Improving Interpretability in Symbolic Regression Models Using Multi-objective GP and the Transient Terminal Set

Asher Stout, 300432820

January 11, 2021

1 Overview of Work & Research

Work began on the project with research into previous approaches for improving the interpretability of Symbolic Regression models, and ultimately Multi-objective GP (MOGP) was selected as an area for further study. Several relevant papers defined the interpretability of a solution as the depth or size of its tree representation, which may not accurately correlate to its computational complexity. Thus, it was decided that a new measure of complexity be developed for use in MOGP. In addition, the entirely random nature of mutation operators in MOGP was identified as a place for potential innovation in improving Symbolic Regression model interpretability. This resulted in a new approach being developed called the **Transient Terminal Set**, which is described in further detail in **Section 2**.

Aside from research, extensive time was invested into learning the tools required for the completion of the project objectives - particularly Python, DEAP, and LaTeX. This involved practical applications in the form of tutorials and implementing a standard MOGP and culminated in the formal definition of the Transient Terminal Set algorithm (see **Section 2**) and its implementation using DEAP (see **Section 3**).

The total hours spent on the project, as of 11 January 2021, were **196**.

2 Transient Terminal Set

Algorithm 1 Multi-objective GP using the Transient Terminal Set (TTSGP)

Input: population size ρ, crossover probability p_c, mutation probability p_m, transient mutation probability p_d, terminal set T, function set F, death age α

Define: generation G_n, individual fitness f_i, transient terminal set M_{G_n}, fitness threshold f_{t,G_n}

```
1:  Initialize starting population $P_{G_0}$, $M_{G_0} \leftarrow \emptyset$, $f_{t,G_0} \leftarrow 0$
2:  while no improvement in $\max f_i \in P_{G_n}$ since $P_{G_{n-5}}$ do        ▷ Evolve generation $G_{n+1}$
3:      $P_{G_{n+1}} \leftarrow \emptyset$, $M_{G_{n+1}} \leftarrow M_{G_n}$
4:      while len$P_{G_{n+1}} \neq \rho$ do        ▷ Update population $P_{G_{n+1}}$
5:          Perform crossover $\forall i \in P_{G_n}$ with $p_c$
6:          Perform mutation $\forall i \in P_{G_n}$ with $p_m$, $T$, $F$
7:          Perform transient mutation $\forall i \in P_{G_n}$ with $p_d$, $M_{G_n}$
8:          $P_{G_{n+1}} \leftarrow P_{G_{n+1}} \cup \{i | i_{offspring}\}$

9:      for all subtree $s \in M_{G_{n+1}}$ do        ▷ Update transient terminal set $M_{G_{n+1}}$
10:         if $age(s) > \alpha$ then
11:             Prune $s$ from $M_{G_{n+1}}$
12:     Compute $f_{t,G_n}$ from $\forall f_i \in P_{G_{n+1}}$
13:     for $i \in P_{G_{n+1}}$ do
14:         $f_c \leftarrow \Delta f_i$ from $G_n$ to $G_{n+1}$
15:         if $f_c > f_{t,G_n}$ then
16:             $M_{G_{n+1}} \leftarrow M_{G_{n+1}} \cup \{$subtree $s \in i\}$
```

Note: The transient terminal set is utilized during a genetic operation called *transient mutation*, in which a candidate solution is mutated with a member of the set. The transient terminal set is composed of subtrees generated in the population (either through crossover or normal mutation) which have resulted in substantial increases in the fitness of candidate solutions.

The Transient Terminal Set seeks to improve the interpretability of Symbolic Regression models via the use of multi-objective GP by improving the search process itself. By utilizing a complexity measure in addition to an error measure as the Pareto-efficient objectives for the algorithm, and pairing this with the proposed transient terminal set, it is theorized that candidate solutions will become less complex when compared with standard multi-objective GP. As the selection process for the transient terminal set follows this multi-objective framework, improvements in either objective will result in a candidate solution's altered subtree being added to the set. Thus, the transient terminal set distributes proven subtrees that result in lower errors and/or complexities throughout the population. This process potentially results in candidate solutions with both minimized error and complexity measures, and is an improvement over the entirely randomized mutation of standard multi-objective GP.

3 Experimentation Results

Note: The code for this experiment can be accessed at `https://github.com/VeryEager/transient-terminal-gp`.
The experiments utilized the following datasets: Red Wine Quality, White Wine Quality, Concrete Strength, Boston House Price, and Bike Rentals (preprocessed). No additional preprocessing was performed on the datasets prior to experimentation, with the exceptions of the removal of a constant feature in the Boston House Price data and the removal of the 'datetime' feature in the Bike data. Both these features were determined irrelevant.

Figure 1: Evolution using Standard MOGP, winequality-red

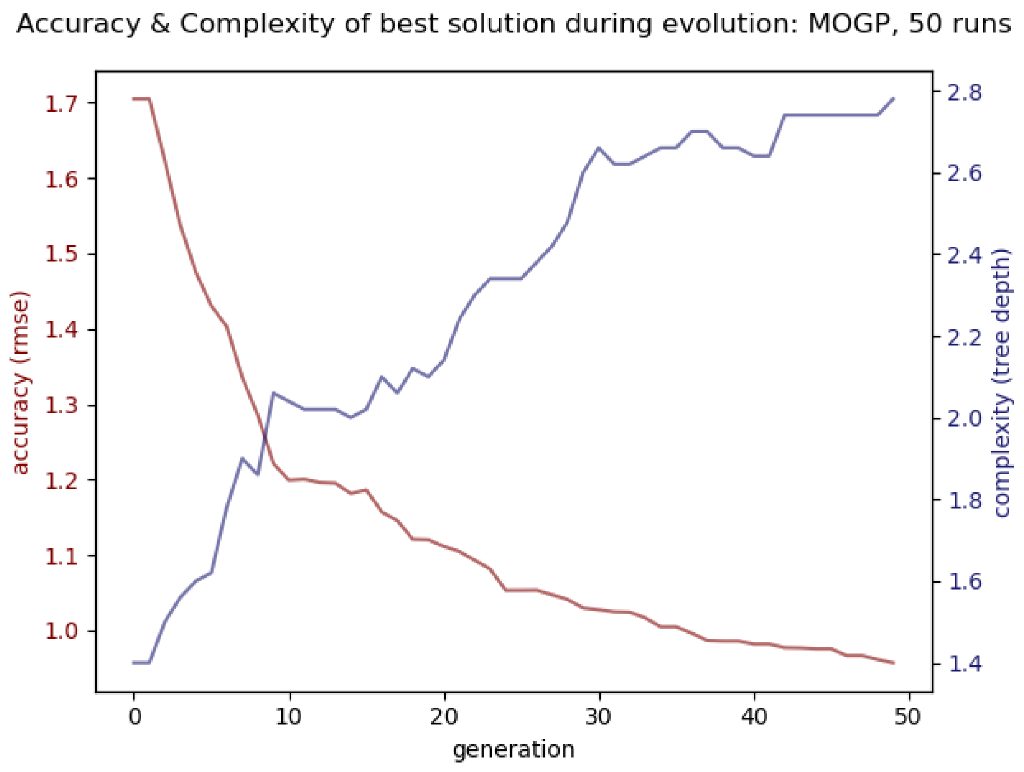


Figure 2: Evolution using Transient Terminal Set GP, winequality-red

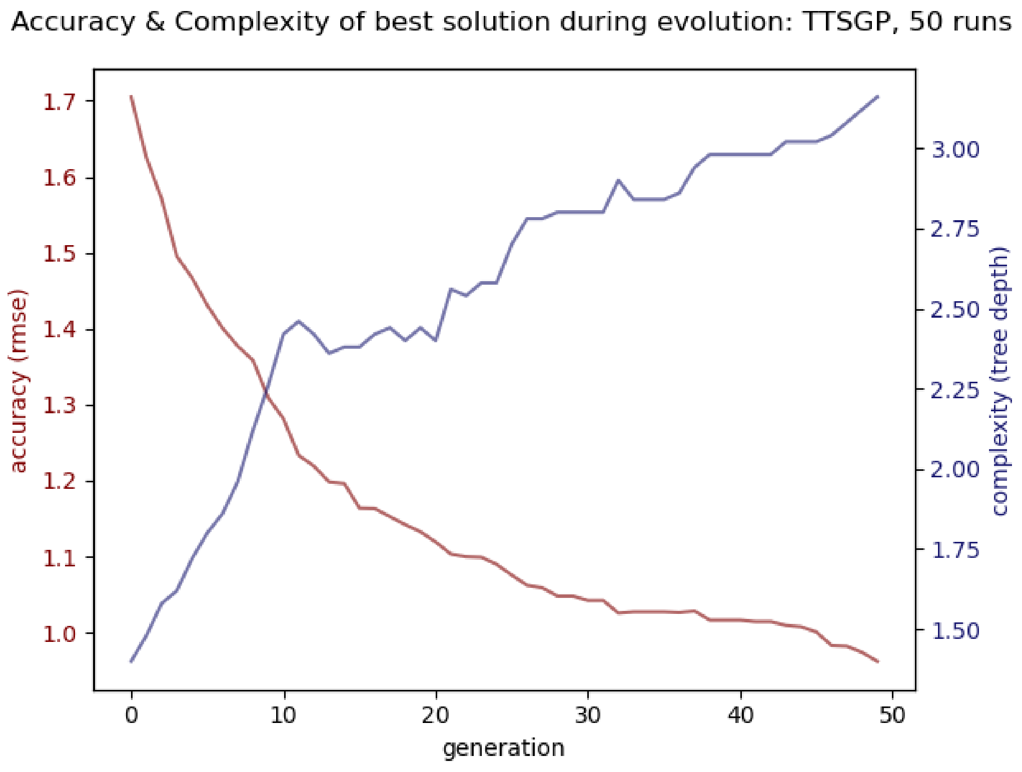


Figure 3: Evolution using Standard MOGP, winequality-white

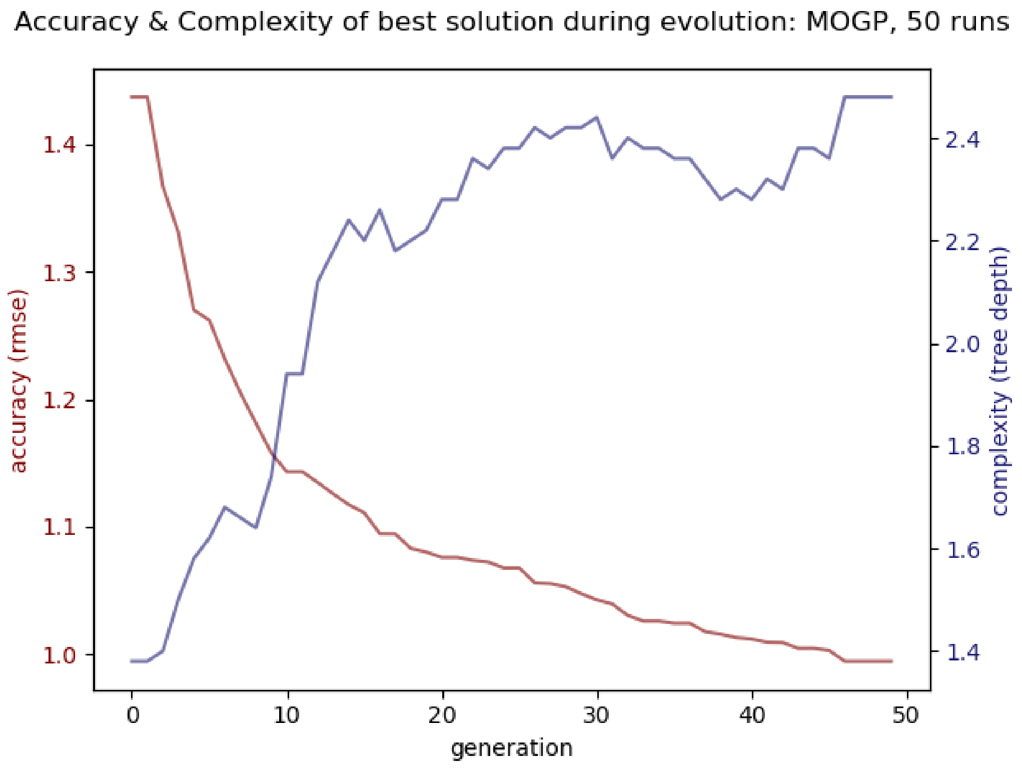


Figure 4: Evolution using Transient Terminal Set GP, winequality-white

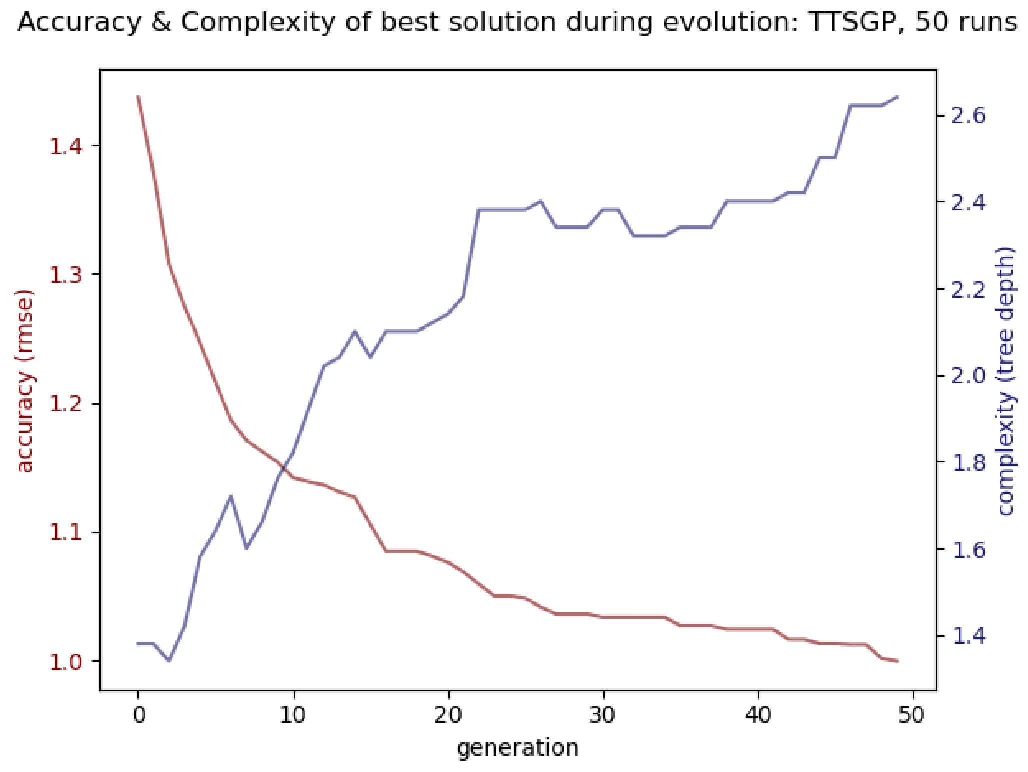


Figure 5: Evolution using Standard MOGP, strength-concrete

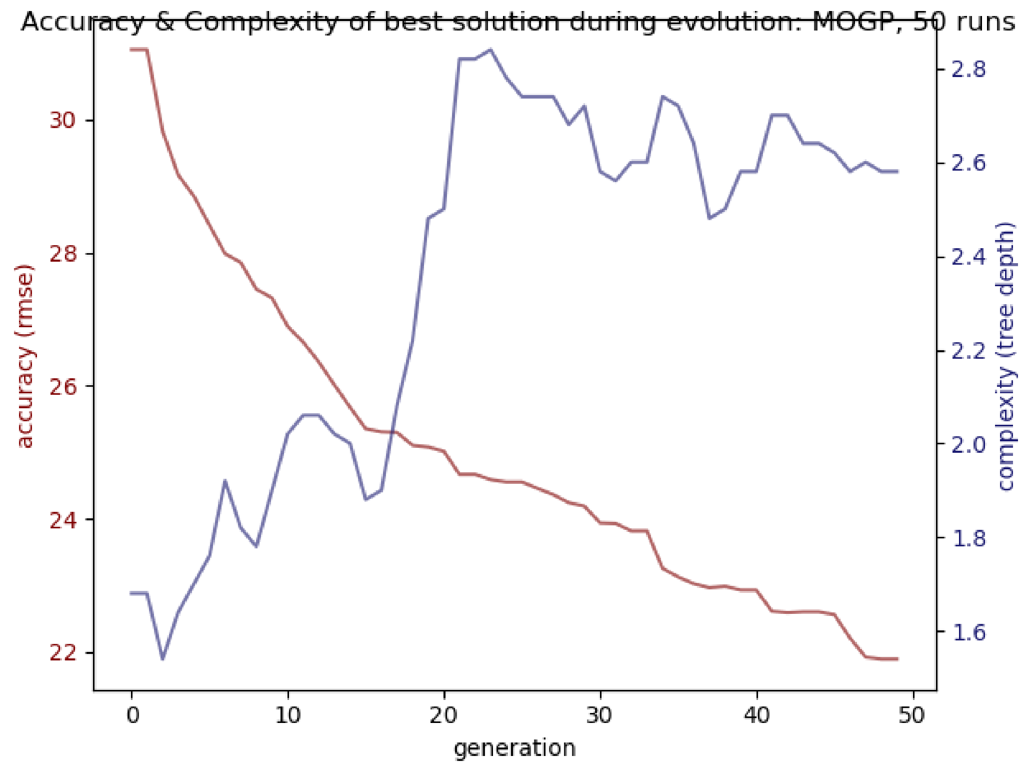


Figure 6: Evolution using Transient Terminal Set GP, strength-concrete

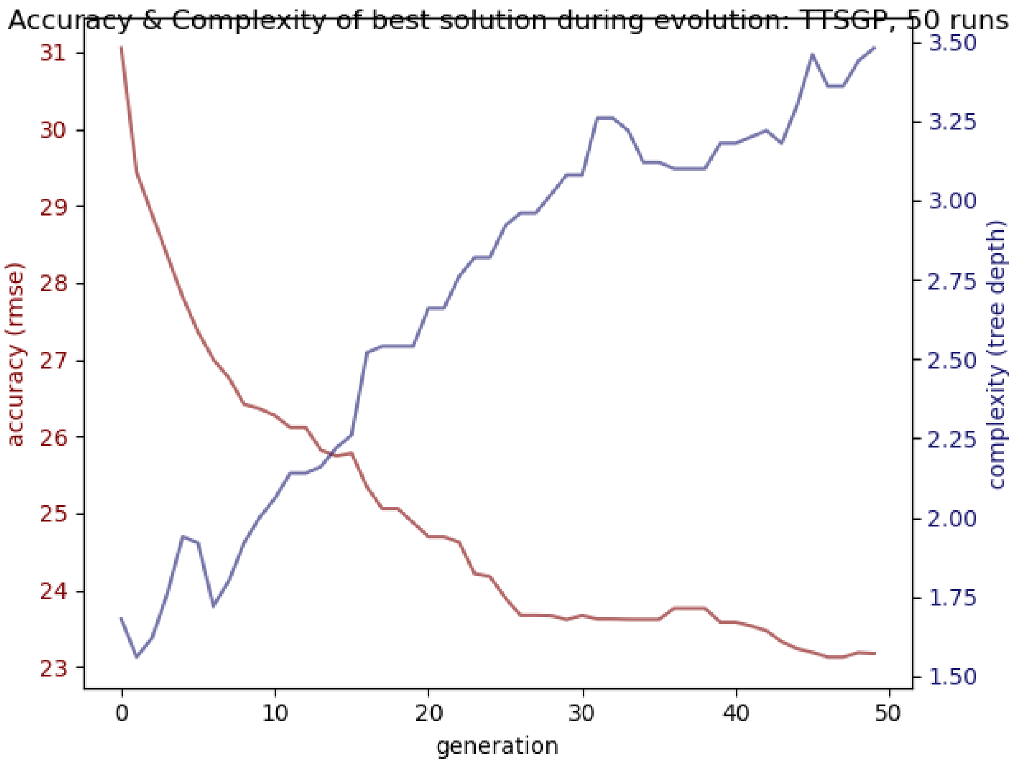


Figure 7: Evolution using Standard MOGP, houseprice-boston

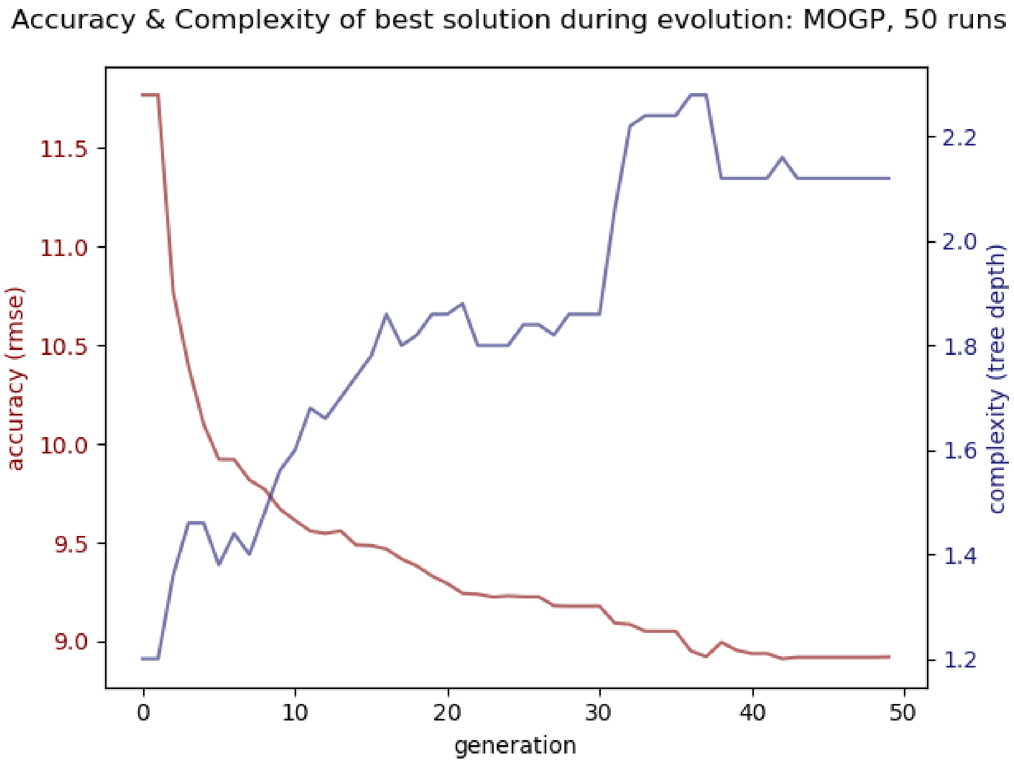


Figure 8: Evolution using Transient Terminal Set GP, houseprice-boston

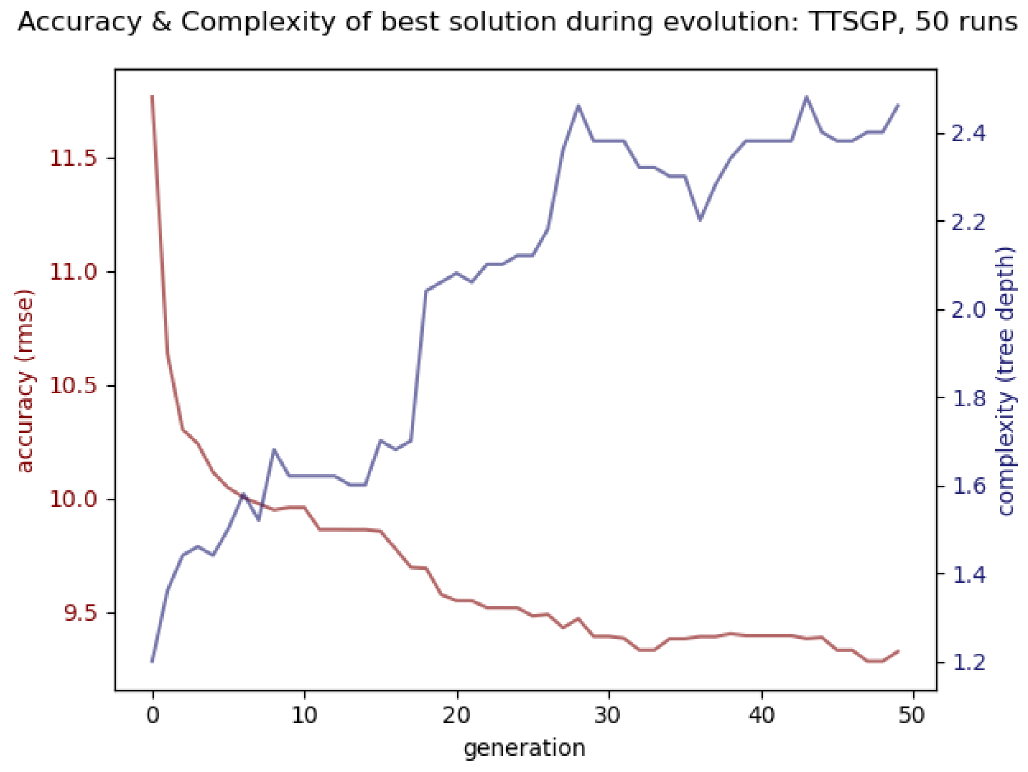


Figure 9: Evolution using Standard MOGP, estcount-bike

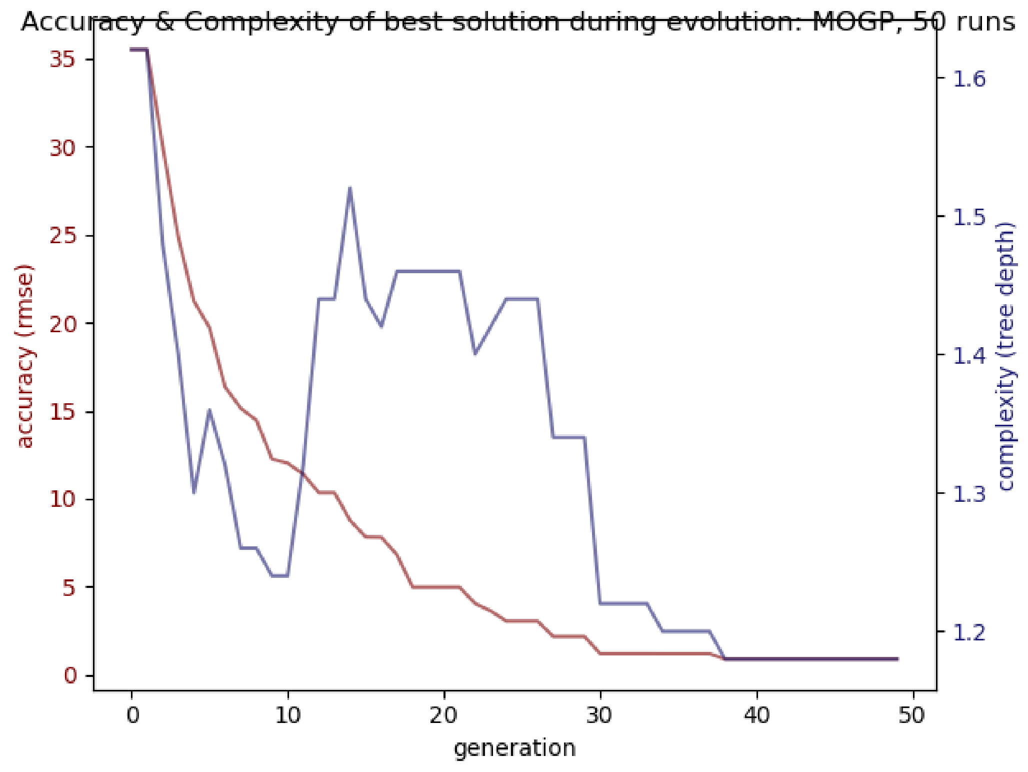


Figure 10: Evolution using Transient Terminal Set GP, estcount-bike

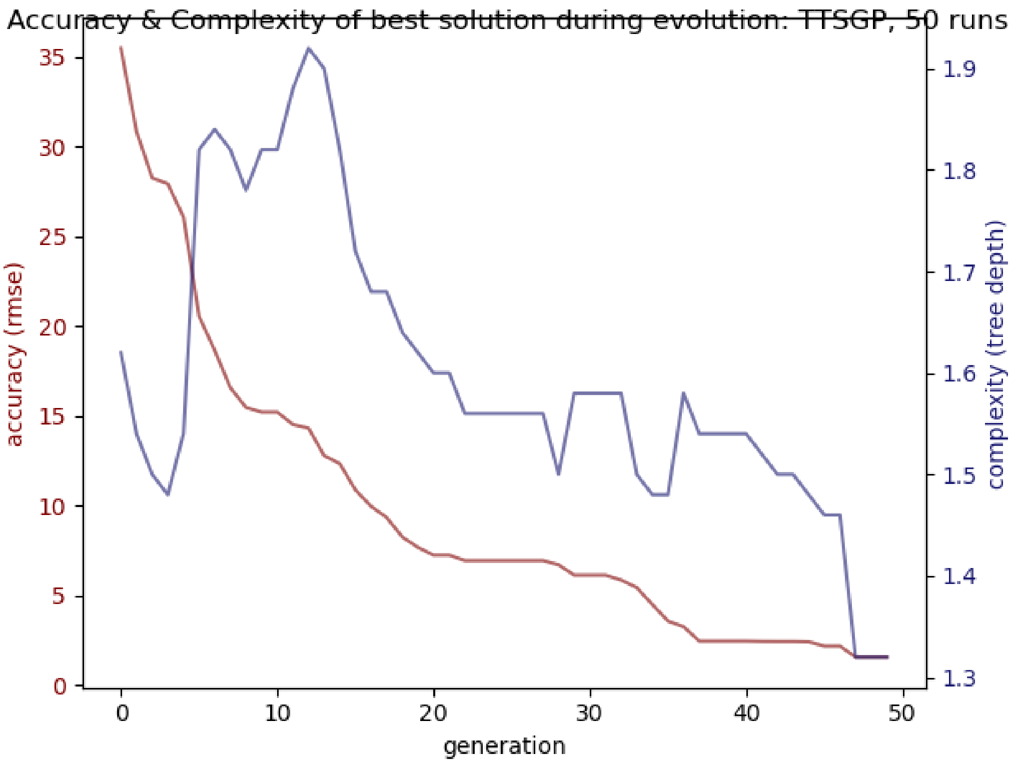


Figure 11: Best individual in Standard MOGP Evolution, winequality-red

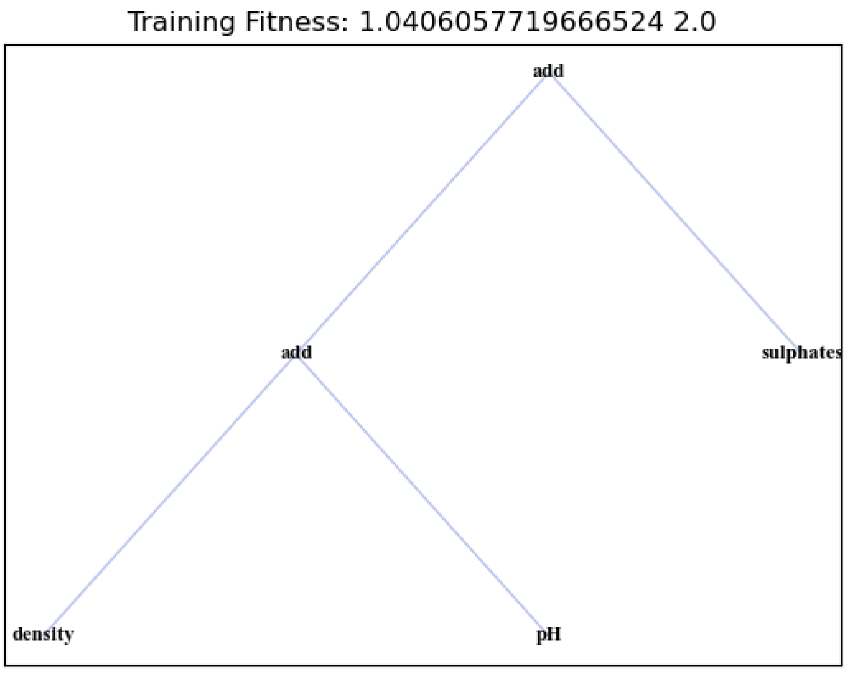
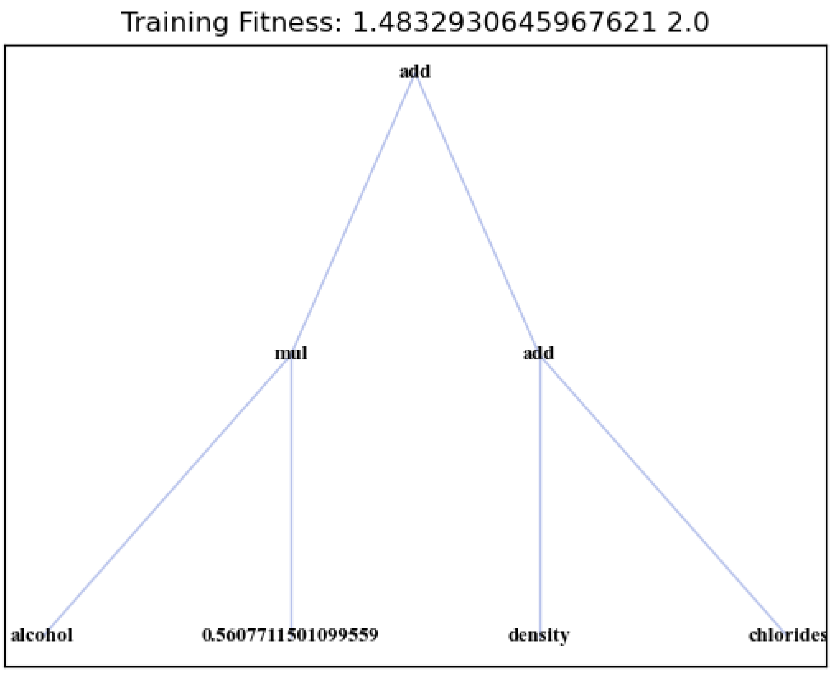


Figure 12: Best individual in TTSGP Evolution, winequality-red



The parameters for the experiment were *generations* = 50, *population* = 100, *crossover probability* = 0.5, *mutation probability* = 0.1, and *transient mutation probability* = 0.1. The dataset was split into 70% training instances and 30% testing instances prior to evolution. Results for the techniques were averaged across 50 random seeds, and the resulting evolution of the averaged best individual are plotted in **Figures 1 through 10** for each dataset. **Figure 3** and **Figure 4** are the best individuals for one of the random seeds (661477758) in the Red Wine Quality dataset, and serve to illustrate the symbolic regression trees generated by both techniques. A full list of the seeds used during experimentation can be obtained from the file *shared.py* in the aforementioned git repository.

The results in **Figures 1 through 10** suggest TTSGP is competitive with Standard MOGP but does not result in improvement in either the accuracy or complexity measures. Across all 5 datasets TTSGP achieved an RMSE approximately equal to the RMSE produced during Standard MOGP or slightly higher, however the tree depth of TTSGP solutions was consistently higher than Standard MOGP, generally within 0.25 nodes greater on average (except for in the case of the Concrete Strength data). This is in contrast to an earlier experiment performed on the Red Wine Quality dataset, which suggested a marginal improvement over Standard MOGP.

There are several hypothetical factors which may have caused the poor performance of TTSGP.

- First, the chosen *transient mutation probability* may be inadequate for symbolic regression. The current value of 10% was arbitrarily chosen to match the *mutation probability*, and further experimentation must be performed to determine whether any improvement can be made by adjusting *tmutpb*.
- Second, the rules for adding subtrees to the TTS may also be overly lenient, as the currently implemented mean approach is heavily susceptible to outliers. As with the mutation probability, further experimentation must be done to determine whether any improvement can be made through adjustment of this approach.
- Finally, the TTSGP approach may be incomplete in its current state. Further extensions can be made to the algorithm to improve the results. Most notably, these are selecting the node to mutate

within a tree via a heuristic function based on node depth (the current implementation utilizes an equal probability for each node) and selecting the member of the TTS to mutate in based on the tree topography at the selected node (the current implementation selects a random member of the TTS).

4 Intended Future Work

While the majority of the Transient Terminal Set algorithm is already implemented, there are several noteworthy aspects which need to be further investigated or developed to improve the results of TTSGP experimentation.

- As mentioned in **Section 3**, further experimentation will be done to analyze the effectiveness of raising or lowering the transient mutation probability on a generated solution's accuracy & complexity.
- Second, the criteria for a subtree to be added to the set will be adjusted, as the currently-implemented mean approach is very susceptible to outliers. Experimentation will be performed to investigate which alternative (median, percentile, etc) is best suited for this problem.
- Third, as mentioned in **Section 1**, a more appropriate measure of model complexity will be created to ensure candidate solutions in TTSGP are actually less complex than those generated in standard MOGP.
- Fourth, further research and experimentation into the *probability of node insertion* and *probability of terminal selection during transient mutation* will be undertaken, as these concepts are closely linked with the Transient Terminal Set.

As of **1/4/20**, major bugfixes and general improvements to the project code have been made and rendered the initial results, presented on 12/15/2020, irrelevant.