A Performance Analysis of Lexicase-Based and Traditional Selection Methods in GP for Symbolic Regression

ALINA GEIGER, DOMINIK SOBANIA, and FRANZ ROTHLAUF, Johannes Gutenberg University, Mainz, Germany

In recent years, several new lexicase-based selection variants have emerged due to the success of standard lexicase selection in various application domains. For symbolic regression problems, variants that use an ϵ-threshold or batches of training cases, among others, have led to performance improvements. Lately, especially variants that combine lexicase selection and down-sampling strategies have received a lot of attention. This article evaluates the most relevant lexicase-based selection methods as well as traditional selection methods in combination with different down-sampling strategies on a wide range of symbolic regression problems. In contrast to most work, we not only compare the methods over a given evaluation budget, but also over a given time budget as time is usually limited in practice. We find that for a given evaluation budget, ϵ-lexicase selection in combination with a down-sampling strategy outperforms all other methods. If the given running time is very short, lexicase variants using batches of training cases perform best. Further, we find that the combination of tournament selection with informed down-sampling performs well in all studied settings.

CCS Concepts: • **Computing methodologies** → **Genetic programming**; **Supervised learning by regression**;

Additional Key Words and Phrases: Symbolic Regression, Genetic Programming, Lexicase Selection, Down-sampling

ACM Reference format:
Alina Geiger, Dominik Sobania, and Franz Rothlauf. 2026. A Performance Analysis of Lexicase-Based and Traditional Selection Methods in GP for Symbolic Regression. *ACM Trans. Evol. Learn. Optim.* 6, 1, Article 5 (February 2026), 40 pages.
https://doi.org/10.1145/3725860

1 Introduction

Symbolic regression searches for a mathematical expression that best describes a given problem [Poli et al., 2008]. Applications of symbolic regression range from finance [Chen, 2012] and medicine [La Cava et al., 2023] to materials science [Hernandez et al., 2019a]. **Genetic Programming (GP)** [Koza, 1992] is an evolutionary computation technique that has often been used successfully to solve even complex symbolic regression tasks [La Cava et al., 2021]. In GP, a population of individuals evolves in an evolutionary process guided by selection and modified through variation operators.

Authors' Contact Information: Alina Geiger (corresponding author), Johannes Gutenberg University, Mainz, Germany; e-mail: geiger@uni-mainz.de; Dominik Sobania, Johannes Gutenberg University, Mainz, Germany; e-mail: dsobania@uni-mainz.de; Franz Rothlauf, Johannes Gutenberg University, Mainz, Germany; e-mail: rothlauf@uni-mainz.de.

ACM 2688-3007/2026/2-ART5
https://doi.org/10.1145/3725860

Recent work has shown that especially the choice of the selection method has a major impact on the solution quality [Helmuth and Abdelhady, 2020].

Traditional selection methods like tournament selection or fitness-proportionate selection evaluate the quality of individuals based on an aggregated fitness score leading to a loss of information about the structure of the training data [Krawiec and O'Reilly, 2014]. Therefore, lexicase selection [Helmuth et al., 2014; Spector, 2012] has been proposed that uses the error values of individuals on each training case. This allows lexicase selection to select specialists that perform particularly well for certain parts of the problem, which has been found to be beneficial for search toward better solutions [Helmuth et al., 2019; Helmuth et al., 2020].

Due to the successful application of lexicase selection in many problem domains [Aenugu and Spector, 2019; Helmuth and Spector, 2015; La Cava et al., 2016; Moore and Stanton, 2017], several variants of lexicase selection have been proposed. In the domain of symbolic regression, ϵ-lexicase selection outperformed standard lexicase selection and tournament selection on the considered problems [La Cava et al., 2016; La Cava et al., 2019]. Others suggested to use batches of training cases instead of single training cases to evaluate the quality of individuals [Aenugu and Spector, 2019; De Melo et al., 2019; Sobania and Rothlauf, 2022].

Recently, especially variants combining lexicase selection with down-sampling strategies received much attention because it led to higher problem-solving success in different applications [Boldi et al., 2024b; Ferguson et al., 2020; Geiger et al., 2023; Hernandez et al., 2019b]. Down-sampling uses only a subset of the training cases in each generation to evaluate the quality of the individuals. For a fixed evaluation budget, the saved evaluations can be allocated to a longer search, which allows the exploration of more individuals. The simplest down-sampling strategy is to randomly sample a percentage of the training cases in each generation [Ferguson et al., 2020; Hernandez et al., 2019b]. However, this might cause the exclusion of important training cases for several generations [Boldi et al., 2024b]. Therefore, Boldi et al. [2024b] proposed informed down-sampling as a strategy to create more diverse subsets.

In our previous conference paper [Geiger et al., 2024], we analyzed and compared several relevant lexicase-based selection methods in combination with random down-sampling. We used tournament selection as baseline and also studied the combination of lexicase variants with batches of training cases. The paper at hand builds upon this work and presents a more comprehensive and more detailed comparison of lexicase-based selection methods. In addition to tournament selection, we include fitness-proportionate selection as a baseline. Geiger et al. [2024] analyzed selection methods only in combination with random down-sampling. The paper at hand also adds selection variants using informed down-sampling. We compare the methods not only for a given evaluation budget (as is common in the literature) but also for a given time budget as the selection methods differ in terms of time complexity. For the comparison, we extend the problem set to 26 synthetic and real-world benchmark problems. Our goal is to provide researchers as well as practitioners with a comprehensive guide for choosing the appropriate selection method.

We find that ϵ-lexicase selection in combination with random or informed down-sampling performs best for a given evaluation budget. Further, we observe that the relative performance of each selection method depends on the given running time. For example, for a time budget of 24 h, the best performing methods are tournament selection in combination with informed down-sampling and ϵ-lexicase selection in combination with random down-sampling. However, if the running time is very short (1 h or less), batch-tournament selection and batch-ϵ-lexicase selection perform best. Additionally, the detailed analysis for each problem shows that while there are general trends, the relative performance of the methods varies between problems.

Section 2 provides a brief overview of the related work. In Section 3, we describe the selection methods analyzed in this study, followed by a description of the applied down-sampling

strategies in Section 4. Section 5 presents our experimental setting and the benchmark problems. In Section 6, we present our results, followed by the conclusions in Section 7.

2 Related Work

We briefly discuss prior work on lexicase selection. More detailed descriptions of the lexicase-based selection methods and down-sampling strategies are given in Sections 3 and 4.

Traditional selection methods (e.g., tournament selection and fitness-proportionate selection) evaluate the quality of individuals based on an aggregated fitness value. This leads to a loss of information about the structure of the training data [Krawiec and O'Reilly, 2014]. Lexicase selection [Helmuth et al., 2014; Spector, 2012] has been proposed as an alternative considering the errors of individuals on each training case separately. Lexicase selection is able to select specialists as parents, meaning individuals that solve some part of a problem better than others while performing worse on average [Helmuth et al., 2019; Helmuth et al., 2020; Pantridge et al., 2018]. In addition, prior work has found that lexicase selection maintains a higher population diversity compared to tournament selection [Helmuth et al., 2016a,b].

For continuous problems, standard lexicase selection is not able to achieve the same performance improvements due to the selection of strictly elite individuals on training cases [La Cava et al., 2016]. Therefore, lexicase variants with a relaxed pass condition have been proposed [La Cava et al., 2016; Spector et al., 2018], with ϵ-lexicase selection [La Cava et al., 2016; La Cava et al., 2019] being the best known. Additionally, variants using batches of training cases have been introduced [Aenugu and Spector, 2019; De Melo et al., 2019; Sobania and Rothlauf, 2022]. It has been found that the use of batches can increase the generalizability of the found solution [Aenugu and Spector, 2019; Sobania and Rothlauf, 2022]. Batch-tournament selection [De Melo et al., 2019] uses batches in combination with tournament selection to improve the efficiency of lexicase selection. Another idea to improve the run time of lexicase selection is to combine lexicase selection and weighted shuffle with partial evaluation [Ding et al., 2022a,b]. Plexicase selection [Ding et al., 2023] improves the run time of lexicase selection by sampling individuals from a probability distribution instead of doing the actual selection procedure. Ni et al. [2024] proposed DALex to improve the run time by performing all calculations as matrix multiplications. In other studies, the combination of lexicase selection and novelty search has been explored to prevent premature convergence [Jundt and Helmuth, 2019; Kelly et al., 2019].

Recently, the combination of lexicase variants with down-sampling strategies has been found to lead to higher performance [Boldi et al., 2024b; Ferguson et al., 2020; Geiger et al., 2023; Hernandez et al., 2019b]. Therefore, the influence of down-sampling has been analyzed in several studies [Boldi et al., 2023b; Helmuth and Spector, 2020; Helmuth and Spector, 2021; Hernandez et al., 2022; Schweim et al., 2022]. Down-sampling reduces the number of evaluations per generations by sampling a subset of training cases from the training set allowing the evaluation of more individuals with the same evaluation budget [Helmuth and Spector, 2020]. While random down-sampling [Ferguson et al., 2020; Hernandez et al., 2019b] randomly creates subsets, informed down-sampling [Boldi et al., 2024b] creates subsets that include distinct training cases. It has been found that informed down-sampling outperforms random down-sampling in combination with lexicase selection on program synthesis problems [Boldi et al., 2024b]. Therefore, informed down-sampling has been further studied in combination with tournament and fitness-proportionate selection for program synthesis problems and synthetic regression problems with an exact solution [Boldi et al., 2024a; Boldi et al., 2023a]. However, to our knowledge, there is no study that analyzes the influence of informed down-sampling for continuous symbolic regression problems.

Lexicase selection and its variants have been successfully applied and analyzed in several domains, like program synthesis [Helmuth and Abdelhady, 2020; Helmuth and Spector, 2015; Sobania and

Rothlauf, 2021; Sobania et al., 2023], symbolic regression [Geiger et al., 2023; La Cava et al., 2016; La Cava et al., 2019], evolutionary robotics [Moore and Stanton, 2017; Moore and Stanton, 2018], rule-based learning systems [Aenugu and Spector, 2019; Wagner and Stein, 2021], and deep learning [Ding and Spector, 2021].

Prior work analyzed and compared several lexicase variants for program synthesis problems [Helmuth and Abdelhady, 2020]. However, to our knowledge, a study that compares lexicase-based selection methods in combination with different down-sampling strategies, including informed down-sampling, on a wide range of symbolic regression problems is missing so far.

3 Selection Methods for Symbolic Regression Problems

We provide a detailed description of tournament selection, fitness-proportionate selection, lexicase selection, as well as the lexicase-based variants studied in this work in the context of symbolic regression.

3.1 Tournament Selection

With tournament selection, k individuals are randomly chosen to participate in a tournament. The participant with the best fitness wins the tournament and is selected as a parent [Poli et al., 2008]. Fitness is measured as an aggregated value, for example, the **Mean Squared Error (MSE)**

$$\text{MSE}(T) = \frac{1}{|T|} \sum_{t \in T} (y_t - \hat{y}_t)^2, \tag{1}$$

where $\hat{y}_t$ is the predicted output of an individual and y_t the desired output for all training cases $t \in T$.

3.2 Fitness-Proportionate Selection

With fitness-proportionate selection (also known as roulette wheel selection), an individual i is selected as a parent with the probability of

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \tag{2}$$

where f_i is the fitness of individual i and N is the number of individuals in the population [Banzhaf et al., 1998]. This means that individuals with higher fitness have a higher probability of being selected than individuals with lower fitness. As with tournament selection, fitness is measured in terms of an aggregated value.

3.3 Lexicase Selection

In contrast to tournament selection and fitness-proportionate selection, lexicase selection [Spector, 2012] considers the error values e_t on each training case $t \in T$ separately rather than comparing individuals based on an aggregated fitness value. Algorithm 1 shows the procedure of a single selection event with lexicase selection with population P, the set of training cases T, the error e_t of an individual $i \in P$ for training case $t \in T$, and the minimal error e_t^* on cases $t \in T$.

With lexicase selection, all individuals $i \in P$ are considered as candidates for selection (line 1), and all training cases $t \in T$ are randomly ordered (line 2). Lexicase selection iterates through all training cases always keeping only the individuals in the candidate pool C with the lowest error e_t^* on the current case t (line 5). This is repeated until there is only one candidate left or there are no more training cases. In this case, one individual is randomly chosen from the remaining candidate pool C (line 8) [Helmuth et al., 2014; Spector, 2012].

Algorithm 1: Lexicase Selection

1: C := all individuals $i \in P$
2: T' := randomly shuffled T
3: **while** $|T'| > 0$ and $|C| > 1$ **do**
4: t := first case of T'
5: C := all candidates with $e_t = e_t^*$
6: **remove** t from T'
7: **end while**
8: **return** random choice from C

3.4 ϵ-Lexicase Selection

For continuous problems, the pass condition of standard lexicase selection that only candidates with the lowest error remain in the candidate pool (Algorithm 1, line 5) is too strict. Therefore, ϵ-lexicase selection has been proposed by La Cava et al. [2016, 2019] for solving symbolic regression problems.

ϵ-lexicase selection modifies the pass condition of standard lexicase selection in such a way that all candidates with an error $e_t <= e_t^* + \epsilon_t$ remain in the candidate pool. ϵ_t is calculated for a training case t using the median absolute deviation [Pham-Gia and Hung, 2001]

$$\epsilon_t = \text{median}(|\mathbf{e}_t - \text{median}(\mathbf{e}_t)|), \tag{3}$$

where $\mathbf{e}_t$ is a vector of all errors on the current case t across the candidate pool C [La Cava et al., 2019].

3.5 ϵ-Plexicase Selection

Ding et al. [2023] proposed plexicase selection as a more efficient alternative to lexicase selection. The idea is to reduce the run time of lexicase selection by sampling individuals from a probability distribution instead of doing the actual selections. However, Dolson [2023] has shown that the problem of calculating the actual selection probabilities using lexicase selection is $\mathcal{NP}$-hard. Therefore, Ding et al. [2023] only approximate the probability of individuals being selected with lexicase selection by finding the Pareto set boundaries through pairwise comparisons of individuals and assigning selection probabilities to each individual based on these comparisons. In order to further improve the solution quality, Ding et al. [2023] introduce a hyperparameter α to manipulate the generated probability distribution. For symbolic regression problems, an ϵ-threshold like in ϵ-lexicase is used to calculate ϵ-relaxed Pareto set boundaries. They call this variant ϵ-plexicase selection.

3.6 Batch-Tournament Selection

The problem-solving performance of lexicase selection is superior to tournament selection [Helmuth et al., 2014; Spector, 2012]. However, lexicase selection is computationally expensive compared to tournament selection [La Cava et al., 2019]. Therefore, De Melo et al. [2019] proposed batch-tournament selection as well as the variant **Batch Tournament Selection Shuffled (BTSS)**, which shuffles the cases in addition to grouping them in batches. Both approaches combine the good performance of lexicase selection with the low computational effort of tournament selection.

For BTSS, the training cases are randomly combined to batches of size b and the fitness values of the individuals are derived from their performance on each batch (e.g., in terms of the MSE). For each selection event, k individuals participate in a tournament. The individual with the lowest MSE on the current batch is selected as a parent. For each selection event, another batch is used to

compare the quality of the individuals. If there are less batches than parent selection events, the batches are used multiple times.

3.7 Batch-ϵ-Lexicase Selection

Using lexicase selection, the outcome of a selection event highly depends on the ordering of training cases [Aenugu and Spector, 2019]. Therefore, lexicase variants using batches of training cases instead of single training cases to compare the performance of individuals in the lexicase selection process have been proposed [Aenugu and Spector, 2019; Sobania and Rothlauf, 2022]. It has been found that the generalization ability of the found solutions is better using batch-lexicase selection compared to standard lexicase selection [Aenugu and Spector, 2019; Sobania and Rothlauf, 2022]. For continuous-valued problems, Geiger et al. [2024] proposed Batch-ϵ-lexicase selection to combine batches with ϵ-lexicase selection. In each generation, the training cases are randomly combined to batches of size b and the fitness values of the individuals are calculated on each batch (e.g., in terms of the MSE). The selection is then performed with ϵ-lexicase selection.

4 Down-Sampling Strategies

We describe the down-sampling strategies, namely random down-sampling and informed down-sampling.

4.1 Random Down-Sampling

In application domains, like program synthesis [Ferguson et al., 2020; Helmuth and Spector, 2021; Hernandez et al., 2019b] and symbolic regression [Geiger et al., 2023], it has been found that the combination of lexicase-based selection methods and random down-sampling [Gonçalves et al., 2012] improves the problem-solving success. Random down-sampling means that in each generation a random subset of training cases is used to evaluate the performance of the individuals. Therefore, the saved fitness evaluations in each generation can be used to search for more generations or to increase the population size. This is beneficial as more individuals can be explored with the same evaluation budget [Helmuth and Spector, 2020]. Since a different random subset of training cases is used in each generation, the population is likely to be evaluated on a large proportion of the training cases over a few generations [Hernandez et al., 2019b].

The down-sampling rate is defined by a parameter d. For example, if $d = 0.1$, only 10% of the training cases are used in each generation, meaning that the search can be performed 10 times longer or the population size can be increased by a factor of 10.

4.2 Informed Down-Sampling

The random creation of subsets of training cases has the problem that important cases might be excluded for several generations while training cases with redundant information might end up in the same subset [Boldi et al., 2024b; Ferguson et al., 2020]. Therefore, Boldi et al. [2024b] proposed informed down-sampling to create subsets consisting of more diverse training cases. The authors define synonymous cases as cases that are solved by the same individuals in a population. If two cases are solved by different individuals, this indicates that the cases measure a different functionality. Hence, it would be beneficial to have both cases in a subset.

Boldi et al. [2024b] define the distance between two cases as the Hamming distance between their two solve vectors (a solve vector contains the output of each individual for that case). Then, they use the Farthest First Traversal Algorithm [Hochbaum and Shmoys, 1985] to select a subset of training cases with a large pairwise distance. However, evaluating all individuals on all training cases in each generation to create informed subsets would mean that the benefit of down-sampling is lost because there are no saved fitness evaluations. Therefore, Boldi et al. [2024b] propose to

Table 1. Parameter Setting of Our GP Approach

Parameter	Value
Population size	500
Primitive set	{x, ERC, +, −, ∗, AQ, sin, cos, neg}
ERC values	{−1, 0, 1}
Initialization method	Ramped half-and-half
Maximum tree depth	17
Crossover probability	80%
Mutation probability	5%
Runs	30

sample a fraction s of the parents that are evaluated on all training cases in order to calculate the distance between training cases. In addition, the distance calculation is only performed every g generations to further reduce the number of fitness evaluations required to create the informed subsets.

Since Boldi et al. [2024b] study informed down-sampling for program synthesis problems, solve vectors are binary. In the domain of symbolic regression, the solve vectors contain continuous errors values. Therefore, we calculate the distances between two cases in this work using Euclidean distances instead of Hamming distances.

5 Experimental Setting

We present our experimental setup, followed by a description of the used benchmark problems.

5.1 Setup

All experiments were implemented within the DEAP framework [Fortin et al., 2012] (version 1.4.1). Table 1 summarizes the parameter values of our GP setting, which is in line with previous work [Geiger et al., 2023; Geiger et al., 2024].

The primitive set consists of all input features $\mathbf{x}$, an **Ephemeral Random Constant (ERC)** where ERC $\in \{-1, 0, 1\}$, and the arithmetic functions addition, subtraction, multiplication, **Analytic Quotient (AQ)** [Ni et al., 2013], sine, cosine, and negative. The population is initialized using ramped half-and-half with tree depths between 0 and 4 and a population size of 500. In each generation, crossover is applied with a probability of 80% and mutation with a probability of 5%. We restrict the maximum tree depth to 17 [Koza, 1992]. Each configuration is repeated 30 times.

Usually, selection methods are compared for a given evaluation budget, which is determined by the number of training cases, the population size, and the number of generations [Geiger et al., 2023; Helmuth and Spector, 2020]. If down-sampling is applied, we allocate the saved evaluations to a longer search. Table 2 shows the number of generations given to each run depending on the down-sampling strategy applied.

If there is no down-sampling used ($d = 1.0$), we set the number of generations to $G = 100$. For random down-sampling, we set a down-sampling rate of $d = 0.1$ as suggested by Geiger et al. [2023], meaning that we can search for $\hat{G} = 1,000$ generations with the same evaluation budget. For runs with informed down-sampling, the down-sampling rate is set to $d = 0.1$ as well. The parent sampling rate is set to $s = 0.01$ and the distance calculation scheduling parameter is set to $g = 10$, as suggested by Boldi et al. [2024b]. This means that every 10 generations, 5 individuals are evaluated on all training cases to calculate the distance between the cases. The generational limit

Table 2. Generational Limit for Runs without Down-Sampling, with Random Down-Sampling, and with Informed Down-Sampling

Down-Sampling Strategy	**# Generations**
No down-sampling ($d = 1.0$)	100
Random down-sampling ($d = 0.1$)	1,000
Informed down-sampling ($d = 0.1, s = 0.01, g = 10$)	991

Table 3. Parameter Settings of the Selection Methods

Selection Method	**Parameter**	**Value**
Tournament selection	Tournament size k	5
ϵ-Plexicase selection	Alpha α	1.0
Batch-ϵ-lexicase selection	Batch size b	$\{0.05, 0.075, 0.1\}$
Batch-Tournament selection	Tournament size k	64
	Batch size b	$\{0.05, 0.075, 0.1\}$

for informed down-sampling is calculated as follows:

$$\hat{G} = \frac{G}{d + \frac{s(1-d)}{g}} \tag{4}$$

where G is the number of generations for runs without down-sampling, d is the down-sampling rate, s the parent sampling rate, and g the distance calculation scheduling parameter [Boldi et al., 2024b]. In our case, this leads to limit of $\hat{G} = 991$ for runs using informed down-sampling.

For each problem, we randomly split all cases into 70% training cases, 15% validation cases, and 15% test cases. The training cases are used to evaluate the fitness of individuals during the evolutionary process to perform the selection. In contrast, the validation cases are used to select a final solution. Since the validation cases were not used for the fitness calculation, we avoid selecting a final solution that is overfitted to the training cases. In detail, the individual with the lowest MSE on the validation cases is stored as the current best solution in the Hall of Fame in each generation.[1] After a run, the final solution is evaluated on the unseen test cases in terms of the MSE.

The calculation of the fitness of individuals during the search depends on the selection method. As tournament selection and fitness-proportionate selection evaluate individuals based on an aggregated value, we use the MSE as the fitness measure. For fitness-proportionate selection, the selection probabilities of individuals are assigned based on the inverse of their fitness (so individuals with a lower MSE have a higher probability of being selected).[2] For ϵ-lexicase selection and ϵ-plexicase selection, it is necessary to evaluate the performance of individuals on each training case separately. Therefore, we calculate for each individual the squared error on each training case. For batch-tournament selection and batch-ϵ-lexicase selection, the performance of individuals is measured in terms of the MSE on each batch of training cases.

The parameter settings of the selection methods are shown in Table 3. They are carefully chosen according to the recommendations from the literature.

[1]Due to the use of a validation fitness, individuals in the Hall of Fame are compared based on an aggregated fitness value. This ensures that the final solution performs well overall.

[2]In case an individual has a fitness of zero, $\epsilon = 1e - 10$ is added to the error values in the population to avoid zero divisions. This only happened during a few runs for problem 523_analcatdata_neavote.

Table 4. All Benchmark Problems Used to Evaluate the Selection Methods with the Number of Instances, the Number of Features, and the Problem Type

Problem	# Instances	# Features	Type
1027_ESL	488	4	Real-world
1028_SWD	1,000	10	Real-world
1030_ERA	1,000	4	Real-world
207_autoPrice	159	15	Real-world
230_machine_cpu	209	6	Real-world
4544_GeographicalOriginalofMusic	1,059	117	Real-world
505_tecator	240	124	Real-world
519_vinnie	380	2	Real-world
522_pm10	500	7	Real-world
523_analcatdata_neavote	100	2	Real-world
560_bodyfat	252	14	Real-world
581_fri_c3_500_25	500	25	Synthetic
589_fri_c2_1000_25	1,000	25	Synthetic
591_fri_c1_100_10	100	10	Synthetic
606_fri_c2_1000_10	1,000	10	Synthetic
607_fri_c4_1000_50	1,000	50	Synthetic
615_fri_c4_250_10	250	10	Synthetic
617_fri_c3_500_5	500	5	Synthetic
621_fri_c0_100_10	100	10	Synthetic
623_fri_c4_1000_10	1,000	10	Synthetic
624_fri_c0_100_5	100	5	Synthetic
641_fri_c1_500_10	500	10	Synthetic
647_fri_c1_250_10	250	10	Synthetic
654_fri_c0_500_10	500	10	Synthetic
665_sleuth_case2002	147	6	Real-world
666_rmftsa_ladata	508	10	Real-world

For tournament selection, we set a tournament size of $k = 5$ [Fang and Li, 2010]. The parameter of ϵ-plexicase selection is set to $\alpha = 1.0$, as suggested by Ding et al. [2023]. For batch-ϵ-lexicase selection, we study batch sizes $b \in \{0.05, 0.075, 0.1\}$ [Aenugu and Spector, 2019], which means that a single batch consists of 5%, 7.5%, or 10% of the training cases. For batch-tournament selection, we set a tournament size of $k = 64$ [De Melo et al., 2019] and we study batch sizes $b \in \{0.05, 0.075, 0.1\}$.

In addition to runs given a fixed evaluation budget, we also study the performance for a given running time as recommended in the literature [Ravber et al., 2022]. We measure time in terms of wall clock time. In each generation, we track the performance of the current best individual on the test cases, the median tree size across the population, and the time. We conducted the experiments on a high-performance computing cluster using Intel 2630v4 2,20GHz CPUs. All processes were executed single threaded. Please note that the comparison of methods over time always depends on the implementation. Here, all methods have been implemented within the same framework, and all experiments run on the same hardware to minimize this limitation.

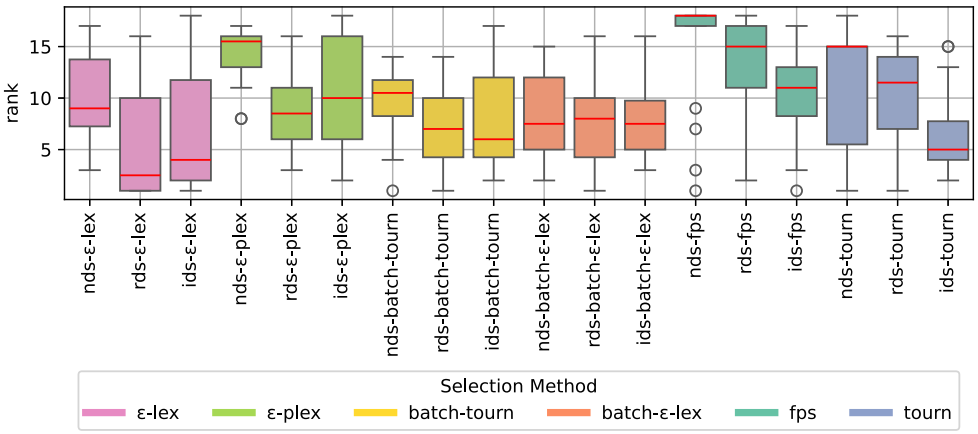

Fig. 1. *Performance* ranking of the selection methods *for a given evaluation budget*. The performance on each problem is measured in terms of the median MSE on the test cases (smaller is better).

5.2 Benchmark Problems

We study 26 problems taken from SR Bench/PMLB [La Cava et al., 2021; Romano et al., 2022] with the number of instances ranging from 100 to 1,059 and the number of features ranging from 2 to 124.[3] Table 4 shows all considered problems with the number of instances, the number of features, and the type (real-world or synthetic). We extended the problem set from our previous paper [Geiger et al., 2024] by six real-world problems to have the same number of real-world and synthetic problems and to cover a wide range of applications and problem sizes.

6 Results and Discussion

We compare lexicase-based selection methods and two traditional selection methods combined with different down-sampling strategies on a wide range of symbolic regression problems under various conditions. In detail, we compare tournament selection (denoted as *tourn*), fitness-proportionate selection (*fps*), ϵ-lexicase selection (ϵ-*lex*), ϵ-plexicase selection (ϵ-*plex*), batch-tournament selection (*batch-tourn*), and batch-ϵ-lexicase selection (*batch-ϵ-lex*) in combination with no down-sampling (*nds*), with random down-sampling (*rds*) and with informed down-sampling (*ids*).

For the sake of simplicity, only the results for the best batch size found are shown for batch-tournament selection and batch-ϵ-lexicase selection, meaning the batch size with the lowest average MSE on the validation cases for each problem.

First, we show and discuss the results for a given evaluation budget. Then, we present the results for a given running time. A statistical analysis of the results is given in Appendix E.

6.1 Results for Runs with a Given Maximum Number of Evaluations

Usually, selection methods are compared over a fixed evaluation budget because fitness evaluations are considered the most expensive operation [Boldi et al., 2024b]. This means for runs without down-sampling that each selection method searches for 100 generations to find a solution. For runs with random down-sampling ($d = 0.1$), the search is performed for 1,000 generations, and for runs with informed down-sampling ($d = 0.1$, $s = 0.01$, $g = 10$) for 991 generations (see Section 5.1).

Figure 1 plots the performance rankings of the selection methods for all 26 problems following the approach of Orzechowski et al. [2018] for comparing multiple algorithms over many datasets.

[3]Due to limited computational resources, we only considered problems with a maximum of around 1,000 instances.

For each method, we measured the relative performance on a problem by comparing the median MSE on the test cases over all 30 runs. The down-sampling strategy is indicated by a prefix (*nds* for no down-sampling, *rds* for random down-sampling, and *ids* for informed down-sampling).

We observe that almost all methods perform better in combination with a down-sampling strategy. The overall best result is achieved by rds-ϵ-lex and the second best by ids-ϵ-lex. Interestingly, we observe that ids-tourn and ids-fps perform significantly better than their counterpart without down-sampling, with ids-tourn being the third best method in terms of the median ranking.

Since studying the ranking of each method only provides a summarized overview of their performance, we also show their actual performance in terms of the median MSE on the test cases for each problem in Table 5. The problems are abbreviated by their number (see Table 4 for the full names), and the prefix *R* is added for real-world problems and *S* for synthetic problems. Boxplots of the distribution of the results for each problem are given in Appendix A (Figures A1–A3).

According to the median MSE on the test cases, rds-ϵ-lex is the best performing method on 11 out of 20 problems and ids-ϵ-lex performs best on 6 out of 20 problems (on three problems, rds-ϵ-lex and ids-ϵ-lex perform equally well). Interestingly, on 12 out of 13 synthetic problems either rds-ϵ-lex or ids-ϵ-lex performed best. Of the 13 real-world problems, rds-batch-tourn performed best on 4, rds-batch-ϵ-lex on 3, and rds-ϵ-lex only on 2. Overall, variants using random down-sampling performed best on 19 out of 26 problems.

La Cava et al. [2023] argue that for symbolic regression tasks one should not only take into account the performance of a model but also its simplicity, which is typically approximated by the size of a model. Therefore, we measured the median tree size of the final population for each run. Then, we compared the median population tree size of each method to derive their ranking for each problem (smaller median tree size is better), which is shown in Figure 2.

Tournament selection generates populations with a large median tree size, especially rds-tourn. Fitness-proportionate selection generates the smallest individuals for the variant without down-sampling, but significantly larger ones when combined with random or informed down-sampling. Apart from that, we found that the trees generated by the best performing methods rds-ϵ-lex and ids-ϵ-lex are relatively large, while the ones generated by the worst performing methods nds-fps, and nds-ϵ-plex are small. We observed that rds-batch-ϵ-lex, ids-batch-ϵ-lex, and ids-batch-tourn generate populations with relatively small individuals while still performing relatively well.

Furthermore, we observed that the tree size usually increases over time for all selection methods [Geiger et al., 2024]. However, evaluating the performance of individuals is more expensive if their tree size is large. As trees grow over time, generations get more expensive. This is not taken into account when comparing methods only for a given evaluation budget.

Additionally, the run time of the selection methods varies greatly. In Figure 3, we can see the ranking of the median time each method required to perform the runs for each problem (less time is better).

For all methods, the variants without down-sampling are faster. We can see that the fastest method is nds-fps, followed by nds-ϵ-plex, nds-batch-tourn, and nds-tourn. The slowest methods are the ϵ-lex variants. This means that some methods could perform more generations in the same time than others. Therefore, we are not only comparing the selection methods over a given evaluation budget but also over a given time period.

To sum up, we found that ϵ-lexicase selection in combination with random and informed down-sampling performs best for a given evaluation budget, especially for synthetic problems. For real-world problems, batch-tournament and batch-ϵ-lexicase selection combined with random down-sampling performed well. Further, we found that the combination with down-sampling improved the performance for almost all selection methods. We observed that the median tree size of the generated populations as well as the run time varies greatly between methods. Therefore, we

Table 5. Median MSE on the Test Cases for Each Selection Method on All 26 Problems

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
Problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	0.53	0.51	0.38	0.51	0.46	0.40	0.40	**0.33**	0.37	0.35	0.36	0.36	0.66	0.57	0.54	0.57	0.39	0.38
R-1028	0.55	0.50	0.49	0.50	0.48	0.51	0.47	0.45	0.48	0.45	**0.44**	0.47	0.62	0.48	0.49	0.48	0.45	0.46
R-1030	2.85	2.81	2.96	3.20	2.95	3.06	2.78	2.81	2.99	2.83	2.75	2.77	3.28	2.99	2.99	2.78	**2.71**	2.88
R-207	1.19e+07	1.05e+07	1.36e+07	1.15e+07	1.25e+07	1.05e+07	1.14e+07	1.14e+07	1.06e+07	1.22e+07	**8.92e+06**	1.16e+07	1.42e+07	1.03e+07	9.88e+06	9.32e+06	1.14e+07	1.02e+07
R-230	4270.43	**3567.15**	4268.94	8685.30	4436.89	4022.64	6131.70	5600.98	5339.38	6049.93	5527.29	4262.77	9720.09	6145.74	5322.44	7565.13	7669.26	4829.54
R-4544	0.48	**0.34**	0.35	0.52	0.39	**0.34**	0.44	0.35	0.39	0.43	0.46	0.49	0.56	0.51	0.43	0.57	0.45	0.43
R-505	46.92	18.78	13.60	75.75	31.28	19.66	37.20	**12.07**	13.27	27.08	13.76	14.99	112.80	24.99	19.39	38.20	17.96	13.37
R-519	2.90	3.10	3.00	2.90	2.95	3.31	2.84	2.84	3.17	2.74	2.71	2.99	**2.49**	2.50	2.63	2.50	2.55	2.84
R-522	0.78	0.75	0.78	0.77	0.78	0.84	**0.69**	0.72	0.78	0.72	0.71	0.73	0.75	0.76	0.77	0.71	0.71	0.72
R-523	1.12	1.11	1.25	1.12	1.05	1.11	1.05	**1.01**	1.15	1.06	1.15	1.07	1.07	**1.01**	1.08	1.03	1.11	1.12
R-560	30.70	24.13	15.03	42.09	23.82	28.09	22.54	8.21	16.96	22.22	**7.30**	10.31	50.87	28.45	22.58	27.36	20.37	18.11
S-581	0.13	0.09	**0.08**	0.47	0.12	0.15	0.21	0.19	0.13	0.18	0.23	0.15	0.76	0.75	0.27	0.58	0.44	0.11
S-589	0.20	**0.07**	**0.07**	0.31	0.18	0.16	0.24	0.20	0.15	0.24	0.21	0.20	0.44	0.33	0.24	0.29	0.23	0.17
S-591	0.32	**0.26**	0.41	0.43	0.45	0.63	0.42	0.34	0.39	0.32	0.36	0.41	0.46	0.44	0.36	0.42	0.44	0.30
S-606	0.19	0.08	**0.07**	0.33	0.17	0.14	0.22	0.22	0.12	0.23	0.22	0.14	0.40	0.37	0.26	0.30	0.32	0.11
S-607	0.13	**0.07**	0.08	0.51	0.11	0.12	0.21	0.20	0.12	0.25	0.14	0.13	0.84	0.79	0.70	0.56	0.32	0.13
S-615	0.19	**0.14**	0.15	0.63	0.15	0.22	0.21	0.19	0.17	0.17	0.20	0.19	0.75	0.76	0.19	0.54	0.38	0.16
S-617	0.16	0.10	**0.08**	0.38	0.15	0.12	0.18	0.18	0.11	0.18	0.17	0.12	0.58	0.37	0.15	0.33	0.24	0.09
S-621	0.39	0.52	0.54	0.51	0.53	0.60	0.57	0.50	0.63	0.36	0.46	0.57	0.64	0.56	**0.35**	0.51	0.59	0.42
S-623	0.15	**0.06**	0.07	0.36	0.12	0.11	0.15	0.13	0.08	0.15	0.15	0.10	0.70	0.53	0.21	0.43	0.20	0.09
S-624	0.44	**0.28**	0.48	0.50	0.52	0.66	0.41	0.43	0.53	0.35	0.42	0.53	0.53	0.49	0.46	0.42	0.50	0.40
S-641	0.16	**0.07**	**0.07**	0.35	0.17	0.16	0.23	0.21	0.16	0.22	0.16	0.18	0.65	0.34	0.28	0.29	0.27	0.16
S-647	0.21	**0.13**	0.16	0.44	0.17	0.21	0.22	0.28	0.21	0.19	0.25	0.18	0.74	0.48	0.26	0.35	0.29	0.18
S-654	0.27	**0.09**	**0.09**	0.36	0.25	0.16	0.27	0.18	0.10	0.23	0.26	0.23	0.58	0.42	0.30	0.34	0.32	0.13
R-665	81.60	81.58	96.96	79.03	75.41	101.67	76.31	94.13	94.13	70.86	79.19	83.57	67.47	75.71	99.07	**61.02**	62.76	92.08
R-666	7.10	6.60	6.36	7.57	6.52	6.85	6.52	**5.01**	6.16	7.20	5.89	5.94	13.22	6.84	6.14	10.18	5.94	7.27

The results refer to the best solution found for *a given evaluation budget*. Best results are shown in bold font. The results are rounded to two decimal places.

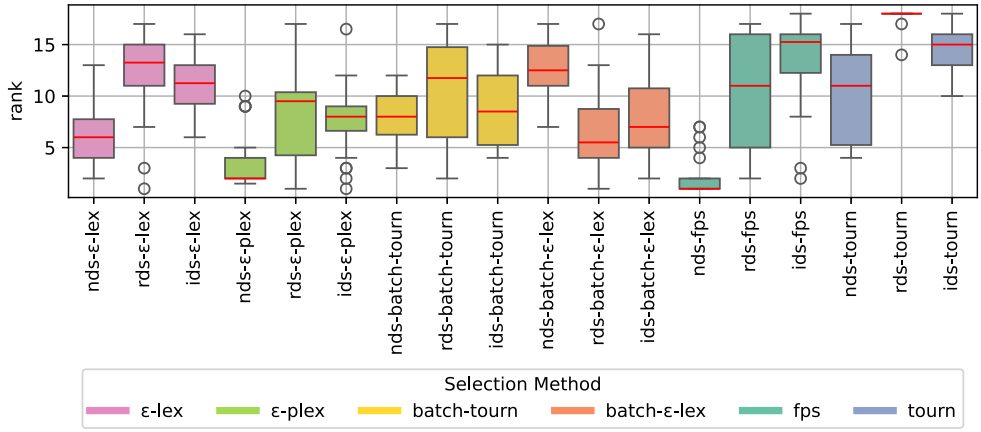

Fig. 2. Ranking of the median *tree size* for all selection methods *for a given evaluation budget*. The tree size is measured in terms of the median tree size of the final population (smaller is better).

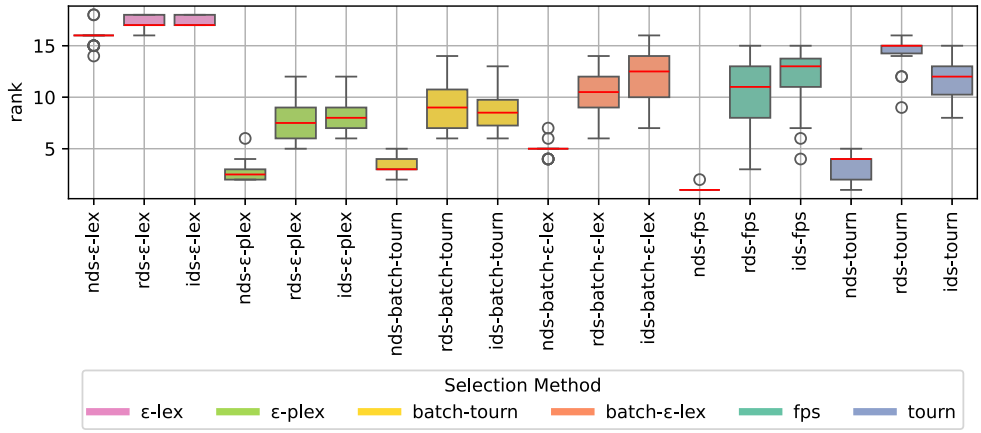

Fig. 3. Ranking of the median *run time* of each selection method *for a given evaluation budget* (less is better).

are not only comparing the methods for a given maximum number of evaluations but also for a given amount of time.

6.2 Results for Runs with a Given Maximum Running Time

Now, we compare all methods for a given running time instead of a given maximum number of evaluations. This means that some methods run for more generations than others. We analyze the results for different run times to observe if there are differences in the relative performance of each method over time. We have chosen time budgets of 24 h, 1 h, and 15 minutes based on our observations in our previous paper [Geiger et al., 2024]. We observed that most changes in the performance of the methods occurred within the first hour. The relative performance remained similar at the end of the observed 24 h period.

In Figure 4, the performance ranking of each method is shown after a running time of 24 h, 1 h, or 15 minutes. First, we compare the best solution found within a running time of 24 h. For all methods, the combination with a down-sampling method improved the performance. For 4 out of 6 methods, the combination with informed down-sampling performed best. Overall, the best

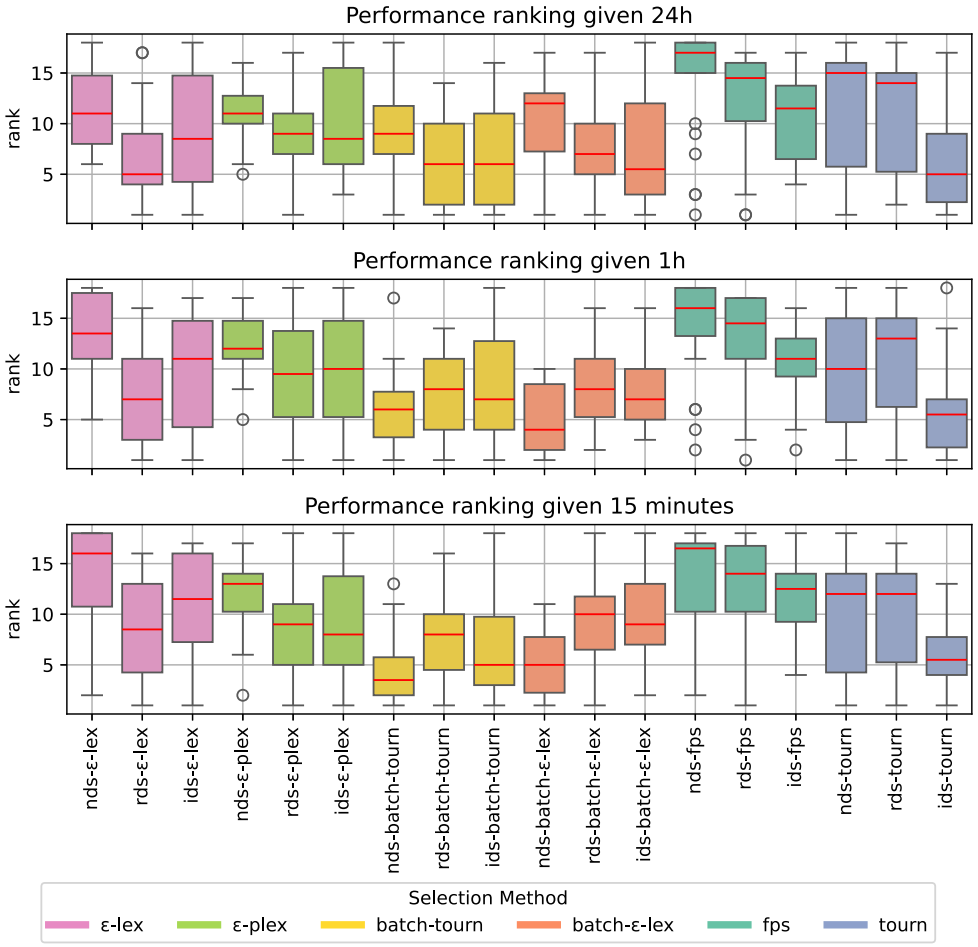

Fig. 4. *Performance* ranking of the selection methods given 24 h, 1 h, or 15 minutes. The performance on each problem is measured in terms of the median MSE on the test cases (smaller is better).

median ranking within 24 h was achieved by ids-tourn and rds-ϵ-lex, followed by ids-batch-ϵ-lex. The worst performing methods are nds-fps and nds-tourn.

For practitioners, the time to find a solution for a problem is often strictly limited. Therefore, we also analyze the results found after a shorter running time. The rankings of the results found within 1 h differ only slightly from the ones for 24 h. One major difference is that the relative performance of batch-tourn and batch-ϵ-lex without down-sampling is much better now. According to the median ranking after 1 h, the best performing method is nds-batch-ϵ-lex, followed by ids-tourn and nds-batch-tourn.

We observed further changes in the performance rankings when we limited the time even more. Looking at the results after 15 minutes, we can see that nds-batch-tourn is the best performing method now, followed by ids-batch-tourn and nds-batch-ϵ-lex. Boxplots of the distribution of the results for 24 h, 1 h, and 15 minutes are shown in Appendix B (Figures B1–B9).

To get a more detailed view at the results, Table 6 shows the median MSE of all methods on all problems after 24 h. Although ids-tourn and rds-ϵ-lex performed best according to the median

Table 6. Median MSE on the Test Cases for Each Selection Method on All 26 Problems

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
Problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	0.59	0.45	0.36	0.43	0.36	0.39	0.37	0.33	0.34	0.35	**0.32**	0.34	0.42	0.36	0.34	0.36	0.38	0.34
R-1028	0.50	0.49	0.48	0.46	0.45	0.46	0.43	0.44	0.45	**0.42**	0.43	0.46	0.47	0.47	0.47	0.44	**0.42**	0.45
R-1030	2.86	2.82	2.84	2.81	2.80	2.75	**2.61**	2.64	2.76	2.67	2.73	2.87	2.99	2.86	2.83	2.65	2.71	2.80
R-207	9.49e+06	1.01e+07	1.20e+07	9.39e+06	9.16e+06	1.15e+07	8.90e+06	**6.83e+06**	7.51e+06	1.02e+07	8.35e+06	9.54e+06	9.33e+06	8.45e+06	1.05e+07	8.14e+06	7.17e+06	8.22e+06
R-230	4297.13	**3247.03**	3419.23	4920.95	4263.54	4705.39	8421.04	4316.76	5056.89	4683.25	4322.90	5403.67	4242.00	4123.41	3543.73	5003.36	3494.20	3733.44
R-4544	0.32	0.31	0.32	0.33	**0.29**	0.33	0.33	0.30	0.33	0.33	0.32	0.32	0.49	0.46	0.36	0.47	0.39	0.40
R-505	11.02	10.17	6.37	11.90	6.31	7.41	7.98	**3.17**	6.15	9.30	4.00	5.34	12.22	8.52	6.07	8.69	13.26	4.50
R-519	3.23	2.91	3.51	2.85	3.05	3.26	2.81	2.93	3.07	2.68	3.01	3.23	2.61	**2.55**	2.85	2.58	2.80	3.03
R-522	0.72	0.75	0.83	0.71	0.77	0.86	0.68	0.68	0.79	**0.67**	0.71	0.77	0.71	0.72	0.70	**0.67**	0.69	0.74
R-523	1.10	1.06	1.12	1.10	1.09	1.09	1.08	1.07	1.10	1.11	1.16	1.17	**1.01**	1.05	1.13	1.06	1.05	1.12
R-560	11.68	2.45	3.44	9.59	2.06	2.62	4.64	**1.92**	2.28	13.74	2.14	2.87	18.46	13.67	11.01	9.66	6.88	2.74
S-581	0.08	0.06	0.06	0.09	0.10	0.07	0.07	0.08	**0.05**	0.11	0.08	**0.05**	0.55	0.18	0.09	0.30	0.31	**0.05**
S-589	0.11	0.05	**0.04**	0.11	0.08	0.05	0.15	0.14	**0.04**	0.14	0.12	0.12	0.32	0.31	0.21	0.21	0.20	0.13
S-591	0.25	0.17	0.30	0.27	0.31	0.43	0.23	0.19	0.31	0.24	0.20	0.20	0.38	0.34	0.22	0.39	0.36	**0.14**
S-606	0.11	0.06	0.06	0.10	0.08	0.07	0.14	0.18	**0.04**	0.17	0.10	0.05	0.33	0.31	0.18	0.23	0.26	0.05
S-607	0.08	0.06	**0.05**	0.09	0.08	0.06	0.08	0.10	**0.05**	0.09	0.08	**0.05**	0.75	0.31	0.08	0.26	0.18	0.06
S-615	0.11	0.09	**0.07**	0.11	0.08	**0.07**	0.10	0.10	0.08	0.11	0.09	**0.07**	0.62	0.34	0.10	0.45	0.30	0.08
S-617	0.08	0.07	0.06	0.09	0.08	0.06	0.08	0.10	0.05	0.10	0.10	0.05	0.17	0.12	0.09	0.24	0.19	**0.04**
S-621	0.33	**0.12**	0.42	0.38	0.57	0.63	0.34	**0.12**	0.39	0.36	0.17	0.24	0.56	0.47	0.16	0.56	0.53	**0.12**
S-623	0.09	0.05	0.05	0.09	0.07	0.06	0.07	0.08	**0.04**	0.07	0.08	**0.04**	0.52	0.17	0.07	0.20	0.16	0.05
S-624	0.24	0.14	0.36	0.29	0.39	0.61	0.32	0.16	0.37	0.25	**0.13**	0.26	0.40	0.37	0.15	0.40	0.38	0.14
S-641	0.07	0.05	0.04	0.08	0.07	0.04	0.14	0.10	**0.03**	0.11	0.06	0.04	0.27	0.22	0.23	0.21	0.20	0.06
S-647	0.08	0.09	0.06	0.12	0.06	0.06	0.13	0.11	0.07	0.17	0.08	0.06	0.25	0.24	0.12	0.25	0.21	**0.05**
S-654	0.10	**0.07**	0.08	0.12	0.09	0.09	0.09	**0.07**	**0.07**	0.12	0.08	**0.07**	0.35	0.27	0.11	0.27	0.25	**0.07**
R-665	93.82	76.34	116.30	78.78	87.83	108.50	103.81	82.36	89.56	86.34	91.18	92.17	71.18	**69.17**	106.33	71.15	80.85	110.90
R-666	6.85	5.27	6.87	6.60	6.27	6.23	5.17	**4.82**	5.64	6.75	5.62	4.85	6.75	6.44	5.97	6.69	5.67	5.78

The results refer to the best solution found after *24h*. Best results are shown in bold font. The results are rounded to two decimal places.

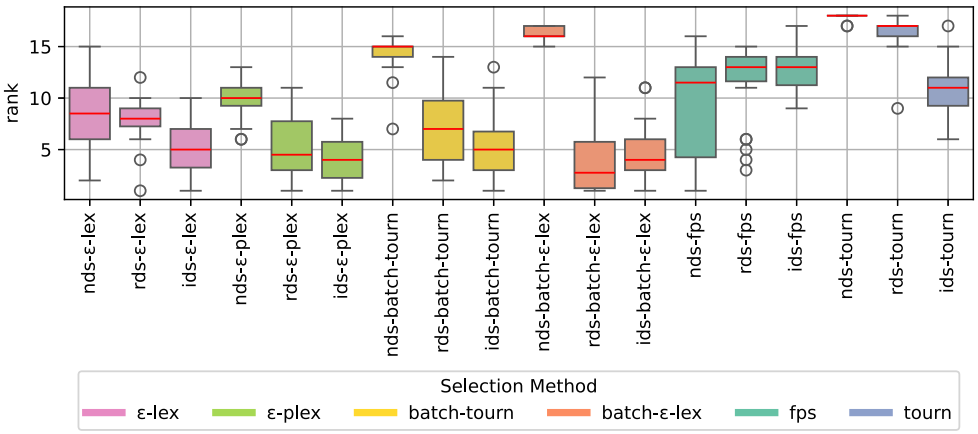

Fig. 5. Ranking of the median *tree size* for all selection methods *given 24 h*. The tree size is measured in terms of the median tree size of the final population (smaller is better).

ranking, they were only the best performing methods on six and three problems, respectively. However, ids-batch-tourn performed best on seven problems, and rds-batch-tourn on six problems. All in all, variants using informed down-sampling achieved the best results on 12 out of 13 synthetic problems, and variants using random down-sampling achieved the best results on 10 out of 13 real-world problems. The detailed results for the time budgets of 1 h and 15 minutes are shown in Appendix B (Tables B1 and B2).

As before, we are not only analyzing the performance but also at the tree sizes generated by each method. Figure 5 shows the ranking of the median population tree size for the population after 24 h. The populations with the smallest individuals are generated by rds-batch-ϵ-lex, followed by ids-batch-ϵ-lex and ids-ϵ-plex. Ids-batch-ϵ-lex was the third best performing method (according to Figure 4), suggesting that this method offers a good compromise between size and performance. Interestingly, in most cases, the variants with down-sampling generate populations with smaller individuals than their counterpart without down-sampling.

Additionally, the rankings of the tree sizes for the populations given 1 h or 15 minutes are shown in Appendix C (Figures C1 and C2). Again, tournament selection generates larger trees compared to the other methods. The populations with the smallest median tree size are generated by nds-ϵ-lex.

Further, Tables D1–D3 in Appendix D show the median number of performed generations for a given running time. Obviously, the variants using down-sampling searched for a larger number of generations compared to the methods without down-sampling. Generally, the ϵ-lexicase variants search for the lowest median number of generations.

6.3 Key Findings

To sum up, we observed that the relative performance of the selection methods as well as the influence of the down-sampling strategies depends on the given termination criteria. For a better overview, we report the median ranking of each method for a given number of evaluation and the studied time budgets in Table 7.

Down-sampling generally improves the performance. Only for shorter running times, the lexicase variants using batches of training cases perform better without down-sampling. It is worth noting that informed down-sampling has the greatest influence on the performance of tournament selection. Tournament selection without down-sampling or with random down-sampling is among

Table 7. Overview of the Median Ranking of Each Method for a Given Termination Criteria

Method	# Evaluations	24 h	1 h	15 min
nds-ϵ-lex	9.0	11.0	13.5	16.0
rds-ϵ-lex	**2.5**	**5.0**	7.0	8.5
ids-ϵ-lex	4.0	8.5	11.0	11.5
nds-ϵ-plex	15.5	11.0	12.0	13.0
rds-ϵ-plex	8.5	9.0	9.5	9.0
ids-ϵ-plex	10.0	8.5	10.0	8.0
nds-batch-tourn	10.5	9.0	6.0	**3.5**
rds-batch-tourn	7.0	6.0	8.0	8.0
ids-batch-tourn	6.0	6.0	7.0	5.0
nds-batch-ϵ-lex	7.5	12.0	**4.0**	5.0
rds-batch-ϵ-lex	8.0	7.0	8.0	10.0
ids-batch-ϵ-lex	7.5	5.5	7.0	9.0
nds-fps	18.0	17.0	16.0	16.5
rds-fps	15.0	14.5	14.5	14.0
ids-fps	11.0	11.5	11.0	12.5
nds-tourn	15.0	15.0	10.0	12.0
rds-tourn	11.5	14.0	13.0	12.0
ids-tourn	5.0	**5.0**	5.5	5.5

The best median rankings are shown in bold.

the worst performing methods in all settings, while tournament selection combined with informed down-sampling performs well in all studied settings and is even the best performing method for a given running time of 24 h. We assume that the subset of training cases created with informed down-sampling leads to an average error that weights different behaviors more equally, since training cases that test the same behavior are excluded.

Further, we observe that ϵ-lexicase selection combined with random down-sampling is the best performing method for a given number of evaluations and a running time of 24 h. However, the median ranking is worse when the ϵ-lexicase variants are compared for a given running time, especially if the running time is short. We assume that this is due to the long running times of ϵ-lexicase (Figure 3), which means that the search is performed for a lower number of generations compared to other methods (Appendix D).

The median ranking of batch-tournament selection and batch-ϵ-lexicase selection gets better for shorter running times, and they even outperform the other methods for a given time of 1 h or 15 minutes. This indicates that the use of batches of training cases is especially useful if the given time to find a solution is short. We assume that this is because the batch variants evaluate individuals from the beginning on several training cases, making the selection process less dependent on the order of the training cases [Aenugu and Spector, 2019].

7 Conclusions

We compared relevant lexicase variants in combination with different down-sampling strategies on a wide range of symbolic regression problems. As a baseline, we included fitness-proportionate selection and tournament selection. To our knowledge, this is the first work studying the influence of informed down-sampling on various selection methods on a wide range of symbolic regression

problems. We not only compared the selection method over a fixed number of evaluations, as usually done in the literature, but also over different running times as users usually have limited time to solve their problems. Therefore, this work provides users with a comprehensive guide for choosing the best selection method for their individual problem. Furthermore, we analyzed the influence of the selection methods on the solution size.

We found that the relative performance of each selection method depends on the given setting. For a fixed number of evaluations, ϵ-lexicase selection in combination with random or informed down-sampling performed best, especially for synthetic problems. For almost all selection methods, the combination with a down-sampling strategy increased the performance. Furthermore, we observed that the median population tree size as well as the median run time differs between methods. This highlights the importance of comparing the selection methods also for a given running time and not just for a given evaluation budget, as it is usually the case in the literature.

Therefore, we studied the solutions found after 24 h, as well as after 1 h and 15 minutes. For a given running time of 24 h, the best performing methods are tournament selection in combination with informed down-sampling and ϵ-lexicase selection in combination with random down-sampling. If running time is more limited (1 h or less), lexicase variants using batches of training cases outperform all other methods.

To sum up, choosing the right selection method depends on the given setting. For a given evaluation budget, we recommend using ϵ-lexicase selection in combination with a down-sampling strategy. If there is a need for a quick solution, batch-ϵ-lexicase selection or batch-tournament selection lead to the best solution. Interestingly, the combination of tournament selection with informed down-sampling performed well in all settings. However, the detailed results show that while there are general trends, the performance of each method depends on the problem at hand.

In future work, we will extend this study by analyzing other population dynamics next to tree size like diversity or specialist selection. Additionally, we will study the influence of informed down-sampling on tournament selection in the domain of symbolic regression further.

References

Sneha Aenugu and Lee Spector. 2019. Lexicase selection in learning classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 356–364.

Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. 1998. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers Inc.

Ryan Boldi, Ashley Bao, Martin Briesch, Thomas Helmuth, Sobania Dominik, Spector Lee, and Lalejini Alexander. 2023a. The problem solving benefits of down-sampling vary by selection scheme. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion)*. ACM, New York, NY, 527–530.

Ryan Boldi, Ashley Bao, Martin Briesch, Thomas Helmuth, Dominik Sobania, Lee Spector, and Alexander Lalejini. 2024a. Untangling the effects of down-sampling and selection in genetic programming. In *Proceedings of the 2024 Artificial Life Conference (ALIFE '24)*. MIT Press.

Ryan Boldi, Martin Briesch, Dominik Sobania, Alexander Lalejini, Thomas Helmuth, Franz Rothlauf, Charles Ofria, and Lee Spector. 2024b. Informed down-sampled lexicase selection: Identifying productive training cases for efficient problem solving. *Evolutionary Computation* 32, 4 (2024), 307–337.

Ryan Boldi, Alexander Lalejini, Thomas Helmuth, and Lee Spector. 2023b. A static analysis of informed down-samples. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion)*. ACM, New York, NY, 531–534.

Shu-Heng Chen. 2012. *Genetic Algorithms and Genetic Programming in Computational Finance*. Springer Science & Business Media.

Vinícius V. De Melo, Danilo Vasconcellos Vargas, and Wolfgang Banzhaf. 2019. Batch tournament selection for genetic programming: The quality of lexicase, the speed of tournament. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. ACM, 994–1002.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.

Li Ding, Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022a. Going faster and hence further with lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 538–541.

Li Ding, Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022b. Lexicase selection at scale. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2054–2062.

Li Ding, Edward Pantridge, and Lee Spector. 2023. Probabilistic lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. ACM, 1073–1081.

Li Ding and Lee Spector. 2021. Optimizing neural networks with gradient lexicase selection. In *Proceedings of the International Conference on Learning Representations*, 1–15.

Emily Dolson. 2023. Calculating lexicase selection probabilities is NP-Hard. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 1575–1583.

Yongsheng Fang and Jun Li. 2010. A review of tournament selection in genetic programming. In *Advances in Computation and Intelligence*. Lecture Notes in Computer Science, Vol. 6382, Springer Berlin, 181–192.

Austin J. Ferguson, Jose Guadalupe Hernandez, Daniel Junghans, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2020. Characterizing the effects of random subsampling on lexicase selection. In *Genetic Programming Theory and Practice XVII*. Springer International Publishing, 1–23.

Félix-Antoine Fortin, François-Michel de Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research* 13, 1 (2012), 2171–2175.

Alina Geiger, Dominik Sobania, and Franz Rothlauf. 2023. Down-sampled epsilon-lexicase selection for real-world symbolic regression problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. ACM, 1109–1117.

Alina Geiger, Dominik Sobania, and Franz Rothlauf. 2024. A comprehensive comparison of lexicase-based selection methods for symbolic regression problems. In *Proceedings of the European Conference on Genetic Programming (Part of EvoStar)*. Springer, 192–208.

Ivo Gonçalves, Sara Silva, Joana B. Melo, and João M. B. Carreiras. 2012. Random sampling technique for overfitting control in genetic programming. In *Genetic Programming*. Springer Berlin, 218–229.

Thomas Helmuth and Amr Abdelhady. 2020. Benchmarking parent selection for program synthesis by genetic programming. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (GECCO '20)*. ACM, 237–238.

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016a. Effects of lexicase and tournament selection on diversity recovery and maintenance. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion)*. ACM, 983–990.

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016b. Lexicase selection for program synthesis: a diversity analysis. In *Genetic Programming Theory and Practice XIII*. Springer International Publishing, 151–167.

Thomas Helmuth, Edward Pantridge, and Lee Spector. 2019. Lexicase selection of specialists. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. ACM, 1030–1038.

Thomas Helmuth, Edward Pantridge, and Lee Spector. 2020. On the importance of specialists for lexicase selection. *Genetic Programming and Evolvable Machines* 21, 3 (2020), 349–373.

Thomas Helmuth and Lee Spector. 2015. General program synthesis benchmark suite. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*. ACM, 1039–1046.

Thomas Helmuth and Lee Spector. 2020. Explaining and exploiting the advantages of down-sampled lexicase selection. In *Proceedings of the 2020 Conference on Artificial Life (ALIFE '20)*. MIT Press, 341–349.

Thomas Helmuth and Lee Spector. 2021. Problem-solving benefits of down-sampled lexicase selection. *Artificial Life* 27, 3-4 (2021), 183–203.

Thomas Helmuth, Lee Spector, and James Matheson. 2014. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2014), 630–643.

Alberto Hernandez, Adarsh Balasubramanian, Fenglin Yuan, Simon A. M. Mason, and Tim Mueller. 2019a. Fast, accurate, and transferable many-body interatomic potentials by symbolic regression. *NPJ Computational Materials* 5, 1 (2019), 112.

Jose Guadalupe Hernandez, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2019b. Random subsampling improves performance in lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19)*. ACM, 2028–2031.

Jose Guadalupe Hernandez, Alexander Lalejini, and Charles Ofria. 2022. An exploration of exploration: Measuring the ability of lexicase selection to find obscure pathways to optimality. In *Genetic Programming Theory and Practice XVIII*. 83–107.

Dorit S. Hochbaum and David B. Shmoys. 1985. A best possible heuristic for the k-center problem. *Mathematics of Operations Research* 10, 2 (1985), 180–184.

Lia Jundt and Thomas Helmuth. 2019. Comparing and combining lexicase selection and novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1047–1055.

Jonathan Kelly, Eric Hemberg, and Una-May O'Reilly. 2019. Improving genetic programming with novel exploration—Exploitation control. In *Proceedings of the European Conference on Genetic Programming*. Springer, 64–80.

John R. Koza. 1992. On the programming of computers by means of natural selection. In *A Bradford Book*, Vol. 1. MIT Press.

Krzysztof Krawiec and Una-May O'Reilly. 2014. Behavioral programming: A broader and more detailed take on semantic GP. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*. ACM, 935–942.

William La Cava, Thomas Helmuth, Lee Spector, and Jason H. Moore. 2019. A probabilistic and multi-objective analysis of lexicase selection and epsilon-lexicase selection. *Evolutionary Computation* 27, 3 (2019), 377–402.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. 2021. Contemporary symbolic regression methods and their relative performance. In *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 1–16.

William La Cava, Lee Spector, and Kourosh Danai. 2016. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, 741–748.

William G. La Cava, Paul C. Lee, Imran Ajmal, Xiruo Ding, Priyanka Solanki, Jordana B. Cohen, Jason H. Moore, and Daniel S. Herman. 2023. A flexible symbolic regression method for constructing interpretable clinical prediction models. *NPJ Digital Medicine* 6, 1 (2023), 107.

Jared M. Moore and Adam Stanton. 2017. Lexicase selection outperforms previous strategies for incremental evolution of virtual creature controllers. In *Proceedings of the 14th European Conference on Artificial Life (ECAL '17)*, 290–297.

Jared M. Moore and Adam Stanton. 2018. Tiebreaks and diversity: Isolating effects in lexicase selection. In *Proceedings of the 2018 Conference on Artificial Life*. MIT Press, Cambridge, MA, 590–597.

Andrew Ni, Li Ding, and Lee Spector. 2024. DALex: Lexicase-like selection via diverse aggregation. In *Proceedings of the European Conference on Genetic Programming (Part of EvoStar)*. Springer, 90–107.

Ji Ni, Russ H. Drieberg, and Peter I. Rockett. 2013. The use of an analytic quotient operator in genetic programming. *IEEE Transactions on Evolutionary Computation* 17, 1 (2013), 146–152.

Patryk Orzechowski, Jason H. Moore, and William La Cava. 2018. Where are we now? A large benchmark study of recent symbolic regression methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1183–1190.

Edward Pantridge, Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2018. Specialization and elitism in lexicase and tournament selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18)*. ACM, 1914–1917.

Thu Pham-Gia and Tran Loc Hung. 2001. The mean and median absolute deviations. *Mathematical and Computer* 34, 7-8 (2001), 921–936.

Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. 2008. *A Field Guide to Genetic Programming*. Lulu Press.

Miha Ravber, Shih-Hsi Liu, Marjan Mernik, and Matej Črepinšek. 2022. Maximum number of generations as a stopping criterion considered harmful. *Applied Soft Computing* 128 (2022), 109478.

Joseph D. Romano, Trang T. Le, William La Cava, John T. Gregg, Daniel J. Goldberg, Praneel Chakraborty, Natasha L. Ray, Daniel Himmelstein, Weixuan Fu, and Jason H. Moore. 2022. PMLB v1.0: An open-source dataset collection for benchmarking machine learning methods. *Bioinformatics (Oxford, England)* 38, 3 (2022), 878–880.

Dirk Schweim, Dominik Sobania, and Franz Rothlauf. 2022. Effects of the training set size: A comparison of standard and down-sampled lexicase selection in program synthesis. In *Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

Dominik Sobania and Franz Rothlauf. 2021. A generalizability measure for program synthesis with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*. ACM, 822–829.

Dominik Sobania and Franz Rothlauf. 2022. Program synthesis with genetic programming: The influence of batch sizes. In *Proceedings of the European Conference on Genetic Programming*. Springer, 118–129.

Dominik Sobania, Dirk Schweim, and Franz Rothlauf. 2023. A comprehensive survey on program synthesis with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 27, 1 (2023), 82–97.

Lee Spector. 2012. Assessment of problem modality by differential performance of lexicase selection in genetic programming: A preliminary report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '12)*. ACM, 401–408.

Lee Spector, William La Cava, S. Shanabrook, Thomas Helmuth, and Edward Pantridge. 2018. Relaxations of lexicase parent selection. In *Genetic Programming Theory and Practice XV*. Springer, 105–120.

Alexander R. M. Wagner and Anthony Stein. 2021. Adopting lexicase selection for Michigan-style learning classifier systems with continuous-valued inputs. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 171–172.

Appendices

A Results for a Given Maximum Number of Evaluations

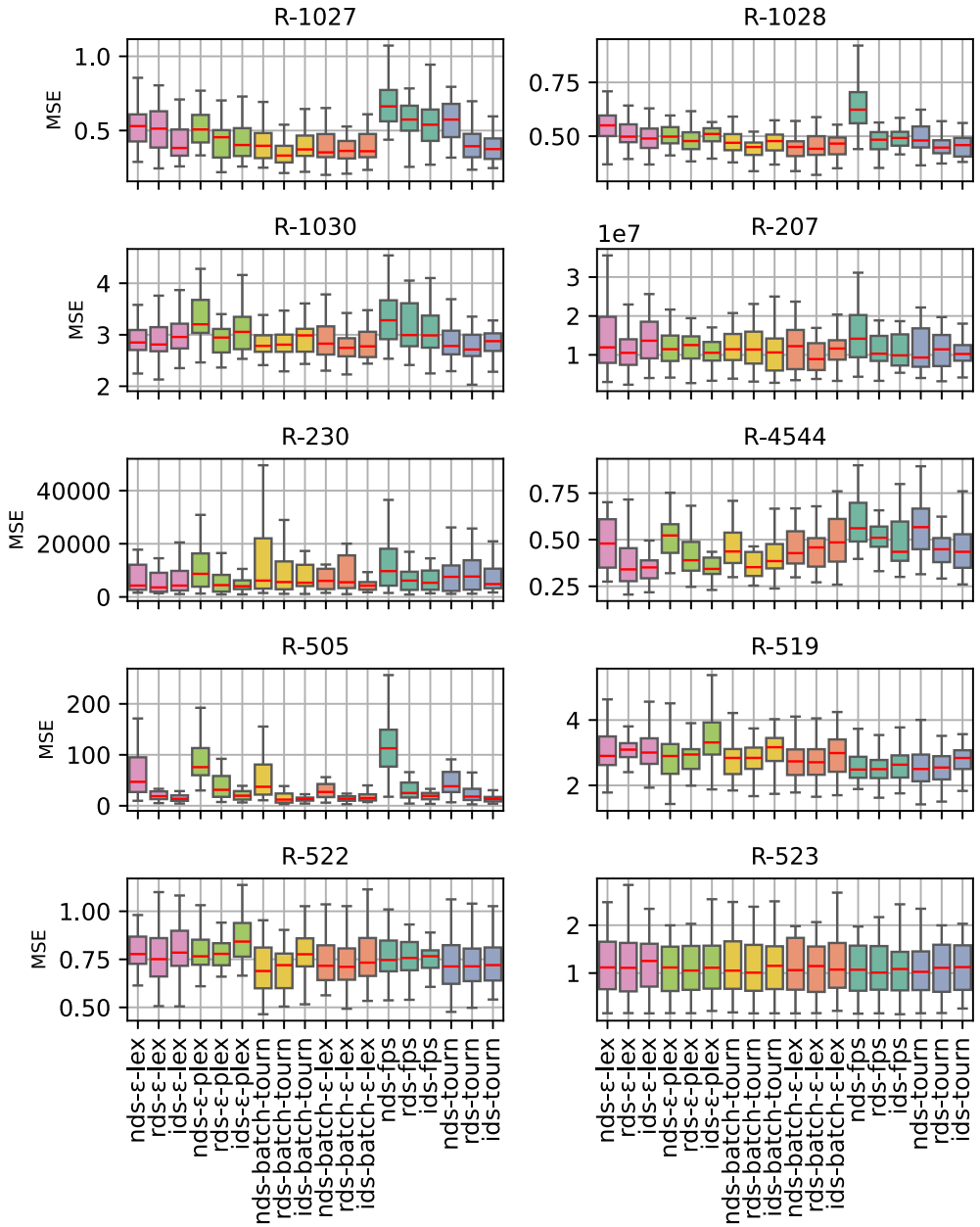

Fig. A1. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after a given evaluation budget*. Outliers are not shown to improve readability.

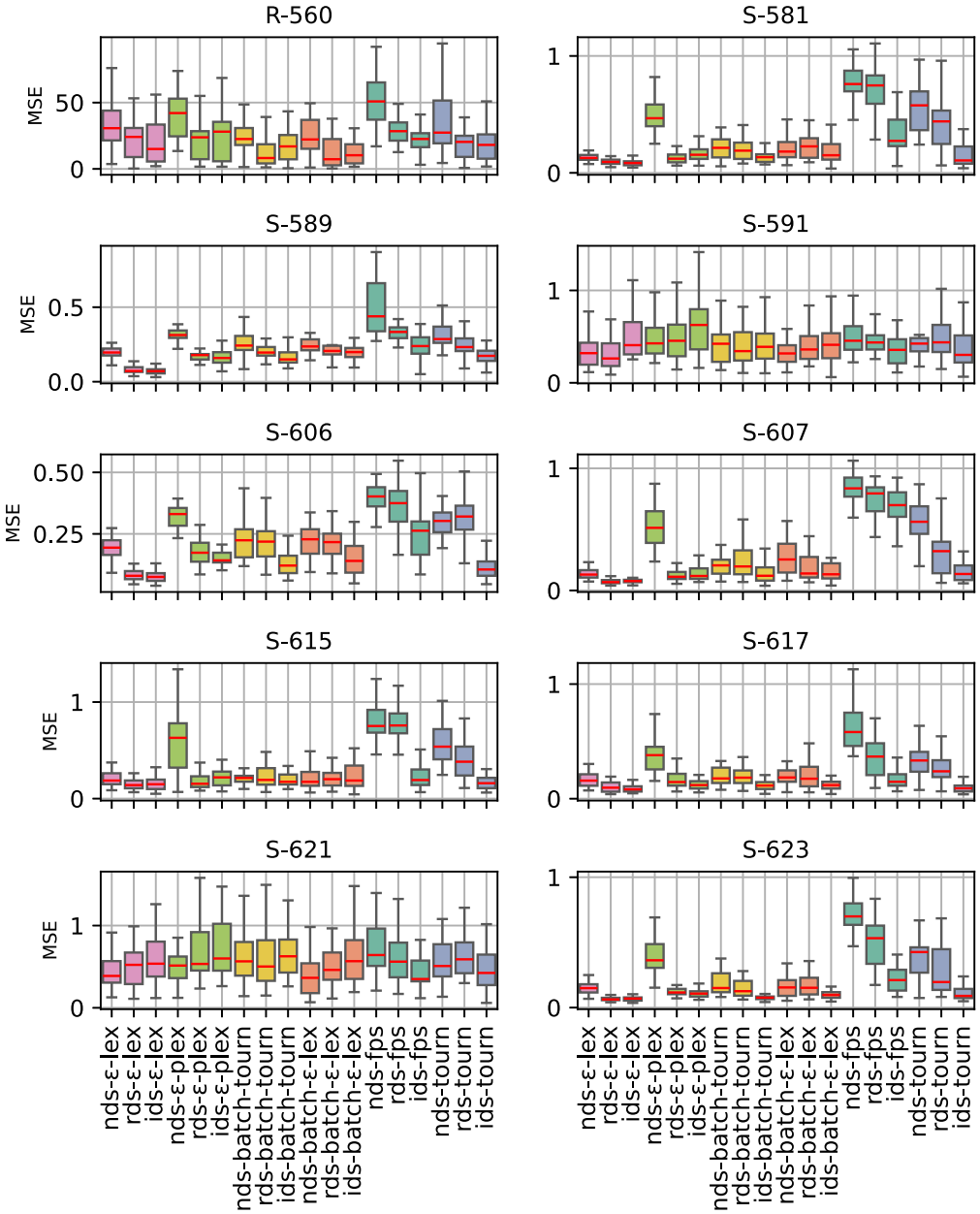

Fig. A2. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after a given evaluation budget*. Outliers are not shown to improve readability.

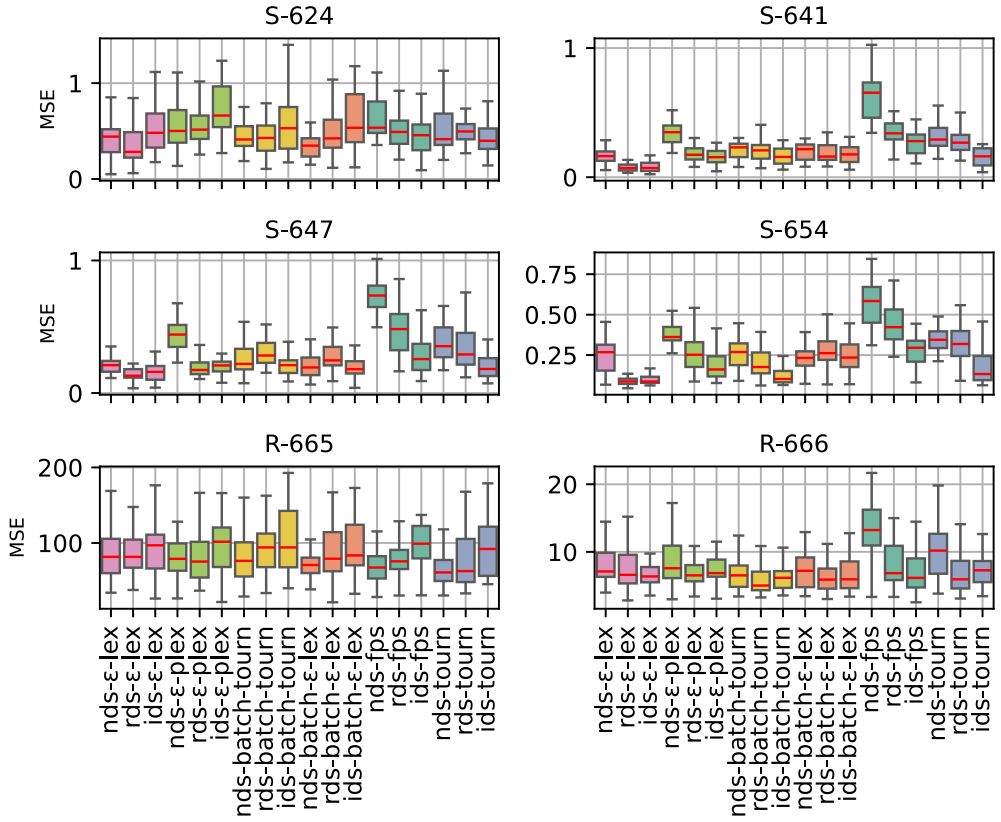

Fig. A3. MSE on the test cases for each selection method on six problems. The results refer to the best solution found *after a given evaluation budget*. Outliers are not shown to improve readability.

B Results for a Given Run Time

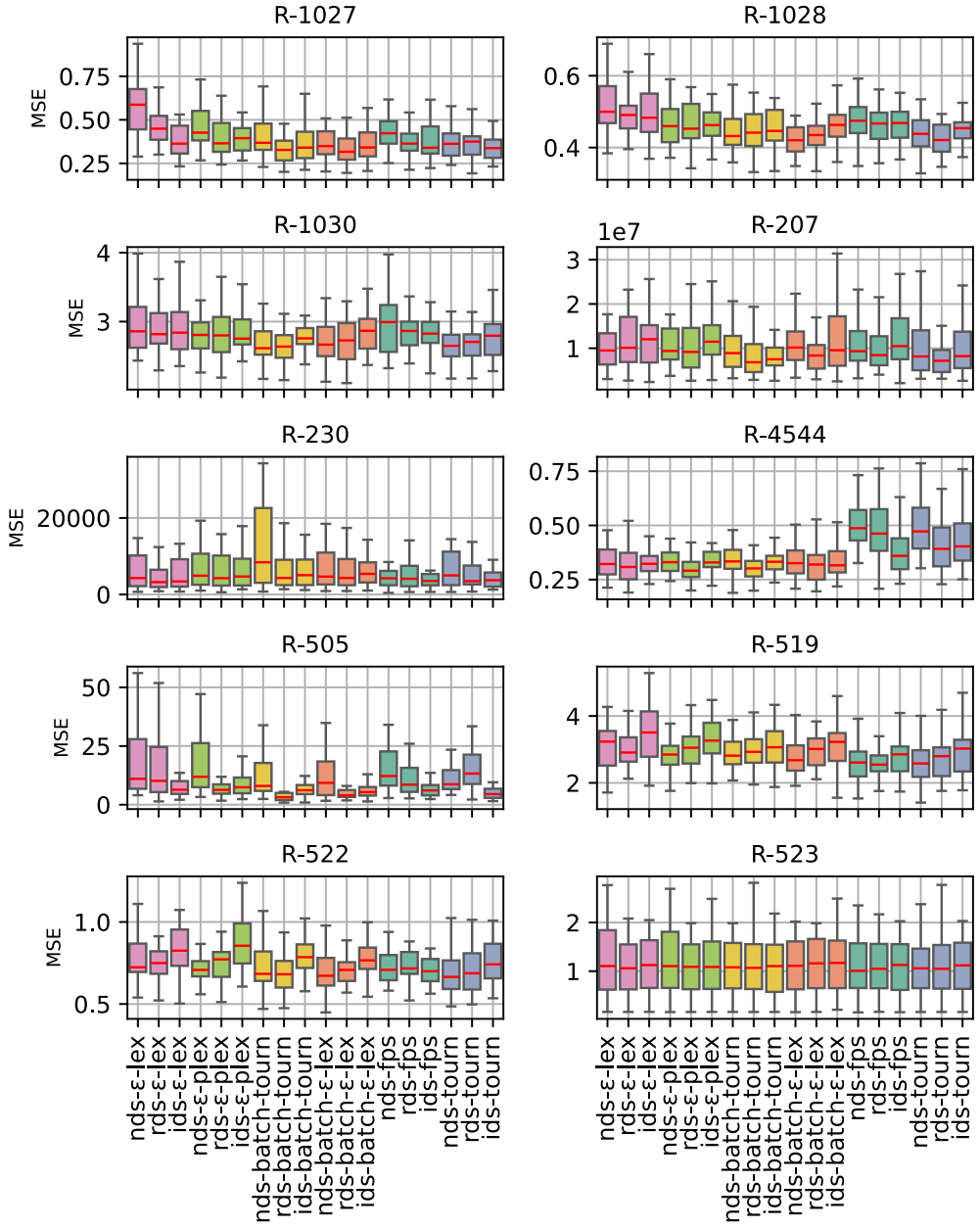

Fig. B1. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 24 h*. Outliers are not shown to improve readability.

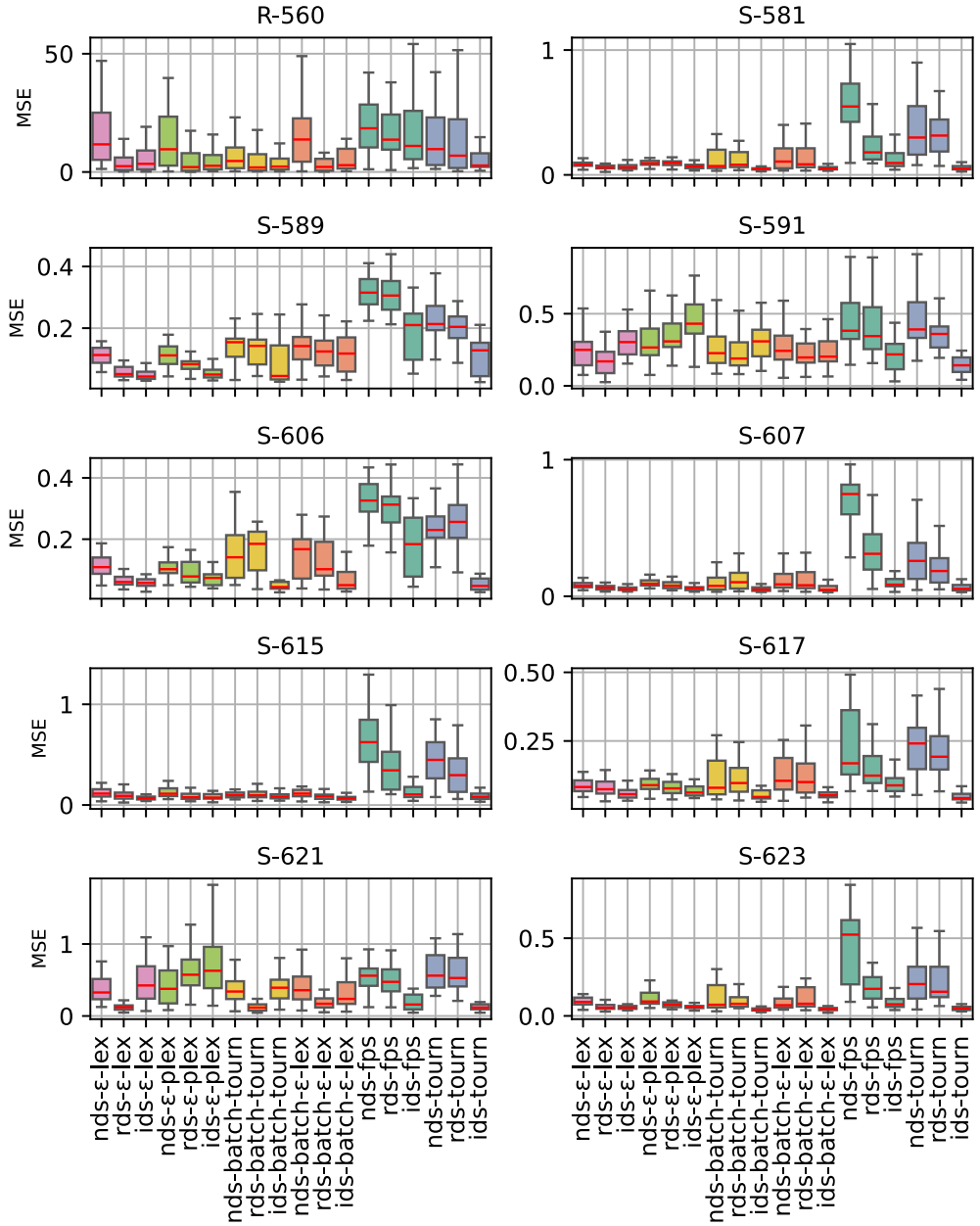

Fig. B2. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 24 h*. Outliers are not shown to improve readability.

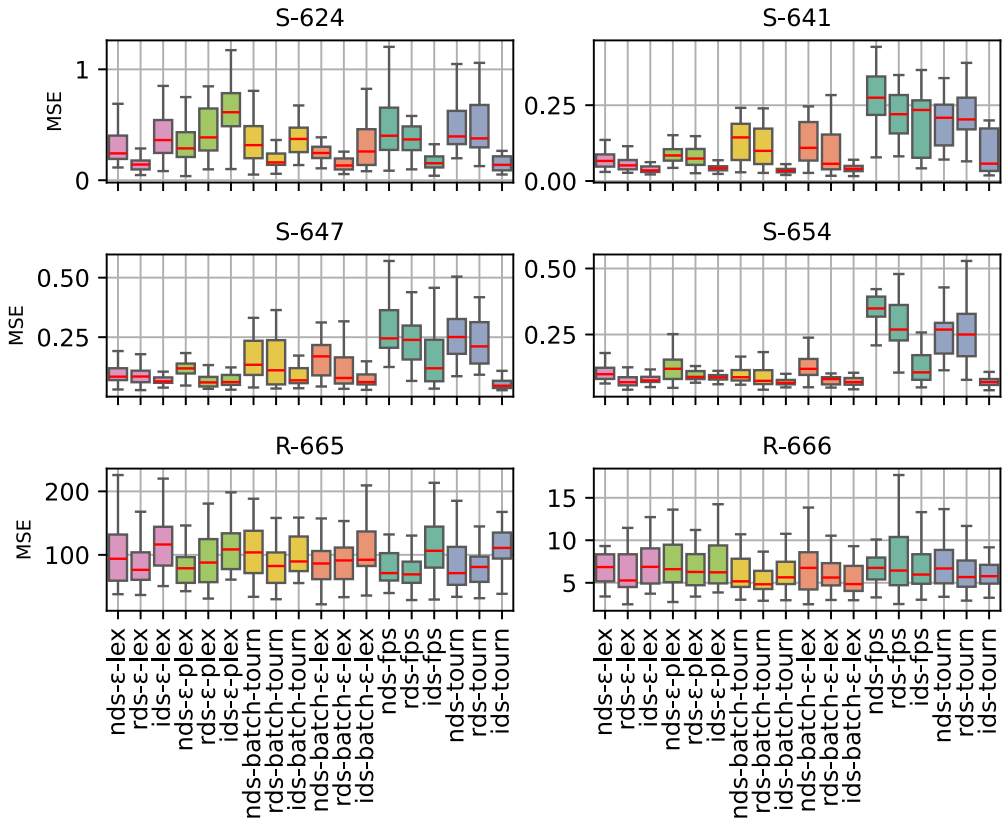

Fig. B3. MSE on the test cases for each selection method on six problems. The results refer to the best solution found *after 24 h*. Outliers are not shown to improve readability.

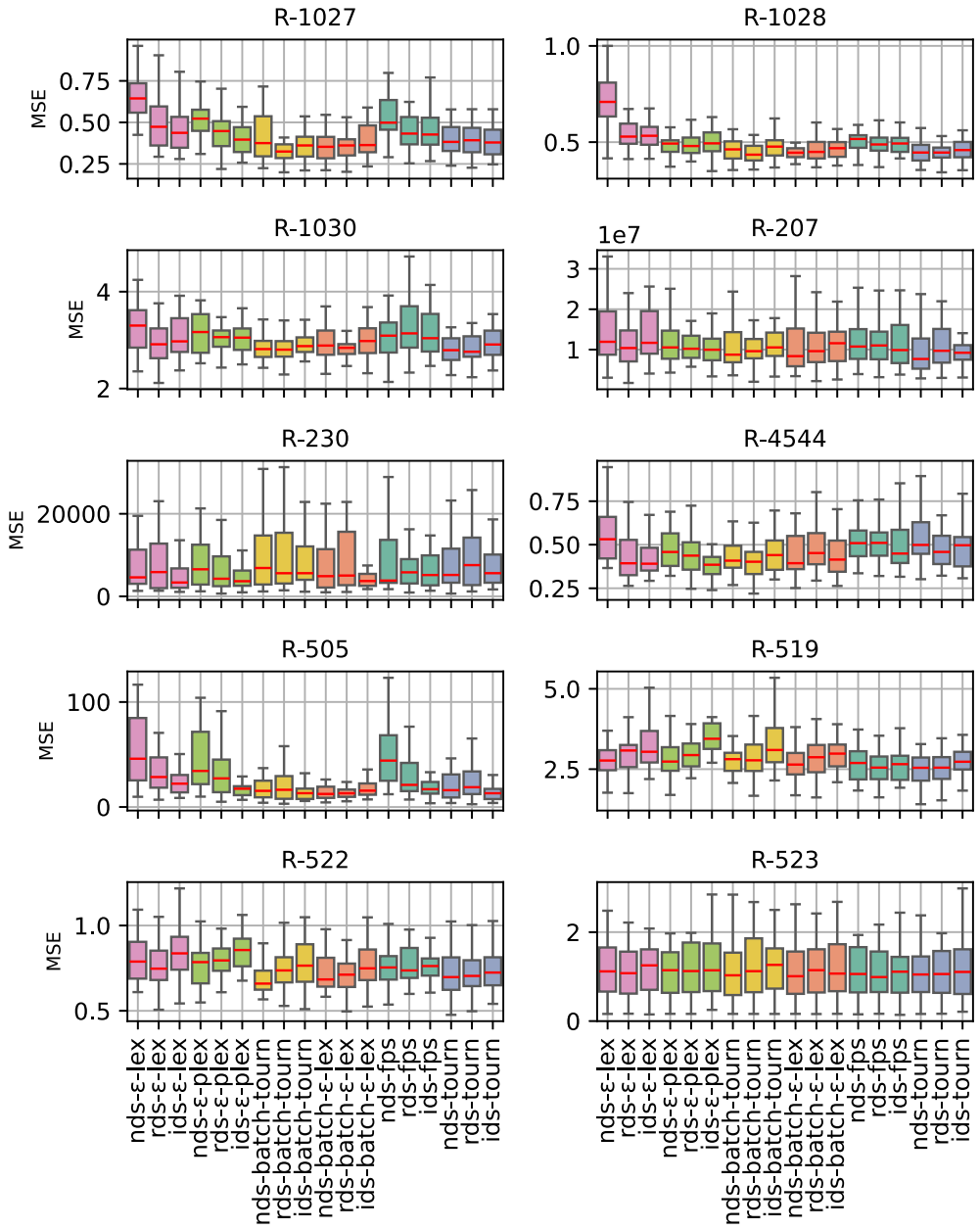

Fig. B4. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 1 h*. Outliers are not shown to improve readability.

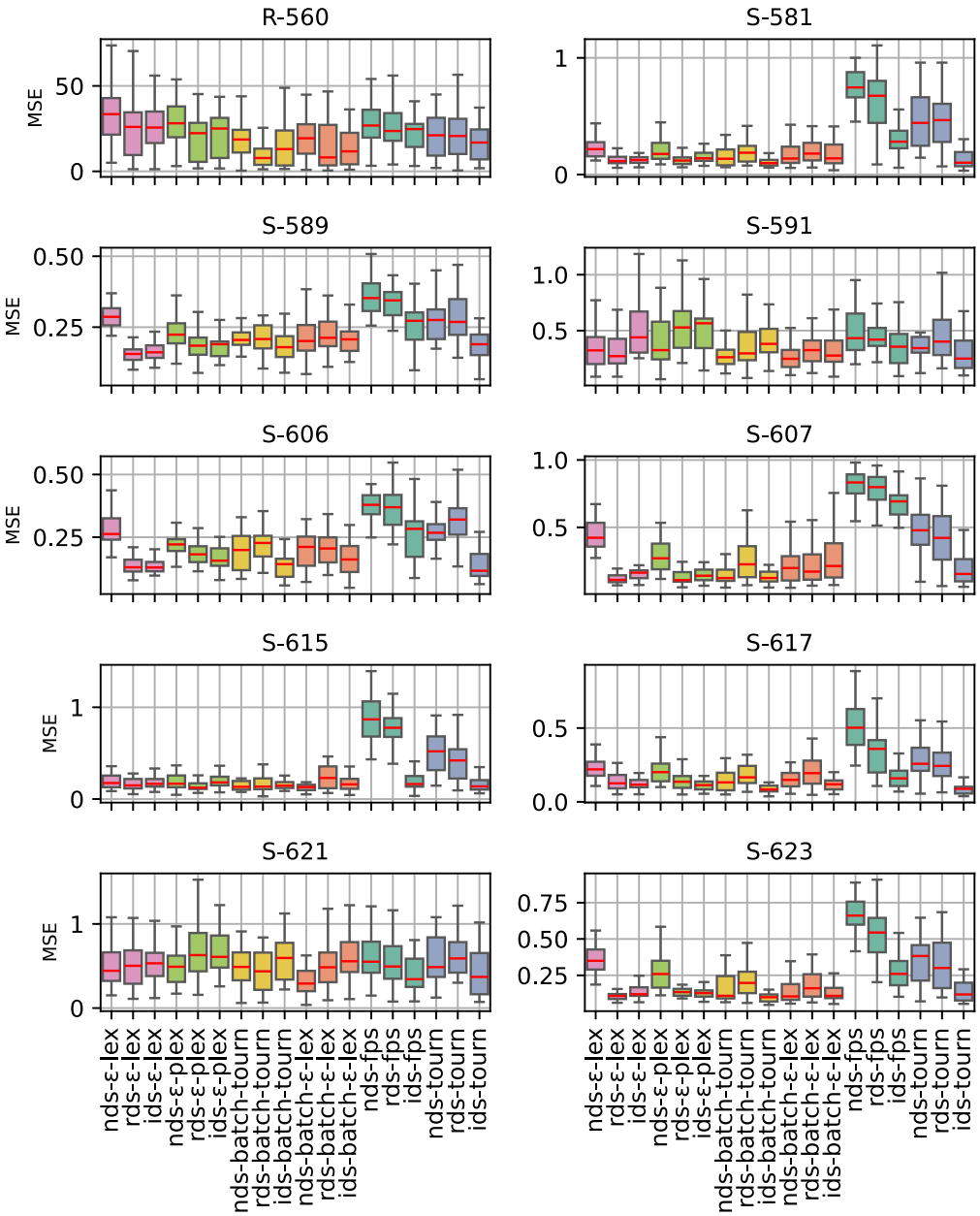

Fig. B5. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 1 h*. Outliers are not shown to improve readability.

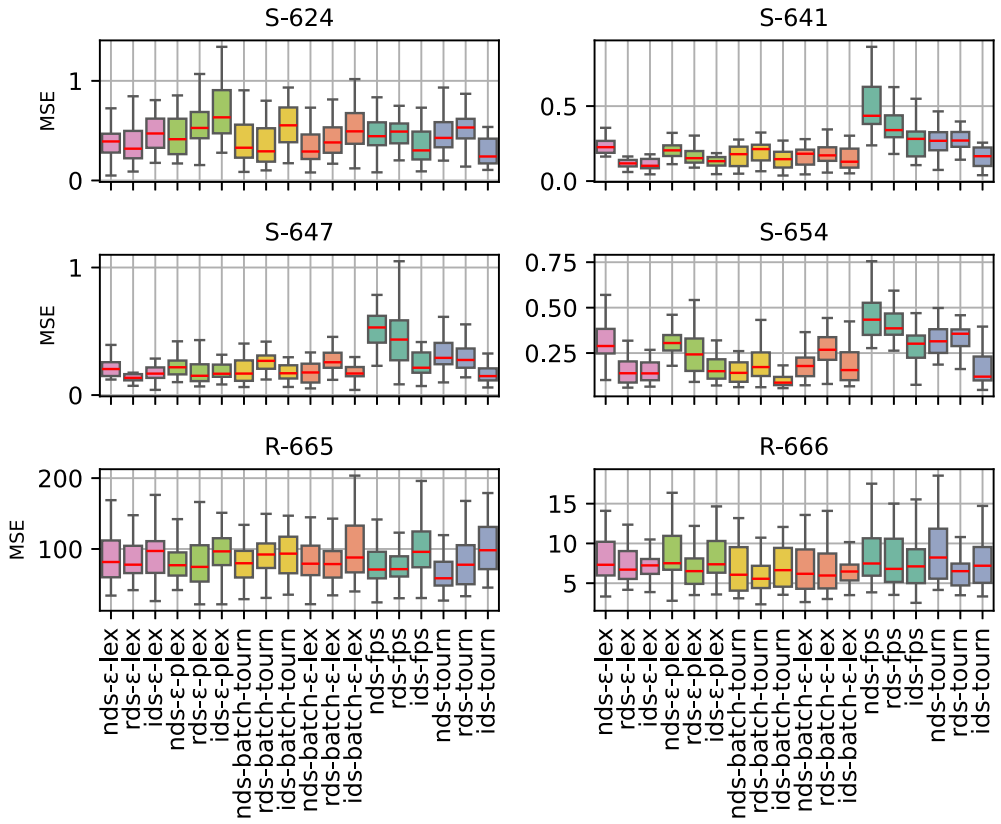

Fig. B6. MSE on the test cases for each selection method on six problems. The results refer to the best solution found *after 1h*. Outliers are not shown to improve readability.

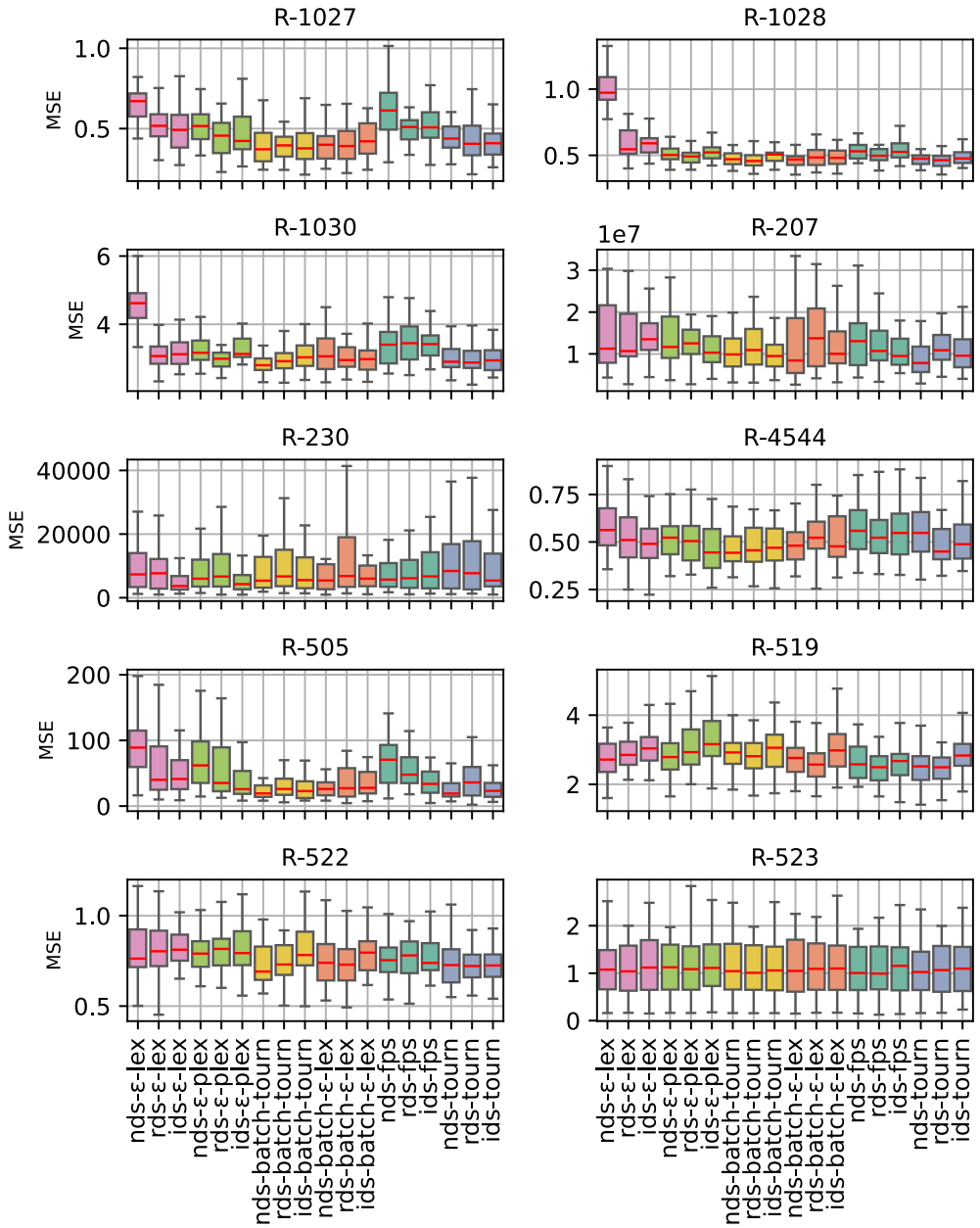

Fig. B7. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 15 minutes*. Outliers are not shown to improve readability.

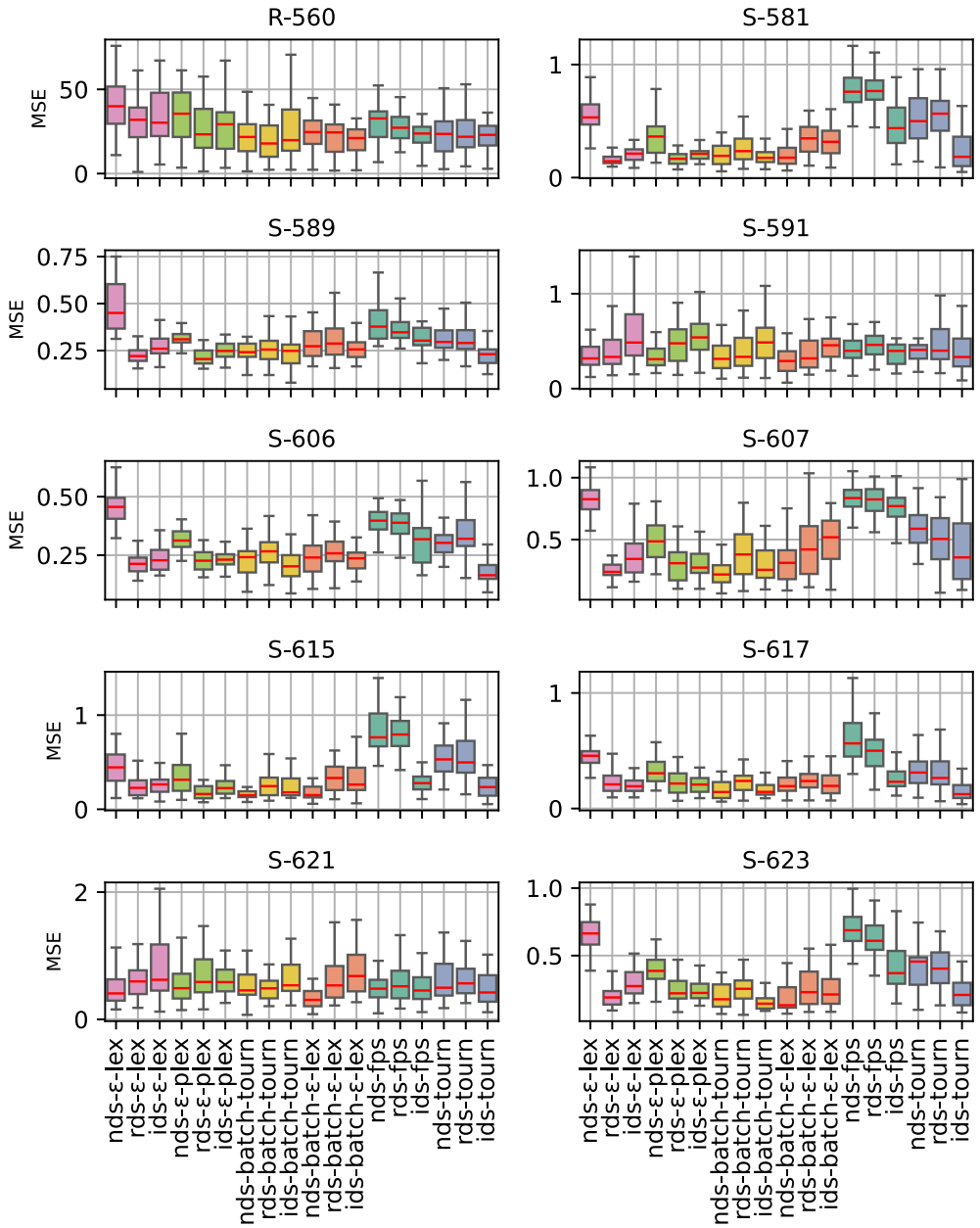

Fig. B8. MSE on the test cases for each selection method on 10 problems. The results refer to the best solution found *after 15 minutes*. Outliers are not shown to improve readability.

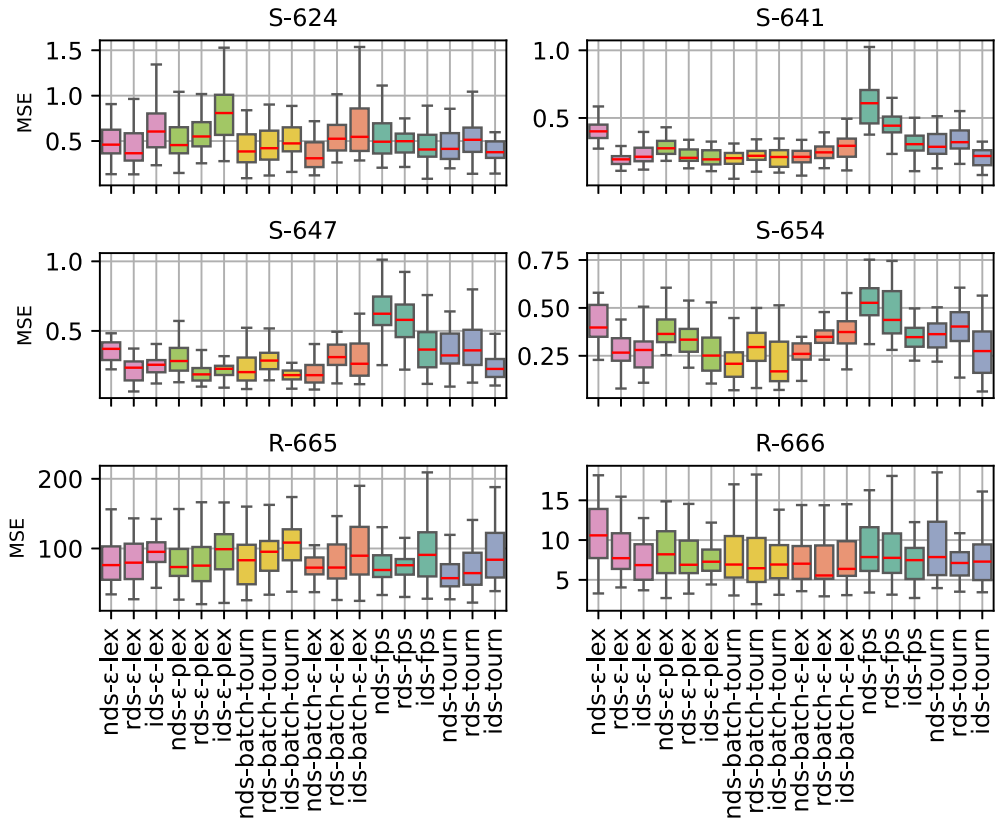

Fig. B9. MSE on the test cases for each selection method on six problems. The results refer to the best solution found *after 15 minutes*. Outliers are not shown to improve readability.

Table B1. Median MSE on the Test Cases for Each Selection Method on All 26 Problems

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	0.64	0.47	0.44	0.52	0.45	0.40	0.38	**0.32**	0.36	0.35	0.36	0.36	0.50	0.43	0.43	0.38	0.39	0.38
R-1028	0.71	0.53	0.53	0.49	0.48	0.49	0.46	**0.43**	0.48	0.45	0.45	0.47	0.52	0.49	0.49	0.45	0.45	0.46
R-1030	3.30	2.91	2.97	3.17	3.06	3.05	2.81	2.80	2.88	2.89	2.84	2.98	3.09	3.14	3.04	2.79	**2.76**	2.91
R-207	1.19e+07	1.04e+07	1.17e+07	1.05e+07	1.02e+07	9.96e+06	8.74e+06	9.61e+06	1.05e+07	8.38e+06	9.61e+06	1.16e+07	1.07e+07	1.10e+07	9.87e+06	**7.67e+06**	9.70e+06	9.22e+06
R-230	4597.58	5877.57	**3353.63**	6552.53	4266.53	3672.30	6863.01	5601.98	5606.45	4890.56	5039.28	3719.03	3825.49	5819.78	5165.00	5182.75	7564.11	5609.15
R-4544	0.53	**0.39**	**0.39**	0.46	0.44	**0.39**	0.41	0.40	0.44	**0.39**	0.45	0.41	0.51	0.51	0.45	0.50	0.46	0.50
R-505	45.98	28.55	22.23	34.36	27.25	17.43	15.40	16.41	13.16	**12.46**	13.06	15.60	44.20	21.08	16.95	16.03	18.86	13.12
R-519	2.77	3.08	3.04	2.74	2.94	3.45	2.81	2.78	3.10	2.64	2.87	2.98	2.69	2.55	2.66	**2.54**	2.55	2.73
R-522	0.79	0.75	0.84	0.79	0.80	0.86	**0.66**	0.74	0.76	0.68	0.71	0.75	0.75	0.74	0.76	0.70	0.70	0.72
R-523	1.12	1.08	1.25	1.14	1.12	1.14	1.03	1.12	1.26	1.01	1.14	1.07	1.06	**0.99**	1.11	1.05	1.06	1.10
R-560	33.47	25.97	25.62	28.12	22.32	25.05	18.61	**7.77**	13.01	19.35	8.13	11.68	26.81	23.61	24.71	21.08	20.70	16.85
S-581	0.22	0.12	0.12	0.18	0.12	0.14	0.14	0.19	**0.10**	0.14	0.18	0.14	0.75	0.68	0.28	0.44	0.47	**0.10**
S-589	0.29	**0.16**	**0.16**	0.22	0.18	0.19	0.20	0.21	0.18	0.20	0.21	0.21	0.35	0.34	0.27	0.28	0.27	0.19
S-591	0.32	0.27	0.44	0.33	0.53	0.57	0.26	0.30	0.38	**0.25**	0.33	0.28	0.43	0.42	0.36	0.34	0.40	**0.25**
S-606	0.26	0.13	0.13	0.22	0.18	0.16	0.20	0.23	0.14	0.21	0.21	0.16	0.38	0.37	0.28	0.27	0.32	**0.12**
S-607	0.42	**0.11**	0.16	0.27	**0.11**	0.14	0.12	0.23	0.12	0.20	0.17	0.21	0.83	0.80	0.69	0.48	0.42	0.15
S-615	0.18	0.15	0.17	0.17	**0.12**	0.18	0.13	0.14	0.15	0.13	0.23	0.16	0.87	0.78	0.16	0.52	0.42	0.14
S-617	0.22	0.13	0.12	0.20	0.14	0.11	0.13	0.17	**0.08**	0.15	0.19	0.12	0.50	0.36	0.16	0.26	0.24	0.09
S-621	0.44	0.50	0.53	0.49	0.63	0.61	0.49	0.44	0.60	**0.29**	0.48	0.56	0.55	0.49	0.34	0.49	0.59	0.37
S-623	0.35	0.11	0.12	0.26	0.13	0.13	0.11	0.20	**0.10**	**0.10**	0.16	0.11	0.66	0.54	0.26	0.38	0.30	0.12
S-624	0.39	0.32	0.47	0.41	0.53	0.63	0.33	0.29	0.55	0.29	0.38	0.49	0.44	0.49	0.30	0.43	0.53	**0.24**
S-641	0.23	0.12	**0.10**	0.20	0.15	0.13	0.18	0.21	0.15	0.18	0.17	0.13	0.44	0.34	0.28	0.27	0.27	0.17
S-647	0.20	**0.14**	0.17	0.22	0.15	0.17	0.17	0.27	0.17	0.18	0.26	0.17	0.53	0.44	0.21	0.29	0.28	0.15
S-654	0.29	0.14	0.14	0.31	0.24	0.15	0.14	0.17	**0.09**	0.18	0.27	0.16	0.43	0.39	0.30	0.31	0.36	0.12
R-665	81.60	77.82	97.16	77.05	74.69	96.61	79.84	92.21	93.41	79.19	78.70	87.95	70.99	71.52	95.93	**58.63**	77.76	98.21
R-666	7.32	6.71	7.24	7.51	6.52	7.38	6.07	**5.56**	6.64	6.18	5.97	6.47	7.48	6.80	7.12	8.23	6.52	7.20

The results refer to the best solution found after *1 h*. Best results are shown in bold. The results are rounded to two decimal places.

Table B2. Median MSE on the Test Cases for Each Selection Method on All 26 Problems

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	0.67	0.52	0.49	0.52	0.46	0.42	**0.37**	0.40	0.38	0.40	0.39	0.42	0.61	0.51	0.51	0.44	0.41	0.41
R-1028	0.97	0.55	0.59	0.50	0.49	0.52	0.47	**0.46**	0.51	0.47	0.48	0.48	0.53	0.50	0.53	0.47	**0.46**	0.48
R-1030	4.62	3.07	3.12	3.17	2.99	3.13	**2.80**	2.92	3.03	3.06	2.95	2.98	3.40	3.45	3.42	2.90	2.88	2.94
R-207	1.13e+07	1.07e+07	1.35e+07	1.17e+07	1.25e+07	1.03e+07	9.87e+06	1.09e+07	9.50e+06	8.43e+06	1.37e+07	1.00e+07	1.31e+07	1.07e+07	9.50e+06	**7.83e+06**	1.09e+07	9.57e+06
R-230	7348.06	7693.86	**3677.54**	5991.03	6677.00	4293.60	5371.53	6711.46	5551.55	5437.96	6827.64	5997.39	5704.11	6152.14	6728.12	8405.18	7721.95	5411.92
R-4544	0.56	0.51	0.49	0.52	0.50	**0.44**	**0.44**	0.46	0.47	0.48	0.52	0.48	0.56	0.52	0.55	0.55	0.45	0.49
R-505	89.09	40.13	41.29	62.03	35.27	25.84	19.31	26.11	22.88	26.11	27.29	27.80	70.65	47.93	33.83	**19.20**	36.03	23.28
R-519	2.72	2.85	3.04	2.79	2.93	3.16	2.92	2.81	3.06	2.76	2.57	2.98	2.58	**2.49**	2.67	2.52	**2.49**	2.83
R-522	0.76	0.80	0.81	0.79	0.82	0.79	**0.69**	0.73	0.78	0.74	0.73	0.80	0.75	0.78	0.74	0.73	0.72	0.72
R-523	1.08	1.04	1.12	1.12	1.09	1.11	1.04	1.01	1.06	1.05	1.09	1.10	1.00	**0.99**	1.16	1.02	1.06	1.10
R-560	39.99	31.94	30.22	35.55	23.31	29.30	21.75	**17.89**	19.86	24.66	24.46	21.09	32.75	27.23	23.80	23.56	21.87	23.01
S-581	0.53	**0.14**	0.21	0.36	0.17	0.21	0.19	0.23	0.17	0.18	0.35	0.32	0.76	0.77	0.44	0.50	0.57	0.18
S-589	0.45	0.22	0.26	0.31	**0.21**	0.25	0.24	0.26	0.25	0.27	0.29	0.26	0.38	0.35	0.30	0.30	0.29	0.23
S-591	0.32	0.33	0.49	0.31	0.48	0.54	0.31	0.33	0.49	**0.29**	0.32	0.46	0.40	0.46	0.40	0.41	0.40	0.33
S-606	0.46	0.21	0.23	0.31	0.23	0.23	0.24	0.27	0.20	0.24	0.26	0.24	0.40	0.39	0.32	0.30	0.32	**0.16**
S-607	0.83	0.24	0.34	0.49	0.31	0.27	**0.22**	0.38	0.26	0.31	0.42	0.52	0.84	0.82	0.77	0.59	0.50	0.36
S-615	0.45	0.23	0.26	0.31	0.16	0.23	**0.15**	0.25	0.18	**0.15**	0.33	0.26	0.76	0.79	0.28	0.53	0.50	0.24
S-617	0.46	0.21	0.19	0.30	0.22	0.21	0.14	0.24	0.14	0.19	0.24	0.20	0.56	0.50	0.23	0.31	0.26	**0.12**
S-621	0.41	0.60	0.62	0.49	0.59	0.58	0.46	0.49	0.54	**0.31**	0.53	0.68	0.48	0.52	0.45	0.50	0.57	0.42
S-623	0.66	0.19	0.27	0.39	0.22	0.22	0.18	0.25	0.14	**0.13**	0.23	0.21	0.69	0.61	0.37	0.46	0.40	0.21
S-624	0.46	0.37	0.60	0.46	0.55	0.81	0.39	0.42	0.47	**0.31**	0.53	0.55	0.49	0.50	0.41	0.41	0.51	0.38
S-641	0.40	**0.19**	0.21	0.28	0.20	**0.19**	0.20	0.22	0.21	0.21	0.25	0.29	0.61	0.44	0.31	0.29	0.32	0.22
S-647	0.37	0.24	0.26	0.28	0.19	0.23	0.20	0.29	**0.18**	**0.18**	0.31	0.26	0.62	0.58	0.37	0.32	0.36	0.23
S-654	0.40	0.27	0.28	0.36	0.33	0.25	0.21	0.30	**0.17**	0.26	0.35	0.37	0.53	0.44	0.35	0.36	0.40	0.27
R-665	76.18	79.66	95.18	73.40	75.41	99.01	83.15	95.31	108.45	72.53	72.67	89.64	69.16	75.97	90.82	**57.45**	64.67	83.87
R-666	10.59	7.73	6.86	8.21	6.90	7.28	6.92	6.46	6.93	7.03	**5.55**	6.38	7.87	7.77	7.48	7.86	7.12	7.29

The results refer to the best solution found after *15 minutes*. Best results are shown in bold. The results are rounded to two decimal places.

C Tree Sizes for a Given Running Time of 1 h or 15 minutes

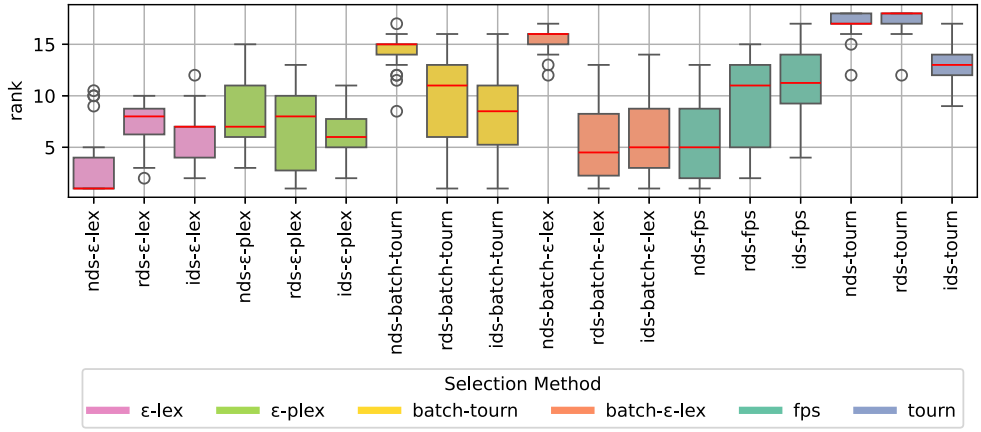

Fig. C1. Ranking of the median *tree size* for all selection methods *given 1 h*. The tree size is measured in terms of the median tree size of the final population (smaller is better).

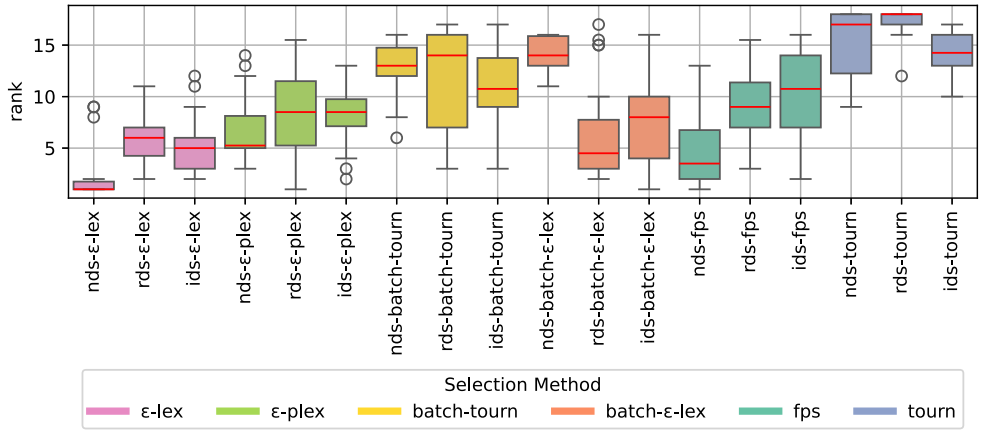

Fig. C2. Ranking of the median *tree size* for all selection methods *given 15 minutes*. The tree size is measured in terms of the median tree size of the final population (smaller is better).

D Number of Generations for a Given Running Time

Table D1. Median Number of Generations the Search Was Performed for Each Method per Problem Given *24 h*

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	726	7,370	6,103	8,351	22,476	20,916	4,545	17,325	17,384	3,755	16,578	15,900	7,193	12,811	11,063	1,860	7,780	12,408
R-1028	336	3,402	3,213	2,474	14,978	14,358	1,408	10,892	10,708	1,350	11,759	10,461	2,161	6,757	6,282	1,462	6,546	7,406
R-1030	406	3,046	3,330	2,444	10,330	17,029	1,267	9,126	13,438	1,271	9,630	11,633	1,464	4,784	5,670	1,120	4,574	9,696
R-207	2,254	12,898	13,588	13,544	60,706	46,500	9,380	56,998	44,394	4,073	35,402	29,886	10,020	23,680	20,574	3,148	15,118	28,002
R-230	1,606	10,098	9,642	7,220	26,814	27,452	5,908	40,460	31,076	3,955	37,480	22,122	6,164	14,618	13,192	2,220	13,750	18,640
R-4544	422	3,105	2,908	2,123	10,156	11,688	1,856	14,958	7,572	1,190	18,478	8,328	4,612	10,143	4,635	1,062	3,778	3,706
R-505	1,466	8,446	8,775	8,122	26,980	27,925	5,440	22,884	26,122	4,958	17,550	18,298	5,307	13,617	13,404	4,942	11,057	15,827
R-519	1,112	7,856	7,584	9,913	27,613	28,882	8,062	42,878	31,030	4,604	27,683	23,184	5,034	15,894	16,696	1,717	15,341	26,258
R-522	816	5,908	6,224	4,828	25,532	26,411	2,722	16,230	25,451	2,266	19,380	20,948	3,674	10,538	9,408	2,044	9,352	15,754
R-523	2,723	15,858	17,674	21,329	1,07,828	57,802	33,116	91,332	65,650	11,917	36,316	33,175	17,044	47,884	33,117	8,702	51,382	51,771
R-560	1,518	8,370	7,736	10,196	35,628	31,050	5,308	16,673	20,019	3,048	19,674	18,777	7,480	17,854	15,118	2,672	10,660	16,252
S-581	814	5,880	5,986	4,096	20,869	19,360	4,766	20,992	25,985	3,700	23,268	23,724	4,883	11,162	10,142	2,191	5,680	13,904
S-589	436	3,294	3,381	2,275	10,563	11,888	2,543	10,180	12,234	1,870	10,130	15,216	3,608	11,379	6,740	1,401	5,303	8,656
S-591	3,288	18,856	22,648	14,151	1,61,357	1,46,314	13,128	89,064	99,588	7,799	47,478	47,554	15,996	28,800	37,904	7,662	24,487	53,304
S-606	441	3,458	3,312	2,464	9,402	9,873	2,643	10,732	11,333	1,875	11,988	13,471	3,260	11,152	6,688	1,603	7,839	8,738
S-607	430	3,365	3,076	2,388	11,134	11,028	2,627	11,380	13,130	2,596	10,345	12,100	3,035	7,510	6,181	1,502	4,510	7,406
S-615	1,680	10,123	10,174	6,908	40,279	36,341	7,150	37,958	46,554	5,534	32,913	32,580	8,574	23,412	15,010	2,092	8,960	24,454
S-617	840	5,614	6,208	3,800	19,376	19,294	4,596	18,181	19,621	3,164	18,765	18,404	3,334	8,770	10,995	3,220	10,717	13,954
S-621	3,148	18,264	21,908	14,560	1,78,452	1,56,577	12,414	82,123	1,22,315	6,466	52,349	54,331	9,024	24,137	31,875	4,670	19,896	50,820
S-623	442	3,159	3,008	2,401	9,734	9,650	2,214	5,756	9,557	1,985	12,600	10,784	2,235	6,801	5,496	1,434	4,269	7,512
S-624	3,167	18,160	20,316	13,643	1,61,639	1,41,082	11,866	63,680	1,09,094	6,092	47,254	52,357	9,236	24,248	32,308	3,826	17,796	50,134
S-641	834	5,664	6,150	4,000	21,644	18,472	3,971	18,260	24,501	3,240	19,970	22,982	3,253	11,094	11,042	2,242	7,514	14,272
S-647	1,538	9,512	9,960	7,463	37,023	36,440	8,894	40,306	52,378	5,476	34,008	35,401	5,430	14,150	16,736	3,456	14,160	27,478
S-654	819	5,736	5,986	3,341	16,753	19,964	3,598	19,174	25,880	1,988	21,656	24,851	3,512	10,972	8,972	1,612	7,434	15,446
R-665	2,187	16,156	14,754	19,353	82,968	57,665	8,657	86,554	55,926	4,325	38,290	30,011	14,932	36,544	23,062	2,896	16,191	38,984
R-666	802	5,207	5,014	5,016	15,972	15,552	2,702	13,404	14,096	1,942	18,920	13,328	3,403	10,388	8,962	1,378	5,728	9,424

Table D2. Median Number of Generations the Search Was Performed for Each Method per Problem Given *1 h*

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	4	374	326	447	1,361	1,323	324	1,013	942	269	831	910	1,792	2,467	2,780	332	938	862
R-1028	8	165	154	188	673	632	150	539	590	123	613	497	320	688	532	218	454	450
R-1030	16	146	144	198	531	528	128	447	440	111	483	454	168	418	429	138	360	392
R-207	117	608	584	1,001	2,958	2,053	754	2,546	2,198	432	1,376	1,410	1,042	2,163	1,558	468	1,192	1,740
R-230	86	462	443	440	1,332	1,294	510	1,874	1,426	358	1,556	951	675	1,296	1,146	400	1,168	1,078
R-4544	17	160	146	262	806	653	174	896	464	148	1,108	470	850	1,124	649	236	411	436
R-505	80	417	419	656	1,488	1,284	489	1,184	1,025	350	898	982	574	1,170	914	492	852	1,020
R-519	49	341	336	457	1,318	1,422	402	1,752	1,374	266	1,161	1,068	364	1,039	1,106	234	871	1,109
R-522	38	284	278	358	1,306	1,358	242	944	990	206	837	946	343	940	812	270	771	842
R-523	124	659	730	954	4,719	2,420	1,212	4,130	2,628	462	1,667	1,227	884	2,389	1,614	758	2,147	2,216
R-560	78	468	426	779	1,586	1,494	392	1,078	1,342	265	1,032	1,048	940	1,770	1,492	344	1,034	1,379
S-581	46	298	294	409	1,180	1,458	395	1,480	1,627	325	1,284	1,266	439	1,026	994	238	502	970
S-589	27	184	181	242	815	753	258	714	898	240	690	876	304	744	572	176	496	590
S-591	171	876	934	987	6,670	6,286	908	3,980	4,170	508	2,067	2,068	1,080	2,042	2,292	640	1,612	2,752
S-606	25	208	198	242	727	713	238	880	802	251	838	682	280	780	527	212	796	706
S-607	26	172	159	257	801	742	230	848	794	189	812	769	306	812	900	180	412	650
S-615	90	490	468	550	1,996	1,758	563	1,866	2,328	396	1,715	1,579	790	1,706	1,329	238	838	1,527
S-617	46	295	322	362	1,116	1,126	473	1,250	1,351	248	1,027	1,108	352	799	905	341	732	990
S-621	166	880	904	932	7,431	6,614	864	4,511	5,210	502	2,302	2,266	777	1,516	2,098	508	1,502	2,914
S-623	26	166	155	230	698	656	234	845	556	180	790	656	276	684	584	192	405	667
S-624	166	846	852	884	6,626	6,115	740	3,602	4,761	444	2,170	2,252	704	1,601	2,093	446	1,290	2,866
S-641	46	296	346	337	1,229	1,368	332	1,185	1,656	274	1,045	1,338	402	924	1,036	248	641	1,002
S-647	85	468	458	542	1,845	2,044	657	2,292	2,645	378	1,797	1,590	654	1,287	1,184	388	1,006	1,610
S-654	42	292	284	317	1,100	1,276	364	1,274	1,672	174	1,211	1,410	462	1,002	864	228	622	1,028
R-665	114	684	619	1,072	3,648	2,602	606	3,144	1,945	391	1,641	1,315	838	1,774	1,404	393	1,394	1,778
R-666	34	262	247	354	938	908	228	648	672	176	813	654	564	1,060	846	256	526	625

Table D3. Median Number of Generations the Search Was Performed for Each Method per Problem Given *15 Minutes*

	ϵ-lex			ϵ-plex			batch-tourn			batch-ϵ-lex			fps			tourn		
problem	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids	nds	rds	ids
R-1027	1	99	92	152	446	448	138	438	320	86	366	333	1,348	2,006	1,860	221	524	525
R-1028	2	44	43	74	212	195	68	224	177	47	216	150	248	362	284	102	198	166
R-1030	2	42	40	74	175	159	54	154	127	42	124	138	82	168	168	59	138	138
R-207	32	168	163	343	894	570	287	759	458	148	448	333	448	730	566	224	482	510
R-230	24	132	127	169	403	374	179	582	420	122	404	292	292	526	449	190	456	364
R-4544	2	46	40	102	298	201	80	332	184	54	288	192	423	578	324	142	197	179
R-505	24	122	118	220	428	392	184	432	313	134	284	235	254	429	366	194	316	358
R-519	10	92	88	138	361	366	118	442	382	90	341	250	138	362	352	112	290	321
R-522	8	78	72	130	366	364	96	346	326	74	234	286	146	333	291	122	298	294
R-523	33	169	188	294	1,222	652	294	854	660	144	378	390	282	630	518	266	666	581
R-560	19	126	120	247	492	432	158	476	418	100	292	343	462	744	608	178	398	428
S-581	14	87	84	153	417	442	138	446	511	112	404	384	197	390	484	106	239	376
S-589	6	54	54	102	280	242	99	256	220	80	224	228	144	282	226	80	203	197
S-591	51	243	244	344	1,758	1,582	308	1,042	1,075	158	564	550	386	676	764	266	526	807
S-606	6	57	54	96	266	232	98	310	217	90	242	198	120	284	205	84	306	246
S-607	6	52	47	111	313	254	92	301	266	78	246	192	143	300	380	76	192	233
S-615	26	138	126	222	588	542	194	603	707	134	505	434	372	602	558	122	328	470
S-617	13	86	95	140	362	366	164	392	380	85	334	298	167	340	384	120	282	319
S-621	48	232	232	304	1,934	1,676	300	1,046	1,294	158	603	591	322	536	710	218	538	879
S-623	6	50	46	95	250	228	88	274	220	62	248	192	138	286	282	78	164	232
S-624	48	228	218	306	1,666	1,552	246	918	1,240	140	568	596	283	560	758	192	462	860
S-641	14	87	104	138	431	420	138	376	451	90	307	324	186	361	455	108	266	400
S-647	28	130	127	209	564	588	242	673	689	122	478	462	280	508	512	161	368	562
S-654	12	81	82	126	365	393	148	382	400	70	358	335	260	424	380	104	242	357
R-665	32	174	163	311	992	621	218	746	554	130	456	380	288	576	531	167	476	488
R-666	6	73	70	128	312	298	106	317	214	64	228	192	307	516	392	130	240	236

E Results of the Statistical Analysis

For our statistical analysis, we followed the recommendations of Demšar [2006] for comparing multiple methods over many datasets. For each configuration, we first use the Friedman test to test the null-hypothesis that all methods perform equivalent. We reject the null-hypothesis if $p < 0.05$. If the null-hypothesis is rejected, we use the Nemenyi post-hoc test for pairwise comparisons. The performance of two methods is significantly different if $p < 0.05$.

The p-values of the Friedman test for each configuration are reported in Table E1. We reject the null-hypothesis in all cases. Therefore, we conducted the Nemenyi test for all configurations. Figures E1–E4 report the corresponding p-values.

Table E1. Results of the Friedman Test for All Configurations

Configuration	p-Value
Given a fixed evaluation budget	3.92e-18
Given 24 h	5.52e-11
Given 1 h	1.37e-15
Given 15 minutes	1.03e-18

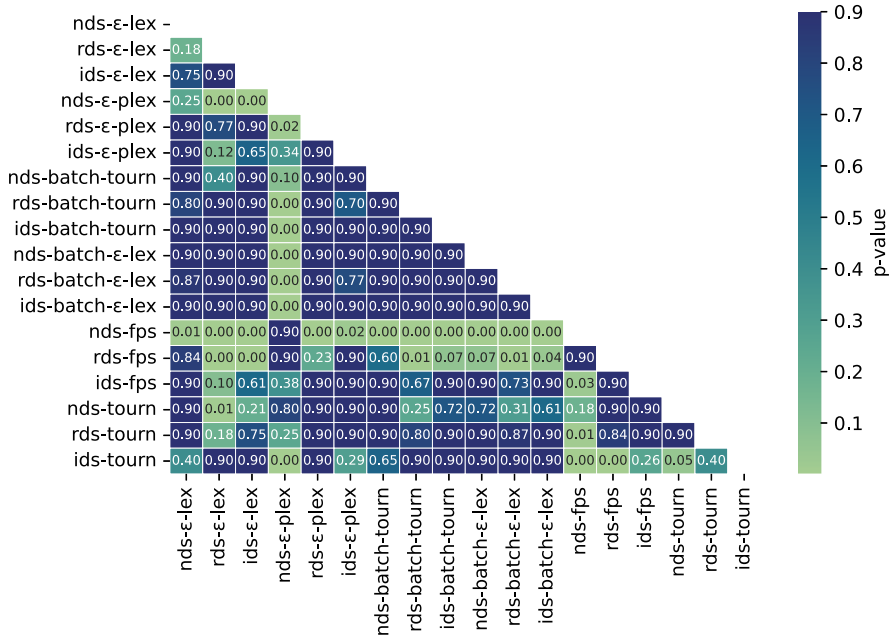

Fig. E1. Results of the Nemenyi test for the results *given a fixed evaluation budget*. p-Values are rounded to two decimal places.

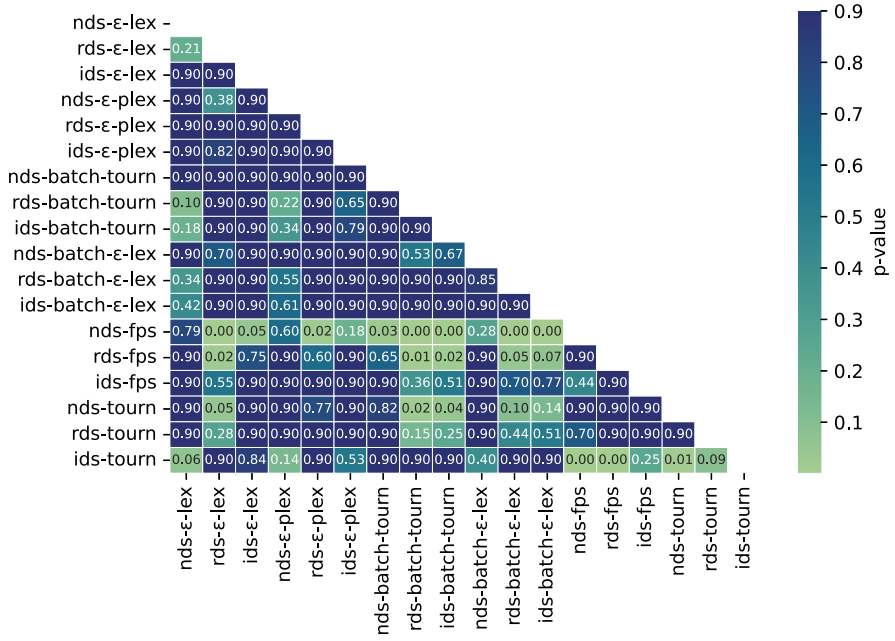

Fig. E2. Results of the Nemenyi test for the results *given 24 h*. p-Values are rounded to two decimal places.

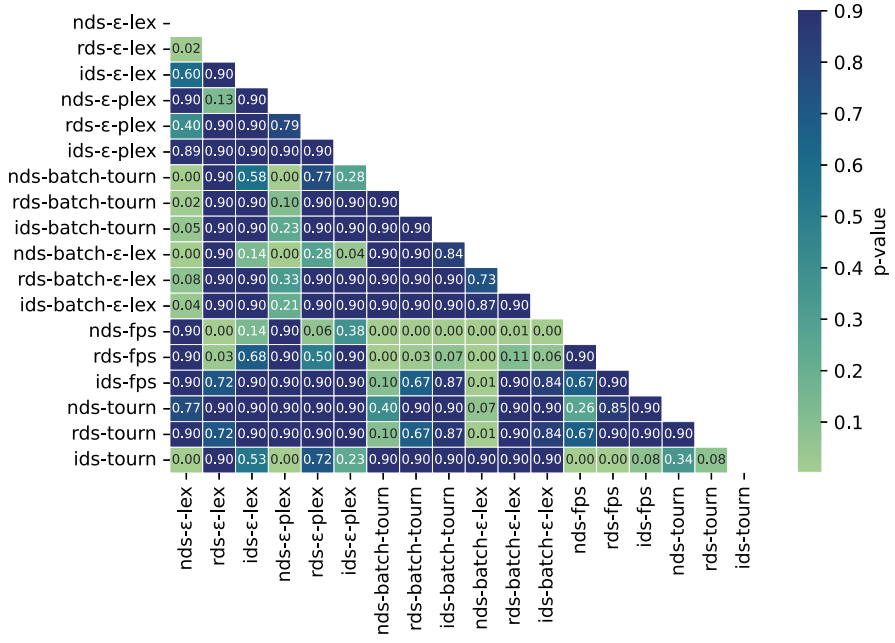

Fig. E3. Results of the Nemenyi test for the results *given 1 h*. p-Values are rounded to two decimal places.

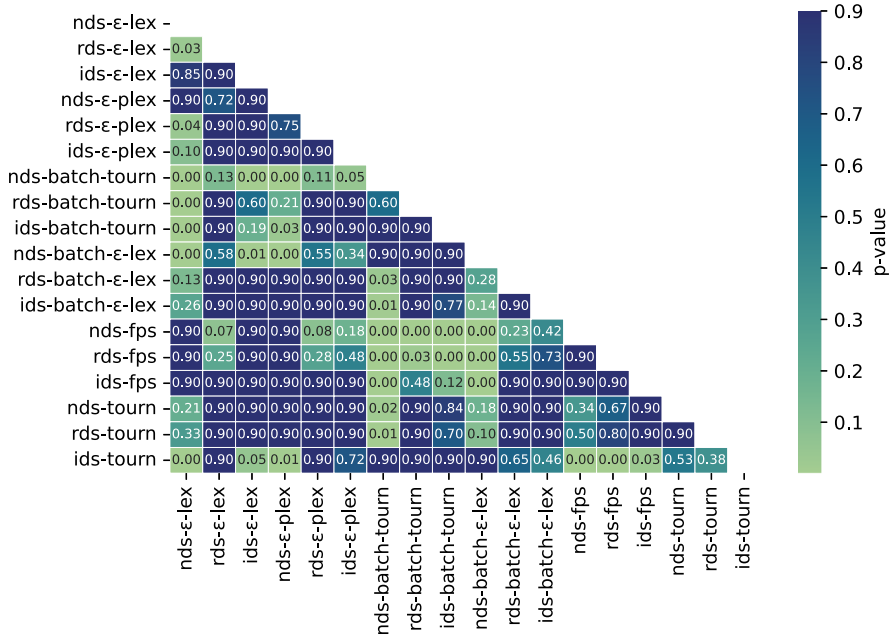

Fig. E4. Results of the Nemenyi test for the results *given 15 minutes*. p-Values are rounded to two decimal places.

Received 2 August 2024; revised 10 March 2025; accepted 17 March 2025