Lemma 21 along with the drift bounds from Lemma 19. The equality for $u_{i,1}$ follows immediately from $\Delta_1(\ell_{i-1,1}) = \ell_{i-1,1}/n$. The lower bound for $\ell_{i,3}$ follows from $\Delta_3(\ell_{i-1,3}) \leq \frac{3\ell_{i-1,3}^2}{n^2}$ and, likewise, the lower bound for $\ell_{i,5}$ follows from $\Delta_5(\ell_{i-1,5}) \leq \frac{10\ell_{i-1,5}^3}{n^3}$. The upper bound for $u_{i,3}$ follows from $\Delta_3(\ell_{i,3}) = \frac{3\ell_{i,3}(\ell_{i,3}-1)}{n(n-1)} \geq \frac{3\ell_{i,3}^2}{n^2} - O(1/n)$. Along with a factor of $n/200$, the term $-O(1/n)$ leads to an error term of $-O(1)$ that is absorbed in the $-o(n)$ term. □

We can now prove Lemma 22.

Proof of Lemma 22. We first argue that it is safe to focus on the leading constants in the recurrences given in Lemma 23, that is, that the terms of $o(n)$ can essentially be neglected. Since the drift $\Delta_k(s)$ is increasing in s, we have $\Delta_k(s - o(n)) \leq \Delta_k(s)$ and thus any negative small order terms in $\ell_{i,1}/200$, $3\ell_{i,3}/(200n)$, and $10\ell_{i,5}/(200n)$ can be ignored since the lower bounds on the distance obtained by ignoring the negative small order terms will be smaller (that is, looser) than those which could be obtained by considering them. Every application of a recurrence formula from Lemma 23 subtracts another term of $-o(n)$. But since we only consider a constant number of applications, the total error term is still $-o(n)$.

For the upper bounds, it is also not hard to show that $\Delta_k(s + o(n)) \leq \Delta_k(s) + o(1)$ for $k \in \{1, 3, 5\}$, which introduces an additional $+o(n)$ term in each application of a recurrence. By the previous arguments, the total error in a constant number of applications sums up to $+o(n)$.

This implies that, modulo small order terms, the distance to the optimum in any period can be bounded by considering the leading constants $c_{\ell,i,k}$ in $\ell_{i,k}$ and $c_{u,i,k}$ in $u_{i,k}$, when taking the inequalities as equalities. Then $c_{\ell,0,k} = c_{u,0,k} = 1/2$ for all k and $c_{\ell,i,1} = c_{\ell,i-1,1} - c_{\ell,i-1,1}/200, c_{\ell,i,3} = c_{\ell,i-1,3} - 3c^2_{\ell,i-1,3}/200, c_{\ell,i,5} = c_{\ell,i-1,5} - 10c^3_{\ell,i-1,5}/200$ and $c_{u,i,3} = c_{u,i-1,3} - 3c^2_{\ell,i,3}/200$.

We solved these recurrences numerically[7]. After 645 periods of length $n/200$, that is, after $3.225n$ iterations, we observe that all distance intervals are non-overlapping and are in what we would assume will be their final ordering (i.e. RLS_1 is the closest to the optimum, followed by RLS_3, then RLS_5: see Figure 1 for a plot of these intervals, including ones not relevant until the proof of Lemma 27). We show that this is indeed the final ordering in the proof of Lemma 25. The resulting leading constants were (we also show $c_{\ell,645,1}$ and $c_{u,645,5}$ defined similarly, though we do not need them):

$$[c_{\ell,645,1}, c_{u,645,1}] = [0.019717738, 0.022119149]$$
$$[c_{\ell,645,3}, c_{u,645,3}] = [0.085458797, 0.089099249]$$
$$[c_{\ell,645,5}, c_{u,645,5}] = [0.120636109, 0.126798327].$$

Noticing that these intervals are non-overlapping, with gaps of order $\Omega(1)$, implies the claim for the stated comparisons of bounds for RLS_1, RLS_3, and RLS_5, even when taking into account error terms of $o(n)$. The results for RLS_2 and RLS_4 follow immediately from these results along with Lemma 23.

The additional statement about the distance being at most $0.13n$ follows since all $c_{u,645,k}$ values are less than $0.13 - \Omega(1)$. □

As a by-product, we can use the techniques from the proof of Lemma 22 to derive closed-form bounds that, w. o. p., contain the distance to the optimal bit string for RLS_1 after a linear number of iterations. These statements of the bounds are not used further in this work, but we nevertheless state them here since they significantly improve on the state-of-the-art fixed-budget bounds for RLS_1 optimising OneMax [25].

Theorem 24. For constant $i \geq 0$, RLS_1 after $\kappa = (n \cdot i)/200$ iterations running on OneMax has Hamming distance s_κ to the optimal bit string that satisfies

$$\frac{n}{2}\left(\frac{199}{200}\right)^i \leq s_\kappa \leq n \cdot \left(\frac{1}{2}\left(\frac{199}{200}\right)^{i+1} + \frac{1}{400}\right),$$

[7]The code we used to do so and the data generated is available at `https://georges-hall-sheff.github.io/rlsk_om_recurrences`.

w. o. p. This implies that

$$\frac{n}{2}\left(\frac{199}{200}\right)^{200\kappa/n} \leq s_\kappa \leq n \cdot \left(\frac{1}{2}\left(\frac{199}{200}\right)^{(200\kappa/n)+1} + \frac{1}{400}\right),$$

w. o. p.

Proof. As analysed in the proof of Lemma 22, the upper and lower bounds on the coefficient of the linear term in the distance to the optimal bit string after i periods of $n/200$ iterations are $c_{\ell,i,1} = c_{\ell,i-1,1} - c_{\ell,i-1,1}/200$ and $c_{u,i,1} = c_{u,i-1,1} - c_{\ell,i,1}/200$, respectively. Recall that $c_{\ell,0} = c_{u,0} = 1/2$. These recurrence relations have solutions

$$c_{\ell,i,1} = \frac{1}{2}\left(\frac{199}{200}\right)^i$$

and

$$c_{u,i,1} = \frac{1}{2}\left(\frac{199}{200}\right)^{i+1} + \frac{1}{400}.$$

The first claims follow by multiplying these solutions by n, and the second follows by replacing i with $200\kappa/n$, the number of iterations that corresponds to the end of period i. □

Lemma 22 states that, when using solution quality as performance metric, for $\kappa = 3.225n$ smaller odd parameter values win comparisons in ParamRLS-F, w. o. p. The following lemma proves that in fact is the case for all cutoff times $\kappa \geq 3.225n$.

Lemma 25. Let (a, b) be a member of the set $\{(1, 2), (1, 3), (3, 4), (3, 5)\}$, let $\kappa \geq 3.225n$, and let RLS_a and RLS_b both be run on OneMax* for κ iterations. Then, w. o. p., RLS_a either has a higher fitness than RLS_b or, if both algorithms have reached the optimum, RLS_a did so first.

Proof. Lemma 22 proves the claim for a cutoff time of $\kappa = 3.225n$. For cutoff times larger than $3.225n$, it is possible for the algorithms that lag behind to catch up after time $3.225n$. To this end, we define the distance between two algorithms RLS_a, RLS_b with $a < b$ as $D_t^{a,b} := s_{t,b} - s_{t,a}$, where $s_{t,a}$ and $s_{t,b}$ refer to the respective distances to the optimum at time t. Initially, by Lemma 22, we have $D_t^{a,b} = \Omega(n)$ w. o. p. for all considered algorithm pairs. We will apply the negative drift theorem [40, 41] in the version for self-loops [42] to show that w. o. p. $D_t^{a,b}$ does not drop to 0 until RLS_a has found an optimum ($s_{t,a} \leq 2$).

Consider the situation where $D_t^{a,b}$ has decreased to a value of at most $n^{1/4}$. We then argue that

$$\mathrm{E}[D_{t+1}^{a,b} - D_t^{a,b} \mid 0 \leq D_t^{a,b} \leq n^{1/4}, s_{t,a} > 2, s_{t,b}] = \Omega(\Delta_a(s_{t,a})).$$

For RLS_1 and RLS_3 the above expectation is at least (using Lemma 19 and

$s_{t,1} \leq 0.13n$)

$$\Delta_1(s_{t,1}) - \Delta_3(s_{t,3}) \geq \frac{s_{t,1}}{n} - \frac{3(s_{t,1}+n^{1/4})^2}{n^2}$$
$$= \frac{s_{t,1}}{n}\left(1 - \frac{3s_{t,1}}{n} - o(1)\right) \geq \frac{s_{t,1}}{n}(1 - 3 \cdot 0.13 - o(1)) = \Omega(\Delta_1(s_{t,1})).$$

For RLS_3 and RLS_5 the above expectation is at least (using Lemma 19 and $s_{t,3} \leq 0.13n$)

$$\begin{aligned}\Delta_3(s_{t,3}) - \Delta_5(s_{t,5}) &\geq \frac{3s_{t,3}(s_{t,3}-1)}{n(n-1)} - \frac{10(s_{t,3}+n^{1/4})^3}{n^3} \\ &= \frac{3s_{t,3}(s_{t,3}-1)}{n(n-1)} - \frac{3s_{t,3}^2}{n^2}\left(\frac{10s_{t,3}}{3n} + o(1)\right) \\ &= \Omega(\Delta_3(s_{t,3}))\end{aligned}$$

The statement also follows for even b as $\Delta_b(s) < \Delta_{b+1}(s)$.

We also have $\Delta_k(s)/k \leq \Pr(s_{t+1,k} < s_{t,k}) \leq \Delta_k(s)$ for all k, s. The above calculations have further established $\Delta_b(s_{t,b}) = O(\Delta_a(s_{t,a}))$. Hence $\Pr(D_{t+1}^{a,b} \neq D_t^{a,b}) = \Theta(\Delta_a(s_{t,a}))$.

This implies that the first condition of the negative drift theorem with self-loops [42] is satisfied with respect to $D_t^{a,b}$ and the interval $[0, n^{1/4}]$. The second condition is trivial as the jump length is bounded by $b = O(1)$. Applying said theorem yields that probability of RLS_b catching up to RLS_a before RLS_a finds an optimum in $2^{\Omega(n^{1/4})}$ generations is $e^{-\Omega(n^{1/4})}$. By Markov's inequality, the probability that RLS_a has not found an optimum within this time is $\Theta(n \log n) \cdot 2^{-\Omega(n^{1/4})} = e^{-\Omega(n^{1/4})}$. Summing up all failure probabilities proves the claim. □

Lemma 25 implies that w. o. p. RLS_1 has a smaller optimisation time than any rival configuration and RLS_3 has a smaller optimisation time than RLS_4 and RLS_5. We prove this in the following corollary.

Corollary 26. The following statements hold when optimising OneMax*:

- RLS_3 has a smaller optimisation time than RLS_4 and RLS_5, w. o. p.
- RLS_1 has a smaller optimisation time than RLS_k with $k \in \{2, 3, 4, 5\}$, w. o. p.

Proof. We prove the claim for RLS_3 in a comparison against RLS_5. The same technique can be used to prove the result for RLS_3 vs. RLS_4, RLS_1 vs. RLS_2, and RLS_1 vs. RLS_3. The remaining claims for RLS_1 hold by transitivity.

Lemma 25 tells us that, for cutoff times $\kappa \geq 3.225n$, RLS_3 beats RLS_5 in a comparison in ParamRLS-F w. o. p. Recall that, in ParamRLS-F, RLS_a beats RLS_b in a comparison if and only if: (1) RLS_a has a higher fitness than RLS_b at the cutoff time; or (2) both configurations have the same fitness at the cutoff time, but RLS_a reached this fitness first.

Let τ_3 and τ_5 be the expected optimisation times of RLS_3 and RLS_5, respectively, and let $\tau = \max\{\tau_3, \tau_5\}$. Then, by Markov's inequality, both algorithms have reached an optimum within $e^n \cdot \tau$ iterations, w. o. p. We can therefore apply Lemma 25 since $\kappa \geq 3.225n$. Lemma 25 tells us that, given both configurations have reached an optimum (and therefore the same fitness) w. o. p., then RLS_3 did so first, w. o. p. That is, in a run with $\kappa = e^n \cdot \tau$, RLS_3 took less time than RLS_5 to reach an optimum, w. o. p. Taking a union bound over all exponentially small failure probabilities proves the claim.

Proving the claim for a run with $\kappa = e^n \cdot \tau$ implies the claim for runs with any other κ. □

From Lemma 25 it follows that w. o. p. configurations with smaller odd values of k will beat their opponents in a ParamRLS-F comparison with cutoff time $\kappa \geq 3.225n$. We now use the techniques introduced in its proof to analyse the relative performance of configurations in ParamRLS-F when using a cutoff time of $\kappa \geq 0.02n$.

Lemma 27. For ParamRLS-F configuring RLS_k for OneMax*, for cutoff times $\kappa \geq 0.02n$, the fitness landscape induced by the configurator has the structure given in the following table:

Region	Cutoff Time	Ordering of Configurations
A	$\kappa \in [0.020n, 0.375n]$	$\text{RLS}_5 > \text{RLS}_4 > \text{RLS}_3 > \text{RLS}_1 > \text{RLS}_2$
	$\kappa \in (0.375n, 0.495n)$	$\text{RLS}_5 > \{\text{RLS}_3, \text{RLS}_4\} > \text{RLS}_1 > \text{RLS}_2$
B	$\kappa \in [0.495n, 0.590n]$	$\text{RLS}_5 > \text{RLS}_3 > \text{RLS}_4 > \text{RLS}_1 > \text{RLS}_2$
	$\kappa \in (0.590n, 0.645n)$	$\text{RLS}_5 > \text{RLS}_3 > \{\text{RLS}_1, \text{RLS}_4\} > \text{RLS}_2$
C	$\kappa \in [0.645n, 0.720n]$	$\text{RLS}_5 > \text{RLS}_3 > \text{RLS}_1 > \text{RLS}_4 > \text{RLS}_2$
	$\kappa \in (0.720n, 0.975n)$	$\{\text{RLS}_1, \text{RLS}_3, \text{RLS}_5\} > \text{RLS}_4 > \text{RLS}_2$
D	$\kappa \in [0.975n, 1.760n]$	$\text{RLS}_1 > \text{RLS}_3 > \text{RLS}_5 > \text{RLS}_4 > \text{RLS}_2$
	$\kappa \in (1.760n, 2.130n)$	$\text{RLS}_1 > \text{RLS}_3 > \text{RLS}_5 > \{\text{RLS}_2, \text{RLS}_4\}$
E	$\kappa \in [2.130n, 2.535n]$	$\text{RLS}_1 > \text{RLS}_3 > \text{RLS}_5 > \text{RLS}_2 > \text{RLS}_4$
	$\kappa \in (2.535n, 3.225n)$	$\text{RLS}_1 > \text{RLS}_3 > \{\text{RLS}_2, \text{RLS}_5\} > \text{RLS}_4$
F	$\kappa \geq 3.225n$	$\text{RLS}_1 > \text{RLS}_3 > \text{RLS}_2 > \text{RLS}_5 > \text{RLS}_4$

Table 2: Ordering of configurations for all cutoff times $\kappa \geq 0.020n$. "$\text{RLS}_a > \text{RLS}_b$" indicates that RLS_a has a higher fitness than RLS_b at the cutoff time w. o. p. and "$\{\text{RLS}_a, \text{RLS}_b\}$" indicates that we cannot draw any conclusions about the relationship between RLS_a and RLS_b. The region names correspond to Figure 1.

Proof of Lemma 27. We use a similar approach to that used in the proof of Lemma 22. We again use periods of length $n/200$ to determine the cutoff times at which the fitness of the individuals in different configurations of RLS_k are distinct by a linear amount. In addition to the quantities defined in the proof

of Lemma 22, we similarly define

$$\begin{aligned}
\ell_{i,1} =& \ell_{i-1,1} - \frac{\ell_{i-1,1}}{200} + o(n)\\
\ell_{i,2} \geq& \ell_{i-1,2} - \frac{2\ell_{i-1,2}^2}{200n} - o(n)\\
u_{i,2} \leq& u_{i-1,2} - \frac{2\ell_{i,2}^2}{200n} + o(n)\\
\ell_{i,4} \geq& \ell_{i-1,4} - \frac{8\ell_{i-1,4}^3}{200n^2} - o(n)\\
u_{i,4} \leq& u_{i-1,4} - \frac{8\ell_{i,4}^3}{200n^2} - o(n)\\
u_{i,5} \leq& u_{i-1,5} - \frac{10\ell_{i,5}^3}{200n^2} - o(n).
\end{aligned}$$

and obtain that the coefficients of their $\Theta(n)$ terms are $c_{\ell,i,1} = c_{\ell,i-1,1} - c_{\ell,i-1,1}/200$, $c_{\ell,i,2} = c_{\ell,i-1,2} - 2c_{\ell,i-1,2}^2/200$, $c_{u,i,2} = c_{u,i-1,2} - 2c_{\ell,i,2}^2/200$, $c_{\ell,i,4} = c_{\ell,i-1,4} - 8c_{\ell,i-1,4}^3/200$, $c_{u,i,4} = c_{u,i-1,4} - 8c_{\ell,i,4}^3/200$, $c_{u,i,5} = c_{u,i-1,5} - 10c_{\ell,i,5}^3/200$.

Iterating these recurrences in the same way as in the proof of Lemma 22, we observe that, for the named ranges of cutoff times in Table 2, the configurations are ordered in the stated way (we illustrate these intervals in Figure 1). Consider, for instance, the range of cutoff times $\kappa \in [0.975n, 1.760n]$ (we only prove the claim for this range of cutoff times: the claim for other ranges of cutoff times named in Table 2 can be proved in the same way). We prove that $\text{RLS}_1 > \text{RLS}_3 > \text{RLS}_5 > \text{RLS}_4 > \text{RLS}_2$ for all cutoff times in this range (where "$\text{RLS}_a > \text{RLS}_b$" indicates that RLS_a has a higher fitness than RLS_b w. o. p.) by recalling that by definition (Lemma 21) the interval $[\ell_{i-1,k}, u_{i,k}]$ w. o. p. contains the distance to the optimum of the individual in RLS_k throughout period i and observing that, for all periods i with $195 \leq i \leq 352$ we have $c_{\ell,i-1,1} < c_{u,i,1} < c_{\ell,i-1,3} < c_{u,i,3} < c_{\ell,i-1,5} < c_{u,i,5} < c_{\ell,i-1,4} < c_{u,i,4} < c_{\ell,i-1,2} < c_{u,i,2}$.

For cutoff times $\kappa \geq 3.225n$, the result follows from Lemma 25.

For the unnamed ranges in Table 2 it is the case that the distance intervals for almost all configurations are non-overlapping, but they are overlapping for the sets of configurations for which no ordering is stated. □

We now derive an upper bound on the expected number of comparisons before ParamRLS-F first sets the active parameter to $k = 1$ for cutoff times $\kappa \geq 3.225n$. We also derive the number of comparisons sufficient for it to return $k = 1$ as optimal for this cutoff time, w. o. p. This finally shows that linear cutoff times are sufficient for it to return the T-optimal configuration, despite cutoff times of $\Theta(n \log n)$ being necessary for any configurator that uses optimisation time as performance metric.

Theorem 28. Consider ParamRLS-F for the configuration of RLS_k for ONEMAX* with $\phi = 5$. Assume that it uses cutoff time $\kappa \geq 0.975n$, a single run per

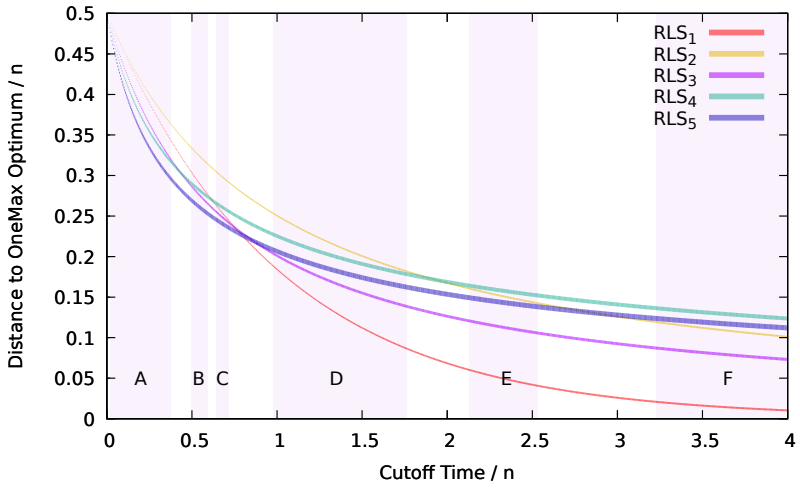

Figure 1: Intervals within which the fitness of the individual in RLS_k, with $1 \leq k \leq 5$, is contained w. o. p. Calculated using periods of length $n/200$. The distance intervals for each configuration at the end of period i for $0 \leq i \leq 800$ (corresponding to linear cutoff times $\kappa \leq 4n$) are displayed (note that each curve shown consists of 801 vertical lines indicating the interval at the end of each considered period).

configuration evaluation (i.e. $r = 1$), and that it uses the local search operator $\pm\{1, 2\}$. Then the expected number of comparisons before ParamRLS-F sets the active parameter to $k = 1$ for the first time is at most $16 + 2^{-\Omega(n^{\varepsilon})}$ (for some constant $\varepsilon > 0$). After $n^{\varepsilon'}$ comparisons, for some constant $\varepsilon' > 0$, ParamRLS-F returns the parameter value $k = 1$ w. o. p.

Proof. Lemma 27 tells us that, for all cutoff times $\kappa \geq 0.975n$, no configuration is more than two away from one that, w. o. p., will beat it in a ParamRLS-F comparison. This implies that the $\pm\{1, 2\}$ operator is sufficient to escape the local optima caused by even values of k for some ranges of cutoff times.

In the gaps between the named ranges of cutoff times given in Table 2, exactly one pair of distance intervals is overlapping (i.e. where the outcome of a comparison between them is ambiguous), meaning that the number of comparisons required to locate $k = 1$ will be no larger than that required before or after the region with the overlap. For all cutoff times $\kappa \geq 0.975n$, the interval corresponding to RLS_1 is non-overlapping with all others and is smaller than those corresponding to all other considered configurations (i.e. RLS_1 has a higher fitness than all other considered configurations by a distance of $\Omega(n)$, w. o. p.). Since for each ambiguity-free ordering of configurations given in the table the $\pm\{1, 2\}$ operator is sufficient to reach RLS_1, we can conclude that it is also sufficient at all points during the gap.

Therefore, at any point, the active parameter can reach $k = 1$ within two comparisons. Each of these two steps towards $k = 1$ happens with probability at least $1/4 - 2^{-\Omega(n^{\varepsilon})}$. The probability that in two consecutive comparisons the

active parameter reaches $k = 1$ is hence $1/16 - 2^{-\Omega(n^\varepsilon)}$. Therefore the active parameter is set to $k = 1$ within $16 + 2^{-\Omega(n^\varepsilon)}$ comparisons, in expectation.

By Markov's inequality we have that the probability that the active parameter has not been set to $k = 1$ after 32 comparisons is at most $1/2 + 2^{-\Omega(n^\varepsilon)}$. Hence the probability that it has not been set to $k = 1$ after $32n^{\varepsilon'}$ comparisons is at most $2^{-\Omega(n^{\varepsilon'})}$, for some positive constant ε'. Combining this with the fact that, in a single run, by Lemma 27, the configuration RLS_1 beats both RLS_2 and RLS_3 w. o. p., a union bound over the polynomially many comparisons proves the claim. □

We now prove that, for cutoff times $0.02n \leq \kappa \leq 0.72n$, ParamRLS-F is able to identify that $k = 5$ is optimal.

Theorem 29. Consider ParamRLS-F for the configuration of RLS_k for OneMax* with $\phi = 5$. Assume that ParamRLS-F uses cutoff time $0.02n \leq \kappa \leq 0.72n$, a single run per configuration evaluation (i.e. $r = 1$), and that it uses the local search operator $\pm\{1, 2\}$. Then the expected number of comparisons before ParamRLS-F sets the active parameter to $k = 5$ for the first time is at most $16 + 2^{-\Omega(n^\varepsilon)}$. After n^ε comparisons, for some constant $\varepsilon > 0$, the tuner returns the parameter $k = 5$ w. o. p.

Proof. As in the proof of Theorem 28, we observe that for $0.02n \leq \kappa \leq 0.72n$ no configuration is more than two away from one that, w. o. p., will beat it in a ParamRLS-F comparison. This implies that, for all cutoff times in these ranges, the $\pm\{1, 2\}$ operator is again sufficient to reach the optimal parameter value of $k = 5$.

In the gaps between the named ranges of cutoff times given in Table 2 it is again the case that exactly one pair of distance intervals is overlapping. However, for all cutoff times satisfying $0.02n \leq \kappa \leq 0.72n$ the interval corresponding to RLS_5 is non-overlapping with all others and is smaller than that corresponding to all other considered configurations (i.e. RLS_5 has a higher fitness than all other considered configurations by a distance of $\Omega(n)$). Since in each named range of cutoff times given in Table 2 the $\pm\{1, 2\}$ operator is sufficient to reach $k = 5$, we can conclude that it is also sufficient at all points during the gap.

The bounds on the number of comparisons required by the configurator to set the active parameter to $k = 5$ for the first time therefore hold by the same reasoning as in the proof of Theorem 28. □

We conclude this section by pointing out that the above analyses reveal the surprising insight that for all considered linear cutoff times, either RLS_1 or RLS_5 (or both) identify higher fitness values than RLS_3 w. o. p. (RLS_2 and RLS_4 are similarly outperformed, but this is expected). This can be observed in Figure 1 for almost all linear cutoff times $\kappa \geq 0.02n$. However, there is one region of ambiguity (for cutoff times $\kappa \in [0.72n, 0.975n]$) in which the fitness interval for RLS_3 is not distinct from those corresponding to RLS_1 and RLS_5. We resolve this ambiguity using a period length of $n/2000$ (instead of $n/200$ as otherwise used in this section), and observe that the fitness interval for RLS_3 becomes

distinct for all cutoff times in this range (see Figure 2). Therefore, for all cutoff times $\kappa \geq 0.02n$, it is always optimal to use either $k = 1$ or $k = 5$ and never optimal to use $k = 3$.

Whilst, for a range of fitness values, RLS_3 has a higher drift than both of these configurations, it is too far behind RLS_5 when entering this region of the search space (failing to overtake it before leaving the region) and not far enough ahead of RLS_1 when leaving this region (being overtaken before taking advantage of its momentarily higher drift) to become the optimal configuration at any point. It is unlikely that using $k = 3$ is optimal for smaller cutoff times, however we do not prove such a result.

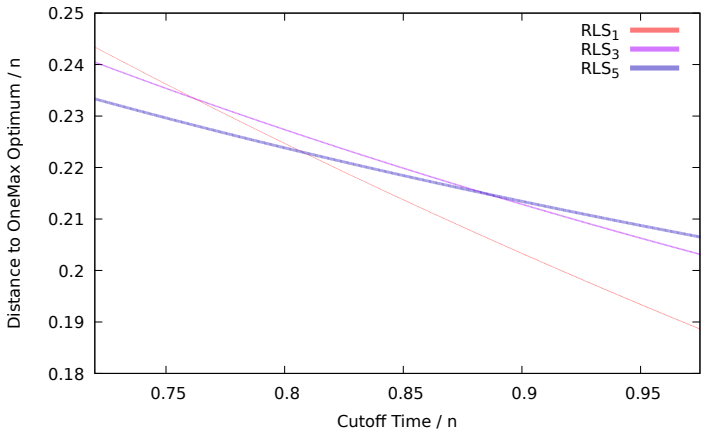

Figure 2: Intervals within which the fitness of the individual in RLS_k, with $k \in \{1, 3, 5\}$, is contained w. o. p. Calculated using periods of length $n/2000$. The distance intervals for each configuration at the end of period i for $1440 \leq i \leq 1950$ (corresponding to cutoff times $0.72n\kappa \leq 0.975n$) are displayed. Note that at no point does RLS_3 have a smaller distance to the optimum than both RLS_1 and RLS_5, w. o. p.

We can also prove that ParamRLS-F returns $k = 1$ if we update the value of the active parameter uniformly at random (that is, we don't only move to neighbours of the active parameter value). This is the mutation operator used in ParamILS, thus the theorem indicates how its performance may be improved by switching performance metric.

Theorem 30. Consider a variant of ParamRLS-F which updates the value of the active parameter by selecting a new, distinct value for k uniformly at random from the set $\{1, \ldots, \phi\}$. Assume that $\kappa \geq 3.225n$, $\phi = 5$, and any number of runs per configuration r are used. Then, when configuring RLS_k for OneMax*, the tuner requires at most 5 comparisons in expectation to set the active parameter to $k = 1$ for the first time. After t comparisons it returns $k = 1$ with probability at least $1 - (3/4 + e^{-\Omega(n^c)})^t - t \cdot r \cdot e^{-\Omega(n^c)}$ for some constant $c > 0$.

Proof. Lemma 25 tells us that, by transitivity, RLS_1 wins an individual run

in a comparison against RLS_k for $2 \leq k \leq 5$ w. o. p. Hence the probability that RLS_1 beats RLS_k, for $2 \leq k \leq 5$, in a comparison is $1 - r \cdot e^{-\Omega(n^c)}$, for some constant $c > 0$. This is overwhelmingly small since we assume r to be polynomial. Therefore, it is a sufficient condition of returning $k = 1$ to compare RLS_k with $2 \leq k \leq 5$ against RLS_1 at any time in the tuning process. That is, we simply need the tuner to set the active parameter to $k = 1$ at some point during the tuning process. Assuming that the current active parameter value is not $k = 1$, then the probability that it is set to $k = 1$ for the next iteration of the tuner is $1/4 - e^{-\Omega(n^c)}$, where the term being subtracted is the probability that RLS_1 does not win in a comparison against RLS_k with $2 \leq k \leq 5$. Therefore in expectation the tuner requires at most $1/(1/4 - e^{-\Omega(n^c)}) = 4 + e^{-\Omega(n^c)}$ comparisons (for some constant $c > 0$) before setting the active parameter to $k = 1$ for the first time. The probability that, after t comparisons, the parameter value $k = 1$ has been evaluated at some point is $1 - (1 - (1/4 - e^{-\Omega(n^c)}))^t = 1 - (3/4 + e^{-\Omega(n^c)})^t$. Subtracting the same failure probability as above yields the claim. □

5.2. ParamRLS-T Succeeds with Large Enough κ

Having shown in the previous section that ParamRLS-F is able to find a configuration appropriate for the cutoff time (i.e. an F-optimal configuration), as well as being able to identify a T-optimal configuration using linear cutoff times, we now show that ParamRLS-T is able to tune RLS_k for OneMax*, provided that we use a cutoff time of at least $n^{1+\varepsilon}$, for a positive constant ε. Note that, by Corollary 5, ParamRLS-T is blind for cutoff times of $\kappa \leq (n \ln n)/2$ since OneMax* has a sub-exponential number of optima (as noted in Section 3).

The following lemma gives an upper bound on the tail of the optimisation time of RLS_1. Its simple proof uses well-known and elementary arguments.

Lemma 31. For every initial search point, RLS_1 reaches the optimum of OneMax* within $n^{1+\varepsilon}$ iterations, for any positive constant ε, with probability at least $1 - n \exp(-n^\varepsilon)$.

Proof. A sufficient condition for RLS_1 having found the optimum is that every bit has been mutated at least once. The probability that a fixed bit i is not mutated in $n^{1+\varepsilon}$ steps is $(1 - 1/n)^{n^{1+\varepsilon}} \leq \exp(-n^\varepsilon)$. The probability that there is a bit that has not been mutated in $n^{1+\varepsilon}$ steps is, by the union bound, at most $n \cdot \exp(-n^\varepsilon)$. □

We now the main result of this section.

Theorem 32. For ParamRLS-T for the configuration of RLS_k for OneMax*, let the cutoff time be $\kappa \geq n^{1+\varepsilon}$ for a positive constant ε, $\phi = 5$, local search operator $\pm\{1, 2\}$ and any polynomial number of runs r per evaluation. Then in expectation at most $32 + 2^{-\Omega(n^\varepsilon)}$ comparisons (for some constant $\varepsilon > 0$) are required for the active parameter to be set to $k = 1$ for the first time. If $t = \Omega(n^\varepsilon)$ for some constant $\varepsilon > 0$ then the tuner returns the parameter $k = 1$ w. o. p.

Proof. By Corollary 26, RLS_1 has a lower optimisation time than RLS_k, with $k \in \{2, 3, 4, 5\}$, w. o. p., and RLS_3 has a lower optimisation time than RLS_4 and RLS_5, w. o. p.

Lemma 31 implies that RLS_1 reaches the optimum within $n^{1+\varepsilon}$ iterations, w. o. p. At any point, the active parameter can reach $k = 1$ within two comparisons: in the worst case it is at $k = 5$ and can move to $k = 3$, which happens with probability at least $1/8 - 2^{-\Omega(n^\varepsilon)}$ (the probability is larger than this lower bound if the cutoff time is large enough to allow RLS_3 to reach the optimum) and then to move to $k = 1$, which happens with probability $1/4 - 2^{-\Omega(n^\varepsilon)}$. The probability that in two consecutive comparisons the active parameter reaches $k = 1$ is hence $1/32 - 2^{-\Omega(n^\varepsilon)}$. Therefore the active parameter is set to $k = 1$ within $32 + 2^{-\Omega(n^\varepsilon)}$ comparisons in expectation. □

5.3. ParamILS Succeeds with Large Enough κ

Using similar arguments to those in the previous section, we now prove that ParamILS is able to tune RLS_k for OneMax*, provided that the cutoff time is set large enough.

Theorem 33. Consider ParamILS for the configuration of RLS_k for OneMax*, with $k \in \{1, \ldots, 5\}$, cutoff time $\kappa \geq n^{1+\varepsilon}$ for a positive constant ε, arbitrary polynomial values for r, s, ρ and arbitrary p_{restart}. Then after $\rho+4$ comparisons, the configuration $k = 1$ is returned by ParamILS w. o. p.

Proof. As shown in the proof of Theorem 32, if $\kappa \geq n^{1+\varepsilon}$ then, w. o. p., RLS_1 reaches the optimum of OneMax* before any other RLS_k with $2 \leq k \leq 5$. Then the result follows from the general upper bound in Theorem 7. □

6. Conclusions

Through a comparative analysis of performance metrics that minimise optimisation time and that maximise solution quality within some time budget we have provided theoretical evidence that the most natural performance metric for a given application is not necessarily the optimal choice. In particular, we have provided general lower bounds for the optimisation time metric that hold for any parameter tuner for the configuration of any unary unbiased target algorithm for the optimisation of any function containing up to an exponential number of optima in the problem size. Furthermore, we have shown that even for simple configuration scenarios the cutoff time required using the optimisation time metric may be considerably larger than the provided lower bound, while using the fitness-based metric allows tuners to configure target algorithms in linear time in the number of possible parameter values.

One by-product of our work is the strengthened state-of-the-art of fixed budget runtime analysis or OneMax. Another is the insight that switching from the optimisation time metric to the fitness-based one may considerably speed up the standard ParamILS configurator as our analysis for OneMax shows.

An obvious advantage in the use of the fitness-based performance metric over optimisation time is that the cutoff time can be naturally set to the time budget that the target algorithm has available. On the other hand, setting the cutoff time for the optimisation time metric requires some knowledge or an informed guess of the target algorithm's optimisation time at least for the optimal parameter values, apart from the requirement of knowing the optimal fitness in the first place. Neither of these requirements are necessarily satisfied in practical applications.

While we have proven the significant reliance on appropriate choices for the cutoff time of the optimisation time performance metric, the same conclusions naturally apply to the performance metric if other solution-quality targets were used (e.g., an approximate solution of at least some quality is sought rather than the global optimum i.e., *fixed target analysis*). Appropriate bounds on the value of the cutoff time would still need to be used (i.e., at least the runtime required by the optimal parameter value to reach a solution of the sought quality) for the configurators to be able to identify the optimal parameter values while using the fitness-based metric may allow the identification of the optimal parameter values faster.

Acknowledgements

This work was supported by the EPSRC [grant number EP/M004252/1].

References

[1] A. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 124–141.

[2] T. Stützle, M. López-Ibáñez, Automated Design of Metaheuristic Algorithms, Springer International Publishing, 2019, pp. 541–579.

[3] A. R. KhudaBukhsh, L. Xu, H. H. Hoos, K. Leyton-Brown, SATenstein: Automatically building local search SAT solvers from components, Artificial Intelligence 232 (2016) 20–42.

[4] A. S. Fukunaga, Automated discovery of local search heuristics for satisfiability testing, Evolutionary Computation 16 (1) (2008) 31–61.

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

[6] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A survey of the state of the art, Journal of the Operational Research Society 64 (12) (2013) 1695–1724.

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

[8] T. Bartz-Beielstein, C. W. G. Lasarczyk, M. Preuß, Sequential parameter optimization, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC '05), Vol. 1, IEEE, 2005, pp. 773–780.

[9] T. Bartz-Beielstein, C. W. G. Lasarczyk, M. Preuß, The sequential parameter optimization toolbox, in: Experimental Methods for the Analysis of Optimization Algorithms, Springer, 2010, pp. 337–362.

[10] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: International Conference on Learning and Intelligent Optimization (LION '11), Springer, 2011, pp. 507–523.

[11] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race and iterated F-Race: An overview, in: Experimental Methods for the Analysis of Optimization Algorithms, Springer, 2010, pp. 311–336.

[12] R. Kleinberg, K. Leyton-Brown, B. Lucier, Efficiency through procrastination: Approximately optimal algorithm configuration with runtime guarantees, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17), AAAI Press, 2017, pp. 2023–2031.

[13] Y. Pushak, H. H. Hoos, Algorithm configuration landscapes: - more benign than expected?, in: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN '18), Springer, 2018, pp. 271–283.

[14] R. J. Quick, V. J. Rayward-Smith, G. D. Smith, Fitness distance correlation and ridge functions, in: International Conference on Parallel Problem Solving from Nature (PPSN '98), Springer, 1998, pp. 77–86.

[15] S. Droste, T. Jansen, I. Wegener, On the analysis of the (1+1) evolutionary algorithm, Theoretical Computer Science 276 (1-2) (2002) 51–81.

[16] G. T. Hall, P. S. Oliveto, D. Sudholt, On the impact of the cutoff time on the performance of algorithm configurators, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19), ACM, 2019, pp. 907–915.

[17] A. Auger, B. Doerr (Eds.), Theory of Randomized Search Heuristics: Foundations and Recent Developments, World Scientific, 2011.

[18] D.-C. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, A. M. Sutton, Escaping local optima using crossover with emergent diversity, IEEE Transactions on Evolutionary Computation 22 (3) (2018) 484–497.

[19] D. Corus, P. S. Oliveto, Standard steady state genetic algorithms can hill-climb faster than mutation-only evolutionary algorithms, IEEE Transactions on Evolutionary Computation 22 (5) (2017) 720–732.

[20] D.-C. Dang, A. Eremeev, P. K. Lehre, Escaping local optima with non-elitist evolutionary algorithms, in: Proc. of AAAI 2021, 2021, pp. 12275–12283.

[21] D. Corus, P. S. Oliveto, On the benefits of populations on the exploitation speed of standard steady-state genetic algorithms, Algorithmica 82 (2020) 3676–3706.

[22] P. S. Oliveto, D. Sudholt, C. Witt, A tight lower bound on the expected runtime of standard steady state genetic algorithms, in: Proc. of GECCO 2020, 2020, p. 1323–1331.

[23] D. Corus, A. Lissovoi, P. S. Oliveto, C. Witt, On steady-state evolutionary algorithms and selective pressure: Why inverse rank-based allocation of reproductive trials is best, ACM Transactions on Evolutionary Learning and Optimization 1 (1) (2021).

[24] B. Doerr, C. Doerr, T. Kötzing, The right mutation strength for multi-valued decision variables, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16), ACM, 2016, pp. 1115–1122.

[25] T. Jansen, C. Zarges, Performance analysis of randomised search heuristics operating with a fixed budget, Theoretical Computer Science 545 (2014) 39–58.

[26] S. Droste, T. Jansen, K. Tinnefeld, I. Wegener, A new framework for the valuation of algorithms for black-box optimization, in: Proceedings of the Seventh Workshop on Foundations of Genetic Algorithms (FOGA '02), Morgan Kaufmann Publishers Inc., 2002, pp. 253–270.

[27] B. Doerr, Probabilistic tools for the analysis of randomized optimization heuristics, in: Theory of Evolutionary Computation, Springer, 2020, pp. 1–87.

[28] C. Scheideler, Probabilistic methods for Coordination Problems, HNI-Verlagsschriftenreihe 78, University of Paderborn, 2000, Habilitation Thesis, available at `http://www14.in.tum.de/personen/scheideler/index.html.en`.

[29] P. K. Lehre, D. Sudholt, Parallel black-box complexity with tail bounds, IEEE Transactions on Evolutionary Computation 24 (6) (2020) 1010–1024.

[30] P. K. Lehre, C. Witt, Black-box search by unbiased variation, Algorithmica 64 (4) (2012) 623–642.

[31] G. Badkobeh, P. K. Lehre, D. Sudholt, Black-box complexity of parallel search with distributed populations, in: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII (FOGA '15), ACM, 2015, pp. 3–15.

[32] W. Feller, An Introduction to Probability Theory and Its Applications, 3rd Edition, Vol. 1, Wiley, 1968.

[33] W. Feller, An Introduction to Probability Theory and Its Applications, 2nd Edition, Vol. 2, Wiley, 1971.

[34] B. Doerr, C. Doerr, J. Yang, Optimal parameter choices via precise black-box analysis, Theoretical Computer Science 801 (2020) 1–34.

[35] B. Doerr, C. Doerr, J. Yang, Optimal parameter choices via precise black-box analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16), ACM, 2016, pp. 1123–1130.

[36] N. Buskulic, C. Doerr, Maximizing drift is not optimal for solving OneMax, Evolutionary Computation (to appear).

[37] J. Lengler, N. Spooner, Fixed budget performance of the (1+1) EA on linear functions, in: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII (FOGA '15), ACM, 2015, pp. 52–61.

[38] B. Doerr, T. Jansen, C. Witt, C. Zarges, A method to derive fixed budget results from expected optimisation times, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13), ACM, 2013, pp. 1581–1588.

[39] S. Nallaperuma, F. Neumann, D. Sudholt, Expected fitness gains of randomized search heuristics for the traveling salesperson problem, Evolutionary Computation 25 (4) (2017) 673–705.

[40] P. S. Oliveto, C. Witt, Simplified drift analysis for proving lower bounds in evolutionary computation, Algorithmica 59 (3) (2011) 369–386.

[41] P. S. Oliveto, C. Witt, Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation, arXiv preprint arXiv:1211.7184 (2012).

[42] J. E. Rowe, D. Sudholt, The choice of the offspring population size in the (1,λ) evolutionary algorithm, Theoretical Computer Science 545 (2014) 20–38.