Carolin Benjamins, Gjorgjina Cenikj, Ana Nikolikj, Aditya Mohan, Tome Eftimov, and Marius Lindauer. 2024. Instance Selection for Dynamic Algorithm Configuration with Reinforcement Learning: Improving Generalization. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24 Companion). Association for Computing Machinery, New York, NY, USA, 563–566. https://doi.org/10.1145/3638530.3654291

Instance Selection for Dynamic Algorithm Configuration with Reinforcement Learning: Improving Generalization

Carolin Benjamins
Leibniz University Hannover
Germany
Gjorgjina Cenikj
Jožef Stefan Institute
Slovenia
Ana Nikolikj
Jožef Stefan Institute
Slovenia
Aditya Mohan
Leibniz University Hannover
Germany
Tome Eftimov
Jožef Stefan Institute
Slovenia
Marius Lindauer
Leibniz University Hannover
Germany

September 9, 2024

Abstract

Dynamic Algorithm Configuration (DAC) addresses the challenge of dynamically setting hyperparameters of an algorithm for a diverse set of instances rather than focusing solely on individual tasks. Agents trained with Deep Reinforcement Learning (RL) offer a pathway to solve such settings. However, the limited generalization performance of these agents has significantly hindered the application in DAC. Our hypothesis is that a potential bias in the training instances limits generalization capabilities. We take a step towards mitigating this by selecting a representative subset of training instances to overcome overrepresentation and then retraining the agent on this subset to improve its generalization performance. For constructing the meta-features for the subset selection, we particularly

account for the dynamic nature of the RL agent by computing time series features on trajectories of actions and rewards generated by the agent's interaction with the environment. Through empirical evaluations on the Sigmoid and CMA-ES benchmarks from the standard benchmark library for DAC, called DACBench, we discuss the potentials of our selection technique compared to training on the entire instance set. Our results highlight the efficacy of instance selection in refining DAC policies for diverse instance spaces.

Keywords: dynamic algorithm configuration, reinforcement learning, instance selection, generalization

1 Introduction

DAC offers an automated solution to the task of setting algorithm hyperparameters dynamically, by determining well-performing hyperparameter schedules or policies. One way to learn such policies is through RL biedenkapp-ecai20a,adriaensen-jair22a. While conceptually appealing, RL algorithms have the notorious tendency to significantly overfit their training environments [Zhang et al.(2018), Justesen et al.(2022), Kirk et al.(2023)]. As a consequence, RL methods for DAC suffer from a lack of generalization to instances not seen during training, thereby limiting their applicability.

We take a step towards improving the generalization performance of RL policies on new test instances by subselecting representative training instances using SELECTOR [Cenikj et al.(2022)]. To capture the dynamic nature of RL, we use trajectory-based representations generated by the RL algorithm after training on the full instance set.

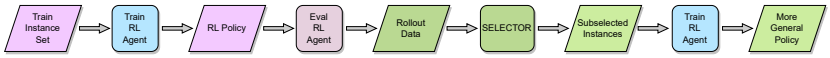

Figure 1: Our proposed flow of subselecting representative instances with SELECTOR for DAC with RL

Concretely, we make the following contributions: i) For DAC with RL, we present a principled framework to select representative instances to train on to improve generalization to the instance space; ii) we propose a new domain-agnostic approach for generating instance meta-features that encode the dynamics of the DAC problem; iii) we demonstrate superior performance training on the subselected instance set; iv) we analyze the selected instances; and v) we provide an insight on how to use the framework SELECTOR.

Reproducibility: Code and data is available here: https://github.com/automl/instance-dac.

2 Related Work

Two cornerstones of our work are: using RL to learn policies for DAC; and improving the performance of RL methods through instance subselection.

Therefore, we split out related work into works concerning each of these points in the following paragraphs.

DAC and Contextual MDPs. DAC, although originally introduced as a term by biedenkapp-ecai20a, has existed as a disparate set of ideas in traditional algorithm configuration [Lagoudakis and Littman(2000), Lagoudakis and Littman(2001), Pettinger and Everson(2002), Sharma et al.(2019)]. Follow-up work has applied these methods to learn step-size adaptation in CMA-ES [Shala et al.(2020)] and learning to select heuristics in the FastDownward planner [Speck et al.(2021)]. eimer-ijcai21a consolidate these approaches into a benchmark suite, called DACbench, that we use to study our approach. adriaensen-jair22a further provide a thorough empirical comparison between DAC methods along with traditional algorithm configuration methods on DACBench. In this work, we particularly focus on the subset of methods for DAC that use RL, first formalized by biedenkapp-ecai20a as a Contextual RL problem [Hallak et al.(2015)], where the RL agent – the solver – interacts with the environment – the algorithm – by setting its hyperparameters. Using this framework, different instances can be modeled as separate MDPs that differ in transition dynamics and rewards, forming a contextual MDP (cMDP). Learning an optimal policy in cMDPs requires additional contextual information [Benjamins et al.(2023)] that can help identify, if not characterize, the MDP in which the agent is operating. Ideally, we want sufficiently rich contextual information related to structural properties of the MDP to help the agent generalize to wider distributions of instances [Kirk et al.(2023), Mohan et al.(2024)]. For MDPs that differ in their transition dynamics, improving generalization may require information related to transition dynamics, which cannot be obtained without interacting with the MDP [Grünewälder et al.(2012), Seo et al.(2020), Guo et al.(2022)]. Consequently, attempts to characterize cMDPs generally focus on learning features by interacting with the environment instead of hand-crafting them [Han and Wu(2022), Shi et al.(2022)].

We focus on a similar problem for DAC by first characterizing the cMDP with which the agent interacts using the trajectories of the agent itself instead of hand-crafted meta-features. We then use this characterization to generate an instance subset that improves the generalization capabilities of the solver, sharing similarities with other methods that exploit structural similarities between MDPs for similar objectives [Kirk et al.(2023), Benjamins et al.(2023), Mohan et al.(2024)].

Instances Selection. In ML, the choice of benchmark data instances included in the analysis significantly influences the experimental design and statistical analysis of performance data. Evaluating the same set of algorithm instances on

different sets of benchmark data may yield varying outcomes [Cenikj et al.(2022)]. Consequently, biased performance analysis, favoring the selection of benchmark data instances in support of the winning algorithm, can inadvertently misleadingly present experimental results, compromising result generalization. Ng [Ng(2022)] recently emphasized the necessity of transitioning from big data to good data in industries where extensive datasets are lacking. Good data refers to representative learning data or datasets with unbiased and diverse characteristics, enhancing the generalization of machine learning models. Previous research in this direction involves representing data instances with meta-features and employing unsupervised learning techniques or graph algorithms to select representative learning data. These studies encompass the performance assessment of univariate time-series classification [Eftimov et al.(2022)], evaluating the performance of black-box single-objective optimization algorithms [Cenikj et al.(2022)], and training an enhanced regression model for food and biomedical prediction tasks [Ispirova et al.(2024)]. The majority of research in this area typically focuses on choosing instances based on meta-features derived from their landscape characteristics, specifically within the feature space, neglecting the performance of the algorithms. However, numerous published studies on evolutionary algorithms emphasize that there is no assurance of a correlation between landscape instance features and performance. In particular, similar landscape features may result in vastly different algorithm performances [Nikolikj et al.(2023), Long et al.(2023)]. This effect might be enhanced in our case of the dynamic behavior of the RL agent we would like to capture. Our work focuses on selecting a subset of available instances using dynamic trajectory-based features in DAC with RL, which can potentially lower the regret over the optimal policy on the test instance.

3 Preliminaries

In this section, we briefly summarize the concepts that form the basis of our method. We start with an introduction to fundamental concepts in RL. We then formally connect the RL problem to DAC using the cMDP framework. We finally describe the instance selection strategy that we employ in our method.

3.1 RL and MDPs

Deep RL deals with sequential decision-making problems, where an *agent* interacts with an *environment*, modeled as an MDP, represented as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \rho, T \rangle$. At each time step t, an agent observes the state $s_t \sim \mathcal{S}$ of the environment and chooses an action $a_t \sim \mathcal{A}$ using a policy $\pi_\theta(a_t \mid s_t)$ – a Deep Neural Network (DNN) with weights θ – to transition into a new state s_{t+1}. The environment transitions are modeled as a function $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and for each transition, the agent receives a reward according to the reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to R$. The MDP is additionally characterized by the initial state distribution $\rho_0(s_0) : \mathcal{S} \to R^+$, and the maximal horizon T.

The agent's objective is to learn an optimal policy $\pi^* \in \Pi$ that maximizes

the expected discounted sum of rewards G:

$$\pi^* \in_{\pi \in \Pi} E_{s_0 \sim \rho}\big[G(\pi, s_0)\big]. \tag{1}$$

Policy gradient methods sutton-nips99a maximize an iterative objective $J(\theta)$ that depends on the gradient $\nabla\theta$ of the policy weights. In this work, we use the well-established policy-gradient algorithm PPO [Schulman et al.(2017)] as the RL agent.

3.2 DAC with RL

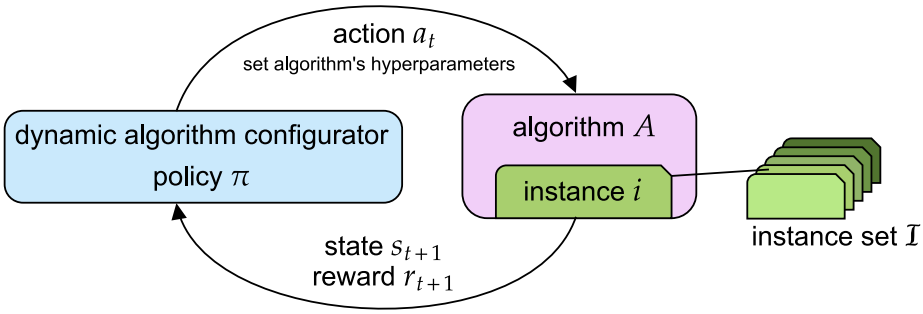

Figure 2: In DAC, we configure an algorithm's hyperparameters dynamically for a given instance set representing the tasks to solve.

DAC [Biedenkapp et al.(2020)] aims to improve a target algorithm's performance through dynamic control of its hyperparameters $\lambda \in \Lambda$. One way to formalize the DAC problem is through Contextual MDPs (cMDPs) [Hallak et al.(2015), Biedenkapp et al.(2020)] extending the MDP presented before: A cMDP is a tuple $\mathcal{M}_{\mathcal{I}} = \langle \mathcal{S}, \mathcal{A}, P_i, R_i, \rho \rangle_{i \in}$, where is a set of instances. In other words, *a cMDP is a collection of MDPs that differ in their transition dynamics and reward functions.* The state $s \in \mathcal{S}$ observed by the DAC agent is the information about an algorithm , such as the progress. The actions $a \in \mathcal{A}$ change the current hyperparameter configuration λ (i.e., $\Lambda = \mathcal{A}$)[1] and the instances are different tasks that need to be solved. A policy π receives the algorithm's state and outputs hyperparameters $\lambda \in \Lambda$, maximizing an instance-specific reward $R_i : \mathcal{S} \times \Lambda \to R$. The implicit transition function $P_i : \mathcal{S} \times \Lambda \to \mathcal{S}$ corresponds to the algorithm behavior.

3.3 Performance Assessment and SELECTOR

A particularly challenging aspect of general algorithm configuration is the selection of appropriate instances to measure performance. To be able to do so,

[1]Besides directly changing the hyperparameter configurations, RL policies could also learn to modify the configuration by, e.g., multiplying a factor to the current hyperparameter values.

we first need an appropriate characterization mechanism for these instances involving a set of meta-features.

The SELECTOR methodology [Cenikj et al.(2022)] selects a representative subset of instances from a large pool of instances with the explicit goal of *maximizing the representativeness* and *minimizing the redundancy* in the selected subset. It involves three steps:

[label=()]Compute a meta-representation of each data instance Construct a similarity graph of the data instances, where the nodes are the instances, and they are connected with an edge if the similarity of their meta-representations exceeds a predefined threshold. Depending on this threshold, the number of edges in the graph varies, as does the number of selected instances. Apply a graph algorithm to select a subset of nodes that are diverse, representative, and non-redundant. This can be accomplished using the Dominating Set (DS) or the Maximal Independent Set (MIS) [Esfahanian(2013), Byskov(2003)] algorithm. The DS algorithm selects a subset of the instances such that every node that is not selected is similar to at least one node in the selected subset. On the other hand, the MIS algorithm selects instances in such a way that it ensures that there is no pair of nodes in the selected subset that is similar to each other.

4 Method

The goal of our study is to improve the generalization of an RL agent in DAC, as measured by the performance of a policy on a test set of target instances. 1 shows the outline of our method. Overall, we use SELECTOR to sample a subset of representative training instances, to which we then allocate more training resources. We fix the total number of times the RL agent interacts with the environment before the subselection and after the subselection to be the same. This means for the same training budget, we train on fewer but more representative instances after subselection.

To enable this workflow, we start by training an RL agent on the train instance set . A key element is using meta-features based on the data from the trajectory generated by the RL agent as it interacts with the algorithm. We do this by evaluating the trained agent on the train instance set and producing rollout trajectories, specifically the actions taken by the agent and the reward received for each action. This data encodes the agent's behavior for each training instance. We feed these meta-feature data for all training instances to SELECTOR, which subselects instances from the train instance set to form the reduced, subselected instance set $\subseteq$. Intuitively, these instances capture the essential aspects of the dynamics observed by the agent during training and should, therefore, enable better generalization. We finally train the RL agent again on the subselected instance set to obtain the final policy, which we can subsequently evaluate on the held-out test set of instances.

Meta-Feature Representations SELECTOR requires the representation of data instances (in our case, episodes from training the RL agent) to be numerical features. In prior work, instance meta-features were obtained via a manual approach [Bischl et al.(2016)], possibly not always reflecting the agent's interaction with the environment. Our approach, however, uses features from the data generated by the agent as it interacts, thus capturing the agent's dynamic behavior. We explore the following representations:

Raw Representations are the raw actions and rewards observed during training. These representations are constructed by simply concatenating the sequence of actions taken by the agent and the corresponding rewards obtained in each iteration.

Catch22 Representation are time-series features extracted from the raw actions and rewards observed during training. These features capture a broad spectrum of time-series characteristics, including the distribution of values in the time series, linear and nonlinear temporal autocorrelation properties, scaling of fluctuations, and other relevant properties. Another advantage to using time-series features is the ability to characterize and compare variable-length episodes. We use the catch22 [Lubba et al.(2019)] library to extract 22 time-series features from the observed sequences of actions and rewards together with mean and standard deviation, resulting in 24 features. Note that we could use any other time-series features.

Both representations (raw and catch22) can also be combined with instance features describing the problem instance and are not directly related to the behavior of the RL agent. An example of such features can be the slope and shift of a sigmoid problem instance.

SELECTOR We execute the SELECTOR methodology using the different aforementioned representations to represent the instances from the training set. We use the Dominating Sets (DS; [Esfahanian(2013)]) and Maximal Independent Set (MIS; [Byskov(2003)]) algorithms with different similarity thresholds, specifically, $0.7, 0.8, 0.9$, and 0.95.

5 Experiments

For evaluating our method, we rely on the benchmark library DACBench [Eimer et al.(2021)], which features DAC benchmarks from different AI domains. We first cover the evaluation protocol, then the DAC benchmarks used, Sigmoid and CMA-ES, and finally, detail the training of the RL agent.

Evaluation Protocol Our overall objective is to assess the generalization performance on the test instance set . Therefore, we evaluate the agent trained on the full, original train instance set and the agent trained on the subselected set $\subseteq$ once again on the test instance set . For an empirical upper limit to performance on the test instance set, we additionally train *Instance-Specific Agents (ISAs)*. Each ISA is an RL agent trained on one instance of the test

instance set and evaluated on that specific instance, serving as a reference. This construction of ISA exploits the notorious property of the RL agents to overfit their training instance: *Each ISA demonstrates the possible reward that an RL agent can accumulate when trained solely on this instance.* In other words, they serve as an empirical performance upper bound that should be hard to achieve for a DAC agent being trained across a variety of training instances. In addition, we also compare to RL agents trained on 5 random subsets of 10% of the train instance set , which is a similar fraction of instances selected by SELECTOR.

We perform experiments on the Sigmoid benchmark, where a Sigmoid curve with varying slope and shift should be approximated, and on CMA-ES, where the step-size σ is adapted. In the following paragraphs, we further explain these benchmarks.

Sigmoid This benchmark challenges DAC agents to approximate a Sigmoid function in different dimensions. It is an artificial white-box benchmark that was proposed to study DAC with full control over the application [Biedenkapp et al.(2020)]. A Sigmoid function is characterized by its shift and slope and has function values between 0 and 1. Actions are discrete and evenly space the interval $[0, 1]$. For example, for an action space of 5 actions, the actions would be $a \in \{0, 0.25., 0.5, 0.75, 1\}$. We approximate Sigmoids in two dimensions, with 5 and 10 actions, respectively. The state features consists of the remaining budget, the shift and slope for each dimension, and the action for each dimension. The difficulty of the problem can be increased by increasing the dimensionality. The training and test instance sets comprise 300 instances of two-dimensional Sigmoids.

CMA-ES CMA-ES (Covariance Matrix Adaption Evolution Strategy) hansen-eda06a is an evolutionary algorithm for continuous black-box problems which can be non-linear and non-convex. In DACBench [Eimer et al.(2021)], the step-size $\sigma \in [0, 10]$ of CMA-ES can be adapted, which is a continuous action space. Others adapt the step-size via a heuristic [Igel et al.(2007), Hansen(2008)] or guided policy search [Shala et al.(2020)]. As a state, the RL agent receives the generation size, the current step-size σ, the remaining optimization budget, as well as the function and instance ID. The reward is the negative minimum function value observed so far since CMA-ES is a minimizer and the RL agent is a maximization algorithm. The train and test instance set comprises ten synthetic blackbox optimization benchmarking (BBOB) functions [Hansen et al.(2020)] – Sphere, Ellipsoidal, Rastrigin, Büche-Rastrigin, Linear Slope, Attractive Sector, Step Ellipsoidal, original and rotated Rosenbrock and Ellipsoidal. All of these functions are either separable or have low or moderate conditioning, except for the last one with high conditioning, $\in R^{10}$. The train set features four instances of each function, and the test set one instance.

Training Details We repeat our training and evaluation pipeline for 10 random seeds. Our training details are as follows: We train a PPO [Schulman et al.(2017)]

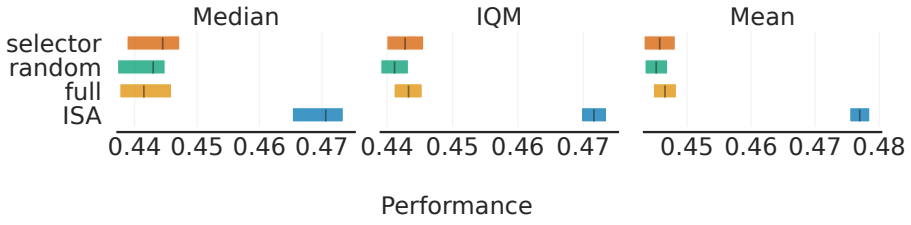

Figure 3: Sigmoid Performance

agent for 10 000 environment steps in Sigmoid, equaling 1 000 episodes, with each episode having a length of 10. For CMA-ES, we train the agent for 1 000 000 steps. However, here, we have variable episode lengths. We evaluate each trained agent with 10 evaluation episodes per instance. Based on the evaluation rollout data, we run SELECTOR 5 times and normalize the agent's performance per instance. We then compute bootstrapped mean, median, and IQM with 5 000 samples using the library rliable [**?**] for the evaluation performance. We additionally use fANOVA [Hutter et al.(2014)] with standard settings to analyze the sensitivity of SELECTOR to its own hyperparameters, namely feature types, the method of selection, the source of features, and the threshold.

Instance representation and selection The chosen benchmark suites encompass training RL in distinct environments: one involving discrete actions (Sigmoid) and the other involving continuous actions (CMA-ES). We employ different representations to depict the behavior of the RL agent. Based on the actions (A) and rewards (R) recorded on evaluation rollouts, we either use the raw (flattened) vectors for fixed-length episodes or the catch22 time-series features for variable-length episodes. We can also concatenate action and reward vectors (RA) and add instance features (I) if applicable.

In both benchmark suites and their respective instance representations, we employ the SELECTOR method (both MIS and DS with similarity thresholds $\in \{0.7, 0.8, 0.9, 0.95\}$ for creating the graph) to choose subsets of instances for retraining the RL agent.

5.1 Results and Discussion

On both benchmarks, Sigmoid and CMA-ES, training on subselected instances from SELECTOR generalizes better to the test instance set than training on the full instance set, see 3 for Sigmoid and 4 for CMA-ES. First of all, this supports our hypothesis that training an DAC agent with RL on a well-constructed subset of instances can be better than simply training on an arbitrary instance set. Secondly, the extraction of trajectory information is sufficiently informative to construct this set.

Interestingly, the performance of the ISA for CMA-ES is worse than the performance of SELECTOR. Initially, our aim was to construct ISA so that

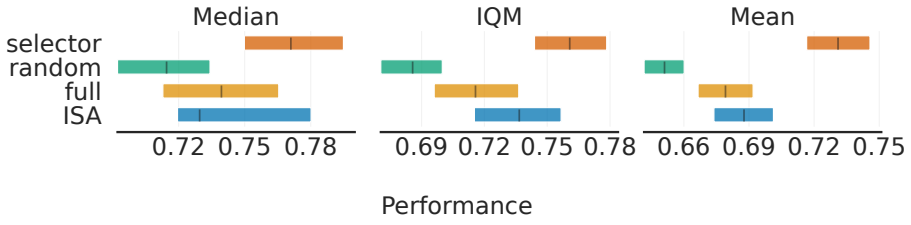

Figure 4: CMA-ES Performance

we get an empirical approximation of a theoretical upper limit; thus our DAC agent on SELECTOR should not be able to outperform ISA. We hypothesize that the diversity in trajectories from multiple instances instead of only one instance allows the optimization process of the RL agent to escape potential local minima in policy space that the ISA agents get stuck in. This corroborates the successful methodology of learning the step size with guided policy search [Shala et al.(2020)], where they guide the optimization and start from a suitable point in the policy space.

Depending on the benchmark we observe different best performing variants of SELECTOR. According to the IQM, SELECTOR with (MIS, Catch22, R, 0.7) for Sigmoid and SELECTOR with (DS, Catch22, R or RA, 0.8) for CMA-ES performed best. So, it is important to study the hyperparameter (HP) sensitivity of our approach. For Sigmoid the type of representation is important, using only actions or combinations with reward yields best results. The other HPs do not have a major impact on Sigmoid. For CMA-ES, the subselection method on the similarity graph (DS or MIS) is the most important HP. Again, representations using actions and rewards together works best. A reasonable robust and general choice would be to use rewards and actions as features sources combined with DS.

The size of the instance set shows strong variation for the threshold of SELECTOR for Sigmoid, but not so much for CMA-ES, as shown in fig:thresholds. Peaking closer into Sigmoid, fig:thresholds (right) indicates that instance features and trajectory features are not very correlated. A small instance set with a high threshold induces a dense graph, i.e. instances are pretty similar in terms of instance features which does not necessarily mean trajectory features are similar.

In addition, the instances selected by SELECTOR evenly cover the full instance set, capturing the diversity that is most apparent for the second dimension (6). For CMA-ES, often only one instance of the BBOB functions 7, 8, 9 is selected. These functions have a more complex local structure compared to the first functions but still are similar in global structure, rendering them suitable to represent the instance set.

Limitations and Future Work One limitation of our method is that it requires training the RL agent twice as well as training SELECTOR. We plan to

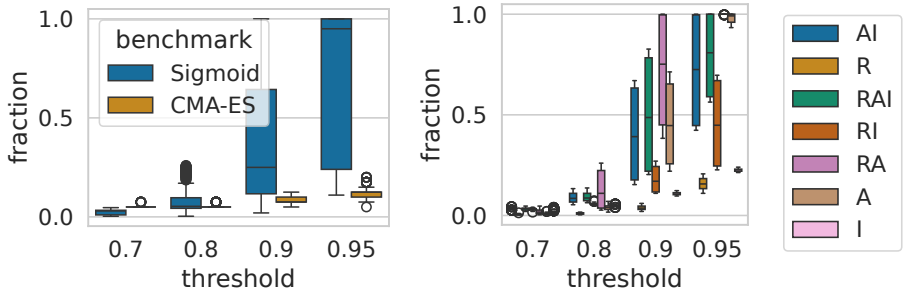

Figure 5: (Left) Size of subselected instance set for different SELECTOR thresholds per benchmark. (Right) Size of subselected instance sets for Sigmoid for different representations.

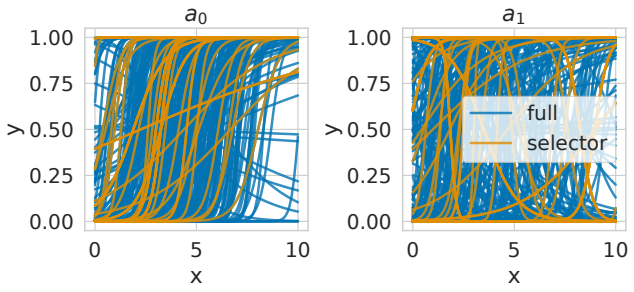

Figure 6: Selected instances by SELECTOR for Sigmoid. A small but diverse set of instances is selected.

investigate the benefits that can be potentially gained from *early-stopping*, such as only training the agent for half of the training budget. Potentially, the benefits of SELECTOR could be attained in the same training budget as a standard baseline agent. We additionally plan to meta-learn well-performing presets for SELECTOR to create a truly end-to-end training and selection pipeline. Lastly, we would like to approach handling instances also at the level of the RL algorithm: For benchmarks like CMA-ES, we have problems with different reward scales, potentially hindering learning, which we could normalize per instance.

6 Conclusion

In this work, we demonstrate the potential of instance selection in enhancing the generalization capabilities of RL for DAC. We first train an RL agent on a train set of instances and then generate rollout trajectories by evaluating the trained agent on the same set of instances. Since these trajectories capture the agent's behavior on the training instances, we use this data to create time-

series features that capture the *dynamic* behavior of the RL policy. We then subselect a representative set of training instances and retrain the RL agent on these instances to obtain better generalization performance on unseen new instances. By meticulously selecting representative instances for training, we not only address the challenge of overrepresentation in training instances but also demonstrate superior performance to agents trained on specific instances on CMA-ES. Our approach marks a step forward in the application of RL to DAC, offering a scalable solution that can adapt to the ever-changing complexities of hyperparameter control using RL.

7 Acknowledgements

Funding in direct support of this work: Slovenian Research Agency: research core funding No. P2-0098, young researcher grants No. PR-12393 to GC and No. PR-12897 to AN, project No. J2-4460, and a bilateral project between Slovenia and Germany grant No. BI-DE/23-24-003. DAAD: 57654659.

References

[Benjamins et al.(2023)] C. Benjamins, T. Eimer, F. Schubert, A. Mohan, S. Döhler, A. Biedenkapp, B. Rosenhan, F. Hutter, and M. Lindauer. 2023. Contextualize Me – The Case for Context in Reinforcement Learning. *Transactions on Machine Learning Research* (2023).

[Biedenkapp et al.(2020)] A. Biedenkapp, H. F. Bozkurt, T. Eimer, F. Hutter, and M. Lindauer. 2020. Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. In *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*, J. Lang, G. De Giacomo, B. Dilkina, and M. Milano (Eds.). 427–434.

[Bischl et al.(2016)] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Frechétte, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. 2016. ASlib: A Benchmark Library for Algorithm Selection. *Artificial Intelligence* 237 (2016), 41–58.

[Byskov(2003)] Jens M. Byskov. 2003. Algorithms for k-colouring and finding maximal independent sets. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*. 456–457.

[Cenikj et al.(2022)] G. Cenikj, R. Dieter Lang, A. Engelbrecht, C. Doerr, P. Korosec, and T. Eftimov. 2022. SELECTOR: selecting a representative benchmark suite for reproducible statistical comparison. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'22)*, J. Fieldsend (Ed.). ACM Press.

[Eftimov et al.(2022)] T. Eftimov, G. Petelin, G. Cenikj, A. Kostovska, G. Ispirova, P. Korošec, and J. Bogatinovski. 2022. Less is more: Selecting the right benchmarking set of data for time series classification. *Expert Systems with Applications* 198 (2022), 116871.

[Eimer et al.(2021)] T. Eimer, A. Biedenkapp, M. Reimer, S. Adriaensen, F. Hutter, and M. Lindauer. 2021. DACBench: A Benchmark Library for Dynamic Algorithm Configuration. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, Z. Zhou (Ed.). ijcai.org, 1668–1674.

[Esfahanian(2013)] A. Esfahanian. 2013. Connectivity algorithms. *Topics in structural graph theory* (2013), 268–281.

[Grünewälder et al.(2012)] S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton. 2012. Modelling transition dynamics in MDPs with RKHS embeddings. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, J. Langford and J. Pineau (Eds.). Omnipress.

[Guo et al.(2022)] J. Guo, M. Gong, and D. Tao. 2022. A Relational Intervention Approach for Unsupervised Dynamics Generalization in Model-Based Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*. Published online: iclr.cc.

[Hallak et al.(2015)] A. Hallak, D. Di Castro, and S. Mannor. 2015. Contextual Markov Decision Processes. *arXiv:1502.02259 [stat.ML]* (2015).

[Han and Wu(2022)] X. Han and F. Wu. 2022. Meta Reinforcement Learning with Successor Feature Based Context. *arXiv preprint arXiv:2207.14723* (2022).

[Hansen(2008)] N. Hansen. 2008. CMA-ES with Two-Point Step-Size Adaptation. *CoRR* (2008). http://arxiv.org/abs/0805.0231

[Hansen et al.(2020)] N. Hansen, A. Auger, R. Ros, O. Mersman, T. Tušar, and D. Brockhoff. 2020. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* (2020).

[Hutter et al.(2014)] F. Hutter, H. Hoos, and K. Leyton-Brown. 2014. An Efficient Approach for Assessing Hyperparameter Importance. In *Proceedings of the 31th International Conference on Machine Learning, (ICML'14)*, E. Xing and T. Jebara (Eds.). Omnipress, 754–762.

[Igel et al.(2007)] C. Igel, N. Hansen, and S. Roth. 2007. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation* 15 (2007), 1–28.

[Ispirova et al.(2024)] G. Ispirova, T. Eftimov, S. Džeroski, and B. Seljak. 2024. MsGEN: Measuring generalization of nutrient value prediction across different recipe datasets. *Expert Systems with Applications* 237 (2024), 121507.

[Justesen et al.(2022)] N. Justesen, R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. 2022. Illuminating generalization in deep reinforcement learning through procedural level generation. (2022).

[Kirk et al.(2023)] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. 2023. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research (JAIR)* 76 (2023), 201–264.

[Lagoudakis and Littman(2000)] M. Lagoudakis and M. Littman. 2000. Algorithm Selection using Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, P. Langley (Ed.). Morgan Kaufmann Publishers, 511–518.

[Lagoudakis and Littman(2001)] M. Lagoudakis and M. Littman. 2001. Learning to Select Branching Rules in the DPLL Procedure for Satisfiability. *Electronic Notes in Discrete Mathematics* 9 (2001), 344–359.

[Long et al.(2023)] F. Long, D. Vermetten, B. van Stein, and A. Kononova. 2023. BBOB Instance Analysis: Landscape Properties and Algorithm Performance Across Problem Instances. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 380–395.

[Lubba et al.(2019)] C. Lubba, S. Sethi, P. Knaute, S. Schultz, B. Fulcher, and N. Jones. 2019. catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery* 33, 6 (2019), 1821–1852. https://doi.org/10.1007/s10618-019-00647-x

[Mohan et al.(2024)] A. Mohan, A. Zhang, and M. Lindauer. 2024. Structure in Deep Reinforcement Learning: A Survey and Open Problems. *Journal of Artificial Intelligence Research* 79 (2024).

[Ng(2022)] A. Ng. 2022. Unbiggen ai. *IEEE Spectrum* 9 (2022).

[Nikolikj et al.(2023)] A. Nikolikj, M. Pluháček, C. Doerr, P. Korošec, and T. Eftimov. 2023. Sensitivity Analysis of RF+clust for Leave-One-Problem-Out Performance Prediction. In *2023 IEEE Congress on Evolutionary Computation (CEC)*. 1–8.

[Pettinger and Everson(2002)] J. Pettinger and R. Everson. 2002. Controlling genetic algorithms with reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)*, W. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz,

J. Miller, E. Burke, and N. Jonoska (Eds.). Morgan Kaufmann Publishers, 692–692.

[Schulman et al.(2017)] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs.LG]* (2017).

[Seo et al.(2020)] Y. Seo, K. Lee, I. Gilaberte, T. Kurutach, J. Shin, and P. Abbeel. 2020. Trajectory-wise Multiple Choice Learning for Dynamics Generalization in Reinforcement Learning. In *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin (Eds.). Curran Associates.

[Shala et al.(2020)] G. Shala, A. Biedenkapp, N. Awad, S. Adriaensen, M. Lindauer, and F. Hutter. 2020. Learning Step-Size Adaptation in CMA-ES. In *Proceedings of the Sixteenth International Conference on Parallel Problem Solving from Nature (PPSN'20) (Lecture Notes in Computer Science)*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann (Eds.). Springer, 691–706.

[Sharma et al.(2019)] M. Sharma, A. Komninos, M. López-Ibáñez, and D. Kazakov. 2019. Deep reinforcement learning based parameter control in differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, M. López-Ibáñez (Ed.). ACM Press, 709–717.

[Shi et al.(2022)] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, and C. Wu. 2022. Self-Supervised Discovering of Interpretable Features for Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2022), 2712–2724.

[Speck et al.(2021)] D. Speck, A. Biedenkapp, F. Hutter, R. Mattmüller, and M. Lindauer. 2021. Learning Heuristic Selection with Dynamic Algorithm Configuration. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS'21)*, H. H. Zhuo, Q. Yang, M. Do, R. Goldman, S. Biundo, and M. Katz (Eds.). AAAI.

[Zhang et al.(2018)] C. Zhang, O. Vinyals, R. Munos, and S. Bengio. 2018. A Study on Overfitting in Deep Reinforcement Learning. *arXiv preprint arXiv:1804.06893* (2018).