Analyzing the influence of Selection on Genetic Programming's Generalization ability in Symbolic Regression:

A comparison of ϵ-Lexicase Selection and Tournament Selection

---

Student: Roman Höhn

Date of Birth: 1991-04-14

Place of Birth: Wiesbaden, Hesse

Student ID: 2712497

Supervisor: David Wittenberg

Research Project for Seminar Information Systems (03.996.3299)

Chair of Business Administration and Computer Science

Johannes Gutenberg University Mainz

Summerterm 2022

Date of Submission: 2022-06-22

Contents

1 Introduction

Genetic programming (GP), a subfield of evolutionary computation (EC), is a metaheuristic that is used to search for computer programs that solve a given problem by simulating the process of darwinian evolution. The basic principle of GP is to gradually evolve solutions by repeatedly selecting parent solutions from a randomized population of computer programs based on a fitness metric. Then, genetic operators are applied on the selected solutions to genereate new offspring candidate solutions. By repeating this process over many generations, GP acts as a guided search for high fitness solutions throughout the decision space. A unique feature of GP among other evolutionary optimization procedures is the possibility to evolve solutions of variable length.

The overall performance of GP can depend strongly on the choice of its underlying operators, one crucial component in this is the operator for parent selection. Tournament Selection is a commonly used selection operator in EC and is the most used operator in GP systems (Fang and Li, 2010, p. 181). A parent solution is selected by randomly sampling k individuals from the current population into a tournament pool and then the solution with the highest fitness score from the tournament pool is selected (Fang and Li, 2010, p. 182). Lexicase selection has been suggested as an alternative to tournament selection that is not based on aggregating fitness scores. It samples n test cases in random order and then eliminates solutions from the selection pool on a per test case basis if they are not performing on an elite level (Helmuth, Spector and Matheson, 2015, p. 1).

Since regular Lexicase selection has been shown to perform suboptimal on continous-valued optimization problems, a modified variation called ϵ-Lexicase selection has been suggested for symbolic regression by La Cava et. al (2016). Here, ϵ-Lexicase selection has shown itself to outperform both in overall performance while showing only negligible computational overhead (La Cava, Spector and Danai, 2016, p. 747).

The goal of symbolic regression is to find a mathematical model for an observed set of datapoints (Paris, Robilliard and Fonlupt, 2004, p. 794). Symbolic Regression has been one of the first GP applications and to this day is an actively studied and highly relevant area of research (Poli, Langdon and McPhee, 2008, p. 114). In most symbolic regression problems, little to no a priori knowledge about the optimal form and structure of the target function is available. The ability of GP to optimize for model structure as well as for parameters has lead to it being one of the most prevalent methods used in the domain of symbolic regression (Paris, Robilliard and Fonlupt, 2004, p. 795).

An important quality of all supervised machine learning applications, including GP, is the ability to not only optimize performance for the test cases a model is trained on but to also perform well on previously unseen cases, this is refered to as generalization. In most real world applications of symbolic regression only a small subset of labeled data is available for training. The aim is to produce a model that not only accurately predicts the provided training data but can also predict previously unseen cases with high precision (Gonçalves, 2016, p. 6). A model that is extensivley optimized on the provided training data, may overfit to this data sample which may lead to a decrease in generalization.

This research project tries to answer the question if the usage of ϵ-Lexicase selection influences the generalization behaviour of programs that are evolved using GP for symbolic regression if compared to programs that are evolved using traditional tournament selection.

2 An Overview of the current state of research

2.1 Selection

In the first description of lexicase[1] selection for GP by Spector (2012) it was suggested as a novel parent selection method for target problems that are modal in nature. The author classified modal problems as "problems that qualitatively different modes of response are required for inputs from different regions of the problem's domain"(Spector, 2012, p. 1).

In a following article Lexicase selection has been proposed specifically for the purpose of solving so called uncompromising problems with GP. Helmuth, Spector and Matheson (2015) defined uncompromising problems as problems that require the final solution to perform optimal on each of the cases it is tested on, examples include symbolic regression, the design of digital multipliers or finding terms in finite algebras. The authors also provided evidence that lexicase selection can significantly improves GP's ability to solve some uncompromising problems if compared to selection methods that are based on aggregated fitness (Helmuth, Spector and Matheson, 2015, p. 12).

Both articles argued that many of the problem domains that GP is commonly used for are prevalent for problems that are modal/ uncompromising in nature. In comparison to other selection operators, populations that are evolved using variations of the lexicase selection operator show a very high degree of genetic diversity which might be a key contributer to the improved performance (Helmuth, Spector and Matheson, 2015, p. 1) (La Cava, Spector and Danai, 2016, p. 745). In theory, Lexicase based selection should be more prone to select solutions that are specialist (high fitness on a small subset of training cases) than solutions that are generalist (high average fitness among all training cases).

The first description[2] of the lexicase parent selection algorithm by Spector (2012) is quoted below for reference:

```
Lexicase - Parent-Selection:

1. Initialize:
  (a) Set candidates to be the entire population.
  (b) Set cases to be a list of all of the fitness cases in random order.

2. Loop:
  (a) Set candidates to be the subset of the current candidates that have
      exactly the best fitness of any individual currently in candidates for
```

[1] The alternative, more descriptive name given by Spector (2012): global pool, uniform random sequence, elitist lexicase parent selection

[2] Spector (2012), p.4

```
     the first case in cases.
  (b) If candidates or cases contains just a single element then return
     the first individual in candidates.
  (c) Otherwise remove the first case from cases and go to Loop.
```

More recent research by La Cava, Spector and Danai (2016) suggested ϵ-Lexicase selection as a modified selection operator that can improve overall GP performance if applied to continous-valued symbolic regression tasks in comparison to tournament and standard lexicase selection. La Cava, Spector and Danai (2016) argued that the original concept behind lexicase selection as formulated by Spector (2012) does not fit the requirements of real-world symbolic regression tasks. The authors identified the pass condition of regular lexicase selection as the main problem if applied to symbolic regression: Individual solutions can only be selected if they perform on an elite level but with noisey and continous-valued data it is very unlikely for two individuals to achieve an exactly equal error on a test case, this finally results in filtering out too many individuals during the selection process(La Cava, Spector and Danai, 2016, p. 742). ϵ-Lexicase adresses this problem by introducing an ϵ parameter that specifies a range around the elite error, individuals that perform inside this range pass the current selection iteration. La Cava *et al.* (2017) describe ϵ-Lexicase selection as a more “relaxed version of lexicase selection”. Different methods to configure the ϵ Parameter have been explored, for the limited scope of this project I will focus on the most promising implementation of ϵ-Lexicase selection that is automatically adjusted on the basis of the median absolute deviation of errors inside the selection pool (La Cava, Spector and Danai, 2016, p. 742) (La Cava *et al.*, 2017, p. 6).

The performance increase of ϵ-Lexicase selection for symbolic regression problems has been demonstrated and reported for many benchmark problems which led to widespread adaption of it in symbolic regression applications (La Cava, Spector and Danai, 2016, pp. 744–745).

In comparison, traditional GP selection methods such as tournament selection most commonly compute the fitness of a program as the mean of its error for each individual fitness case. One downside to this approach is that the total amount of information is reduced from a wide range of individual errors to a single metric. Helmuth, Spector and Matheson (2015) suspected that this loss in overall information provided to GP might reduce overall performance especially if applied to the class of uncompromising problems that require the solution to perform well on a wide range of diverse cases. Since tournament selection selects individuals based on their average fitness across all test cases, the resulting solutions should be expected to be more biased towards being generalists instead of specialists.

The basic algorithm for tournament is given below for comparison (Fang and Li, 2010, pp. 182–183):

```
Tournament - Parent-Selection:

k = tournament size

1. Sample:
  (a) Randomly sample k individuals from the current population
     into a tournament pool.
```

```
2. Select:
  (a) Compute the mean fitness for each individual inside the tournament pool
      based on all fitness cases.
  (b) Select individual from the tournament pool with the highest mean fitness.
```

2.2 Generalization

Generalization, the ability of a model to perform well on previously unseen cases, is one of the fundamental goals in most real world machine learning applications. O'Neill *et al.* (2010) raised awareness to the fact, that the topic of generalization in GP has not gotten the attention that other machine learning-based areas, e.g, deep learning, have attributed to it. A similiar statement was proposed by Kushchu (2002) in his review of research in generalization in GP. Both authors adressed the need for additional research regarding the topic.

Kushchu (2002) critized that almost all applications published in the initial GP literature by John Koza (Koza, 1992) are not using separate training and testing datasets which might result in overfitting and a poor overall generalization ability of the programs that are produced. The author calls for a more widespread adoption of methods like generational sampling of new training cases or the overall separation into training and testing datasets to improve generalization in GP (Kushchu, 2002, p. 10).

An aspect that is suspected to negatively correlate with GP's generalization is overall size and in particular bloating of the resulting programs. Bloating can be described as a growth in the total size of a program that does not also improve it's performance in any meaningful way (Silva and Vanneschi, 2009, p. 1). It has been widely suspected that GP configurations that produce very large programs also have a higher tendency to specialize on difficult or unusual test cases which could result in lower generalization (Wang, Wagner and Rondinelli, 2019, p. 268). The minimum description length principle (MDLP) describes this phenomenom by claiming that less complex models are more likely to perform better at generalizing than more complex models that achieve a comparable fitness during the training phase (O'Neill *et al.*, 2010, p. 349). However MDLP should be considered carefully, experiments by Silva and Vanneschi (2009) on the prediction of bioavailability for drugs demonstrated that GP based techniques can either bloat without overfitting or overfit without bloating (Silva and Vanneschi, 2009, p. 8).

2.3 Symbolic Regression

The task of finding a mathematical function that fits a given set of datapoints has been one of the first applications of GP in (Koza, 1992, p. 238). Practical examples given by Koza (1992) included tasks such as the discovery of scientific laws, finding solutions to differential and integral equations or the programmatic compression of images.

Since it's inception, GP based symbolic regression has been widely used in practical application ranging over a wide field of disciplines. A well known example of an industrial use-case has been published by Jordaan *et al.* (2004). The authors used GP based symbolic regression to derive nonlinear functions that could be used to significantly increase the robustness of industrial sensors which resulted in wide-spread adoption and a significant reduction in costs.

Many variations of GP based symbolic regression have been proposed to improve overall performance, Wang, Wagner and Rondinelli (2019) summarized the main characteristics and differences for 6 common variations including procedures such as geometric semantic genetic programming, cartesian genetic programming or GP-based relevance vector machines. A large benchmarking study on symbolic regression published by Orzechowski, Cava and Moore (2018) compared 4 GP-based procedures to several state-of-the-art machine leaning techniques. The authors find that GP, eventhough more time consuming, can achieve better results if compared to the state-of-the-art machine learning algorithm gradient-boosting.

3 Experimental study

3.1 Research Design

To study the influence of selection on generalization I selected a dataset about energy efficiency in buildings that is part of the UC Irvine Machine Learning Repository (Dua and Graff, 2017). The dataset contains eight individual building attributes that map to two different outcomes, heating and cooling load, for N=768 cases (Tsanas and Xifara, 2012).

Using two otherwise identical GP systems, one deploying tournament selection and the other ϵ-lexicase selection, the objective is to find a computer program that best predicts the outcome variable heating load ($Y1$)[3] of the buildings using a subset of the eight building attributes ($X1, .., X8$) for input. The specific meaning of all attributes inside the dataset are described in table 1.

Table 1: Overview - Energy Heating Dataset

Variable	**Description**
X1	Relative Compactness
X2	Surface Area
X3	Wall Area
X4	Roof Area
X5	Overall Height
X6	Orientation
X7	Glazing Area
X8	Glazing Area Distribution
y1	Heating Load
y2	Cooling Load

To measure the generalization ability of each model the dataset will be randomly split in half, resulting in a training and testing dataset each containing 384 individual cases. Each model will be evolved by traditional GP using only the fitness cases present in the training

[3] The symbolic regression is performed on one of the two provided outcome variables, the variable Cooling Load will be excluded.

dataset. For each generation the highest fitness model of the current population will be tested with the previously unseen fitness cases that are part of the testing dataset. For each run of the experiment statistics will be collected on fitness which will form the basis of my further statistical analysis. Additional statistics will be collected for the average length of the population and the length of the elite program for each generation to explore differences in their distribution and possible correlations to generalization.

Since GP is a stochastic optimization algorithm, the basic experiment will be run for a total of 50 times to ensure a fair and meaningful comparison based on a large number of runs for both algorithms. For each run of the experiment the dataset will be randomly split in half as described, both models are then trained and tested using the exact same set of fitness cases.

The statistical analysis of the collected data will first focus on examining the question if, on average, the usage of ϵ-Lexicase selection will result in models that perform significantly different than models that are evolved using tournament selection both.

> $H0_1$: The distribution underlying the samples of training/testing errors produced by torunament selection is the same as the distribution underlying samples of training/testing errors produced by ϵ-lexicase selection

The next question of interest is, to examine if statistical significant differences between the mean errors of training and testing data exist for both selection operators:

> $H0_2$: The distribution underlying the samples of training errors produced by ϵ-Lexicase/torunament selection is the same as the distribution underlying samples of testing errors produced by ϵ-Lexicase/tournament selection

To gather further insight into the differences between both GP systems, an additional test will be performed on the question if differences in the total size of the resulting programs exist:

> $H0_3$: Size differences exist between the distribution underlying the samples produced by ϵ-Lexicase and the distribution underlying samples of tournament selection

All hypothesis will be tested using a level of significance of $\alpha = 0.05$.

To further examine the difference in generalization behaviour, the mean testing and training errors over each generation will be visualized for both algorithms. The specific aim of this visualization is to explore the mechanism of overfitting and to examine if differences between both methods can be detected.

All GP experiments will be implemented by using the python programming language in conjunction with DEAP, a framework for distributed evolutionary algorithms that implements various tools and algorithms for genetic programming (Fortin *et al.*, 2012).

3.2 Gentetic Programming Configuration

3.2.1 Evolutionary parameters

The basic evolutionary parameters for both systems are presented in table 2. Tournament selection is used with a default tournament size of 3 individuals while the ϵ parameter in lexicase selection is selected automatically as previously mentioned in subsection 3.1.

Table 2: Evolutionary Parameters

Parameter	**Value**
Population Size	500
Number of Generations	100
Mutation Rate	20%
Crossover Rate	80%
Tournament Size	3
Epsilon selection	automatic
Elite Size	0

3.2.2 Fitness Evaluation

The fitness f for each model will be based on the mean squared error (MSE) over all fitness cases for prediction and the empirically measured values as described by eq.1.

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (Y_i - \hat{Y_1})^2 \tag{eq. 1}$$

where:

- n: total number of test cases
- Y_i: empirical value for case i
- $\hat{Y_1}$: predicted value for case i.

The resulting fitness function f for an individual program i is descibed by eq 2:

$$f(i, \tau) = \frac{1}{N} * \sum_{t \epsilon \tau} (y_t - \hat{y_t}(i, x_t))^2 \tag{eq. 2}$$

where[4]:

- τ: the set of N fitness cases
- y_t: empirical value of the target for case t
- $\hat{y_t}(i, x_t)$: predicted value for the target for case t by running the program i with the total set of input variables x_t

[4] Naming and notation was adopted from La Cava, Spector and Danai (2016)

3.2.3 Primitive Set

The primitve set consists of the terminals that are listed in table 3 and the functions that are listed in table 4. To avoid runtime errors I implemented protected version of the operators for division, natural logarithm and square root (Koza, 1992, pp. 82–83). As suggested by Koza (1992) I also included ephemeral constants to the set of terminals to provide the evolutionay search the opportunity to explore and include randomly generated constants.

Table 3: Terminals

Terminal	**Description**
X1	Relative Compactness
X2	Surface Area
X3	Wall Area
X4	Roof Area
X5	Overall Height
X6	Orientation
X7	Glazing Area
X8	Glazing Area Distribution
random_int	Ephemeral Constant (integer)
random_float	Ephemeral Constant(float)

Table 4: Functions

Function	**Arity**
Addition	2
Subtraction	2
Multiplication	2
Negation	1
Sine	1
Cosine	1
Protected Division	2
Protected Natural Logarithm	1
Protected Square Root	1

3.2.4 Genetic Operators

GP operators are represented in table 5. Both genetic operators use a static limit to control for the height of the reulting trees (Koza, 1992, p. 104). Individual programs are initialized by using the ramped half-and-half method, 50% of the population are created by using the Growth algorithm and the remaining 50% are created by using the Full algorithm (Koza, 1992, p. 93).

The crossover operator implemented by DEAP randomly selects a crossover point in each individual and exchanges each subtree with the point as root between each individual (Fortin

Table 5: GP Operators

Operator	**Implementation**	**Static.Height.Limit**
Initilization	Ramped Half/Half	2
Crossover	One Point Crossover	17
Mutation	Uniform Mutation	17

et al., 2012). Mutation also randomly selects a point in the tree individual, it then replaces the subtree below that point as a root by the expression generated using the full grow initialization method (Fortin *et al.*, 2012).

4 Results

Tables 6 and 7 summarize the results for all fitness scores collected over 50 total runs of the experiment[5].

Table 6: Summary - Tournament Selection

X	gen	nevals	mean_training_error	std_training_error	min_training_error	max_training_error	testing_error	std_testing_error	avg_size	elite_size
count	101.0	101.000	1.010000e+02	1.010000e+02	101.000	1.010000e+02	100.000	100.000	100.000	100.000
mean	50.0	420.742	2.092614e+18	3.305357e+20	16.483	1.046309e+21	16.905	28.858	55.674	62.070
std	29.3	8.115	2.098028e+19	3.313954e+21	12.514	1.049016e+22	11.117	15.732	34.504	36.850
min	0.0	417.300	2.582030e+08	1.463390e+10	7.066	1.193720e+11	7.847	16.217	3.331	4.340
25%	25.0	418.780	2.151894e+10	3.044239e+12	8.479	1.070232e+13	9.350	18.344	25.133	28.195
50%	50.0	420.000	3.494635e+11	5.511534e+13	11.918	1.747285e+14	12.874	22.715	57.299	66.400
75%	75.0	421.000	4.892936e+12	6.455802e+14	18.773	2.238973e+15	19.685	32.750	87.852	96.625
max	100.0	500.000	2.108540e+20	3.330558e+22	77.243	1.054272e+23	62.693	95.282	109.761	115.340

Table 7: Summary - Epsilon-Lexicase Selection

X	gen	nevals	mean_training_error	std_training_error	min_training_error	max_training_error	testing_error	std_testing_error	avg_size	elite_size
count	101.0	101.000	1.010000e+02	1.010000e+02	101.000	1.010000e+02	100.000	100.000	100.000	100.000
mean	50.0	420.781	7.533781e+12	1.186094e+15	22.013	3.762638e+15	22.456	39.048	58.416	63.819
std	29.3	8.109	5.900311e+13	9.313627e+15	16.670	2.950203e+16	15.904	19.225	34.515	36.356
min	0.0	416.240	4.660413e+05	3.314323e+07	9.680	1.539125e+08	11.129	21.487	3.219	4.280
25%	25.0	418.900	2.528193e+08	2.727451e+10	11.316	1.192373e+11	12.089	24.860	28.233	29.615
50%	50.0	420.040	2.196352e+09	2.647194e+11	14.478	1.098068e+12	15.223	30.057	64.689	71.440
75%	75.0	421.020	2.027020e+11	3.198174e+13	23.511	1.012976e+14	23.305	51.081	87.394	94.910
max	100.0	500.000	5.788911e+14	9.137151e+16	74.622	2.894465e+17	77.518	110.244	108.785	115.660

Some interesting observation from tables 6 and 7 can be made by comparing the mean of the variables *min_training_error* and *testing_error*. Variable *min_training_error* represents the fitness, measured by the MSE, of the best performing solution that has been found during a single run of the experiment while the variable *testing_error* represents the fitness of the same solution if evaluated for the unknown fitness cases inside the testing dataset. Tournament selection based GP achieves a mean *min_training_error* of 16.483 and a mean *testing_error* of 16.905 while the GP system using ϵ-Lexicase selection can only achieve a mean *min_training_error* of 22.013 and a mean *testing_error* of 22.456. These results indicate an overall performance benefit for torunament selection based GP in the experiment. It should also be mentioned that the best performing model over all 50 runs,

[5] All floating point numbers in tables have been rounded to 3 decimal places if not stated otherwise

measured by the minimum of variables $min\_training\_error$ and $testing\_error$, has been achieved by tournament selection with an MSE of 7.066 for the training data and 7.847 for the testing data.

If we compare the differences of $min\_training\_error$ and $testing\_error$ on the basis of each algorithm, we can observe that both tournament and ϵ-lexicase achieve a slightly lower error on the training data than on the testing data. If we compute the total difference between the means of $min\_training\_error$ and $testing\_error$ for both algorithms tournament selection amounts to 0.422 and ϵ-Lexicase selection to 0.443. Regarding our primary research question, the differences in generalization, the initial observations do not seem to indicate a significant difference in the the relative gap between the solutions performance on testing data and training data.

Figure 1 visualizes the distribution of $min\_training\_error$ and $testing\_error$ for both selection operators over the total 50 runs using a boxplot and summerizes the initial observations from tables 6 and 7 described above.

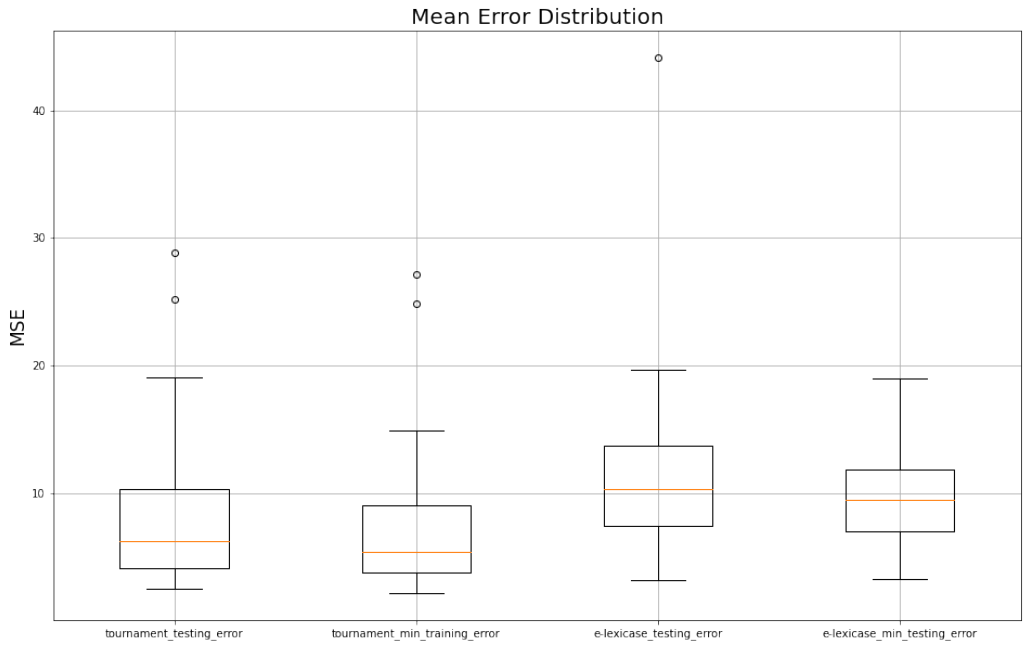


Figure 1: Distribution of Errors

The four samples from figure 1 have been tested for normal distribution by computing the D'Agostino and Pearson test (D'Agostino, 1971) (D'Agostino and Pearson, 1973) to determine if they satisfy the requirements of the students t-test. The results are summarized in table 8. Since not all samples test positive for a normal distribution a Mann-Whitney U (MWU) ranksum test will be performed to test for statistical significance between the four variables $tournament\_min\_training\_error$, $tournament\_testing\_error$, $e - lexicase\_min\_training\_error$ and $e - lexicase\_testing\_error$.

Table 8: Normal Distribution Tests

sample	**statistic**	**p.value**	**alpha**	**normal_distributed**
tournament_min_training_error	36.146	0.000	0.05	False
e-lexicase_min_testing_error	1.575	0.455	0.05	True
tournament_testing_error	32.503	0.000	0.05	False
e-lexicase_testing_error	59.135	0.000	0.05	False

Table 9 displays the resulting p-values from computing the MWU test in a matrix format.

Table 9: Mean Error - P-Values

	tournament_training_errors	**tournament_testing_errors**	**elexicase_training_errors**	**elexicase_testing_errors**
tournament_training_errors	1.000	0.309	0.000	0.000
tournament_testing_errors	0.309	1.000	0.002	0.000
elexicase_training_errors	0.000	0.002	1.000	0.257
elexicase_testing_errors	0.000	0.000	0.257	1.000

The results show two major findings:

1. The differences in mean fitness between tournament selection and ϵ-Lexicase selection are highly statistical significant even if considering an α of 0.01. For the given symbolic regression task tournament selection achieves a better mean performance on both the training as well as the testing dataset than ϵ-Lexicase selection. These findings contradict the hypothesis $H0_1$ descibed in subsection 3.1 and are contrary to some of the research that was summarized in subsections 2.1 and 2.3.(La Cava, Spector and Danai, 2016) (La Cava *et al.*, 2017).

2. The differences in mean fitness between $min\_training\_error$ and $testing\_error$ are not statistically significant for both selection algorithms. If comparing the mean training error to the mean testing error, the p-values only computes to 0.309 for tournament selection and to 0.257 for ϵ-Lexicase selection. These findings fail to reject $H0_2$, based on the obtained results we can not find evidence that both algorithms demonstrate a significantly different performance between the training data and the previously unseen testing data.

To further examine the generalization behaviour of both selection operators I plotted the mean fitness scores for both algorithms over the total number of 100 generations in figure 2. Figure 2 shows that both algorithms show little visual evidence of overfitting which would result in a growing gap between between $min\_training\_error$ and $testing\_error$ in the final stages of the evolutionary optimization. For ϵ-Lexicase selection a small gap can be observed starting at around the 90.th generation, while the minimum training error slowly converges a small increase in the testing error can be observed. Based on the limited amount of collected data, it is questionable if this small sign of overfitting can be interpreted as a weak indicator for an inferior generalization behaviour of ϵ-Lexicase in comparison to tournament selection.

Figure 2 also shows a visual representation of the described overall performance benefit of using tournament selection instead of ϵ-Lexicase selection for the choosen symbolic regression task. It can be observed that at all stages of the evolution tournament selection based GP achieves a lower mean error for both datasets.

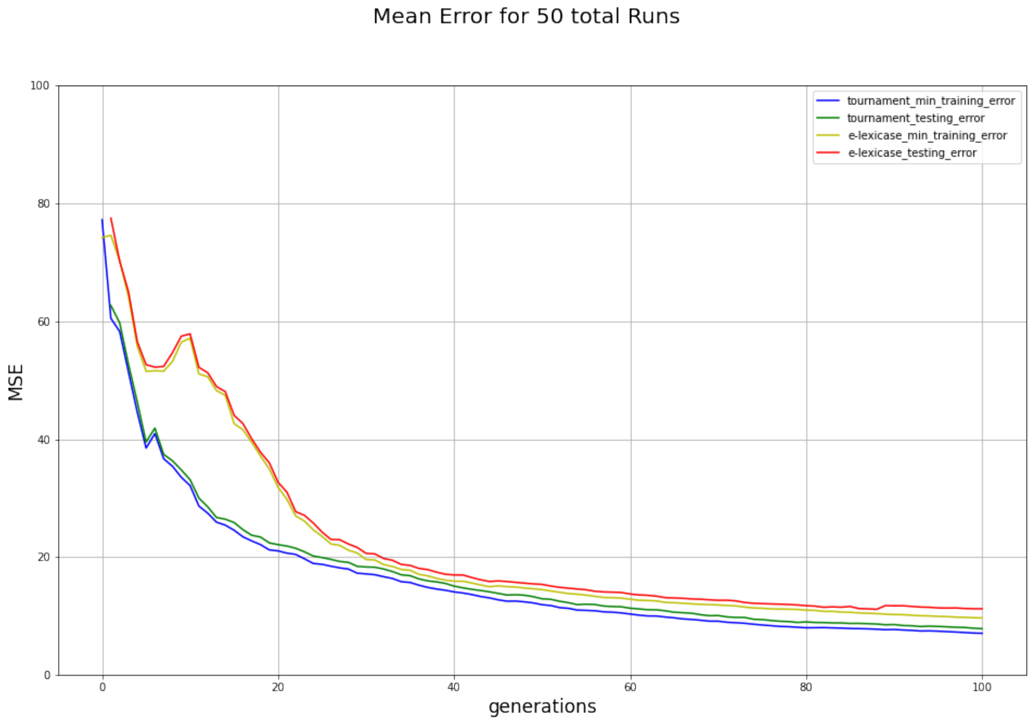
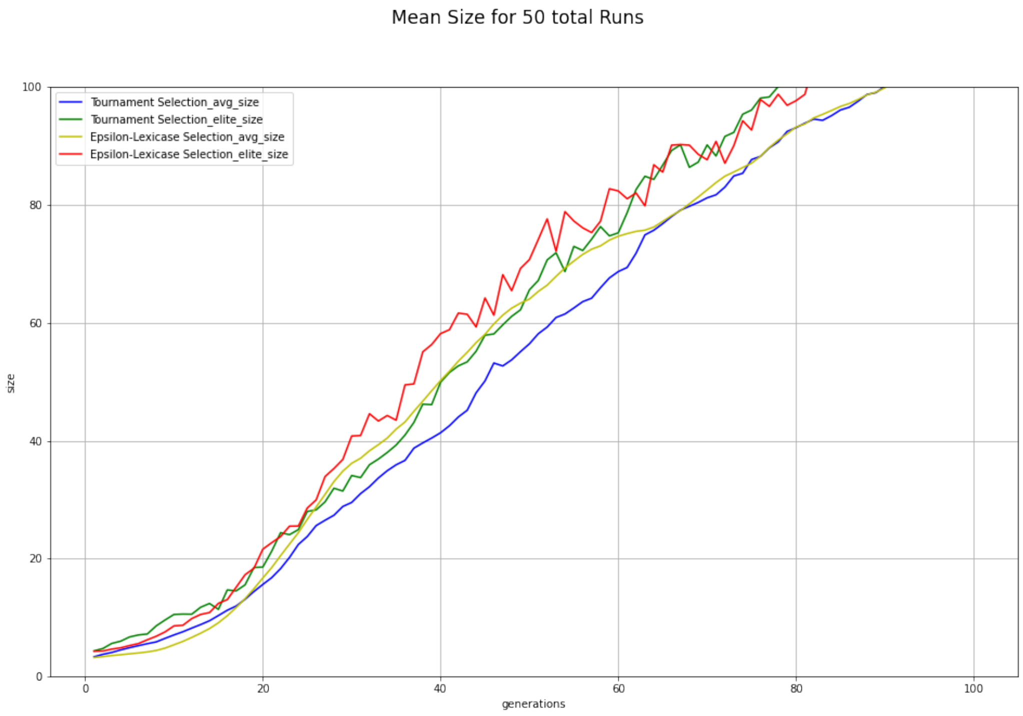


Figure 2: Mean Errors

In a final step I analyzed the growth behaviour for both GP systems. Figure 3 visualizes the average size of individuals inside the population as well as the average size of the best performing individual for each generation of the evolution.



. . . interpretation. . .

Again, a MWU test is conducted to test for statistical differences in the underlying distribution of the sizes measured during the experiment. The results are summarized in table 10.

Table 10: Mean Size - P-Values

	tournament_elite_size	**elexicase_elite_size**	**tournament_avg_size**	**elexicase_avg_size**
tournament_elite_size	1.000	0.858	0.533	0.652
elexicase_elite_size	0.858	1.000	0.764	0.682
tournament_avg_size	0.533	0.764	1.000	0.992
elexicase_avg_size	0.652	0.682	0.992	1.000

5 Conclusion

6 Limitations and open questions

I faced two major difficulties in the preparation and evaluation of this research project:

1. Configuration of evolutionary parameters
2. Limited computational ressources

The combination of both issues might have resulted in a significant reduction in overall robustness of the results represented in sections 5 and 6. The performance of all evolutionary algorithms, including GP, can be highly dependant on a large number of parameters and implementation details. Although the series of experiments conducted in this project are comparably simple in nature, the results achieved might still be highly influenced by the GP configuration detailed in subsection 4.2.

To gain further insight into the differences in generalizaion behaviour and to adress the research question with a higher level of certainty, the experiment I conducted should be repeated for many different GP configurations. Some exemplary parameters that could drastically influence the results of this experiment include the population size, the total number of generations, the inclusion of elitism or the computation of ϵ in lexicase selection. Other factors whose influence on generalization behaviour could be important include different methods to compute an individuals fitness (e.g. mean absolute error or root mean squared error) and testing out different variations of the genetic operators for mutation and crossover.

Another obvious weakness of my project is that the experiment is only based on a single symbolic regression application, the prediction of the heating load of buildings based on a relativley small dataset. To gain further trust in the obtained results, the experiment should be repeated for other symbolic regression tasks and datasets. At the current state it remains unknown if the generalization behaviour of both selection operators might be highly problem specific and dependant on the dataset used for training and testing.

All limitations and shortcomings described above are also connected to the second limitation of this research project, the limited amount of computational ressources. The total computation time consumed to run the current experiments (50 runs, 2 algorithms, 100 generations, 500 individuals) has already been close to 24 hours on a modern, arm-based Apple-M1 system. Eventhough the overall computation time could probably be reduced by further optimization of the source code, e.g. using tools for parallel computation, the ressources necessary to repeat the whole experiment for different parameter configurations would certainly be beyond the scope of this research project.

I References

D'Agostino, R.B. (1971) "An omnibus test of normality for moderate and large size samples," *Biometrika*, 58, pp. 341–348.

D'Agostino, R.B. and Pearson, E.S. (1973) "Tests for departure from normality," in.

Dua, D. and Graff, C. (2017) "UCI machine learning repository." University of California, Irvine, School of Information; Computer Sciences. Available at: http://archive.ics.uci.edu/ml.

Fang, Y. and Li, J. (2010) "A review of tournament selection in genetic programming," in Cai, Z. et al. (eds.) *Advances in computation and intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 181–192.

Fortin, F.-A. *et al.* (2012) "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, 13, pp. 2171–2175.

Gonçalves, I. (2016) "An exploration of generalization and overfitting in genetic programming: Standard and geometric semantic approaches," in.

Helmuth, T., Spector, L. and Matheson, J. (2015) "Solving uncompromising problems with lexicase selection," *IEEE Transactions on Evolutionary Computation*, 19(5), pp. 630–643. doi:10.1109/TEVC.2014.2362729.

Jordaan, E. *et al.* (2004) "Robust inferential sensors based on ensemble of predictors generated by genetic programming," in Yao, X. et al. (eds.) *Parallel problem solving from nature - PPSN VIII*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 522–531.

Koza, J.R. (1992) *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press. Available at: http://mitpress.mit.edu/books/genetic-programming.

Kushchu, I. (2002) "An evaluation of EvolutionaryGeneralisation in genetic programming," *Artificial Intelligence Review - AIR*, 18, pp. 3–14. doi:10.1023/A:1016379201230.

La Cava, W. *et al.* (2017) "A probabilistic and multi-objective analysis of lexicase selection and epsilon-lexicase selection." arXiv. doi:10.48550/ARXIV.1709.05394.

La Cava, W., Spector, L. and Danai, K. (2016) "Epsilon-lexicase selection for regression," in *Proceedings of the genetic and evolutionary computation conference 2016*. New York, NY, USA: Association for Computing Machinery (GECCO '16), pp. 741–748. doi:10.1145/2908812.2908898.

O'Neill, M. *et al.* (2010) "Open issues in genetic programming," *Genetic Programming and Evolvable Machines*, 11, pp. 339–363. doi:10.1007/s10710-010-9113-2.

Orzechowski, P., Cava, W.L. and Moore, J.H. (2018) "Where are we now?" in *Proceedings of the genetic and evolutionary computation conference*. ACM. doi:10.1145/3205455.3205539.

Paris, G., Robilliard, D. and Fonlupt, C. (2004) "Exploring overfitting in genetic programming," in Liardet, P. et al. (eds.) *Artificial evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 267–277.

Poli, R., Langdon, W.B. and McPhee, N.F. (2008) *A field guide to genetic programming*. Published via `http://lulu.com`; freely available at `http://www.gp-field-guide.org.uk`. Available at: https://digitalcommons.morris.umn.edu/cgi/viewcontent.cgi?article=1001&context=cs_facpubs.

Silva, S. and Vanneschi, L. (2009) "Operator equalisation, bloat and overfitting: A study on human oral bioavailability prediction," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*, pp. 1115–1122. doi:10.1145/1569901.1570051.

Spector, L. (2012) “Assessment of problem modality by differential performance of lexicase selection in genetic programming: A preliminary report.” doi:10.1145/2330784.2330846.

Tsanas, A. and Xifara, A. (2012) “Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools,” *Energy and buildings*, 49, pp. 560–567. doi:10.1016/j.enbuild.2012.03.003.

Wang, Y., Wagner, N. and Rondinelli, J.M. (2019) “Symbolic regression in materials science,” *MRS Communications*, 9(3), pp. 793–805. doi:10.1557/mrc.2019.85.

II Statutory Declaration