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

Asher Stout, 300432820

January 8, 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 4 January 2021, were **175**.

2 Transient Terminal Set

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

```
Input: population size $\rho$, crossover probability $p_c$, mutation probability $p_m$, transient mutation probability $p_d$, terminal set $T$, function set $F$, death age $\alpha$
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`, and the data utilized is freely accessible at `https://archive.ics.uci.edu/ml/datasets/wine+quality`. No 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, which was determined to be irrelevant.

Figure 1: Evolution using Standard MOGP, winequality-red

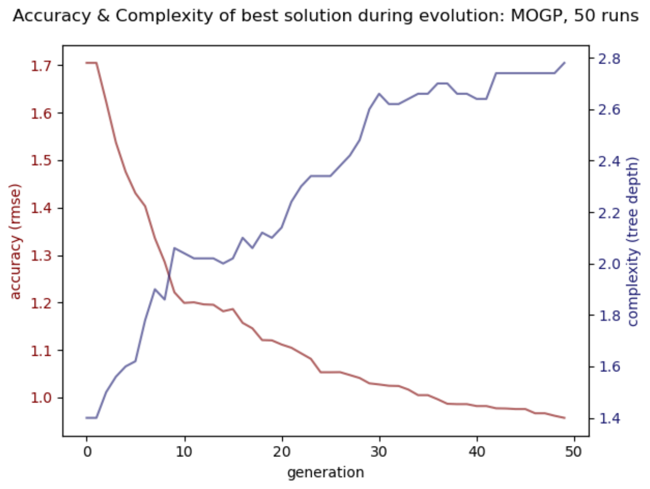


Figure 2: Evolution using Transient Terminal Set GP, winequality-red

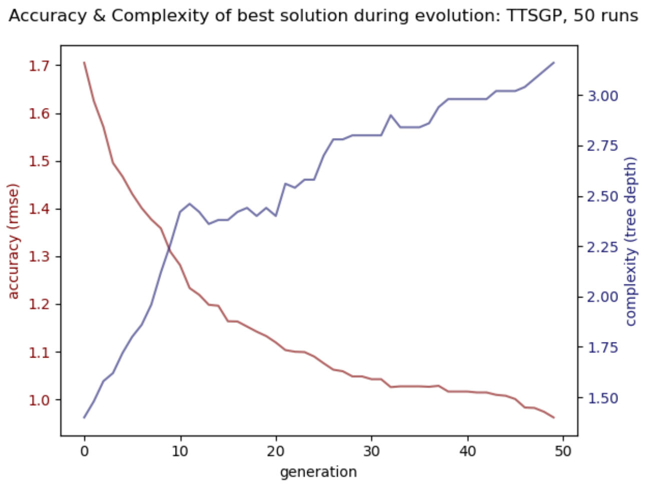


Figure 3: Best individual in Standard MOGP Evolution, winequality-red

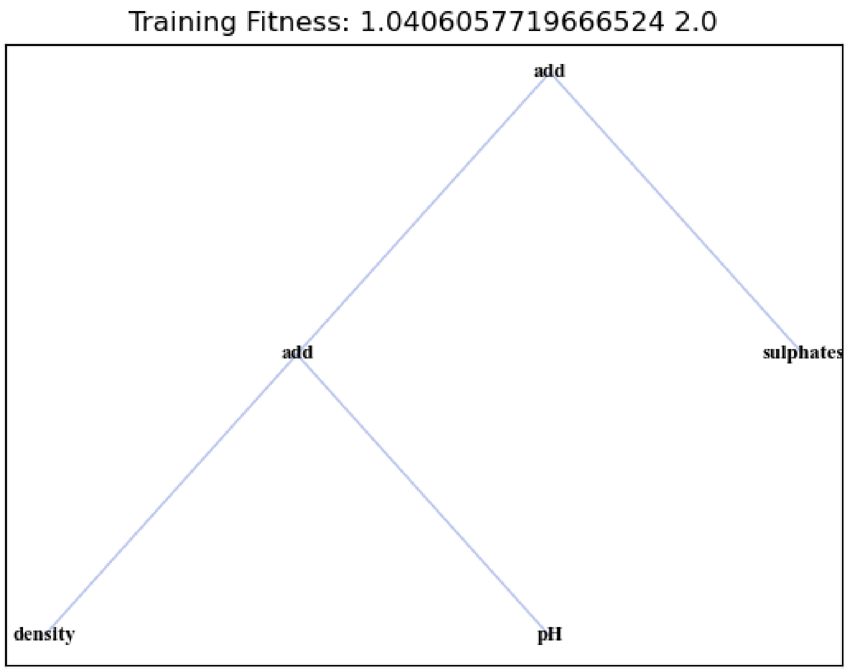
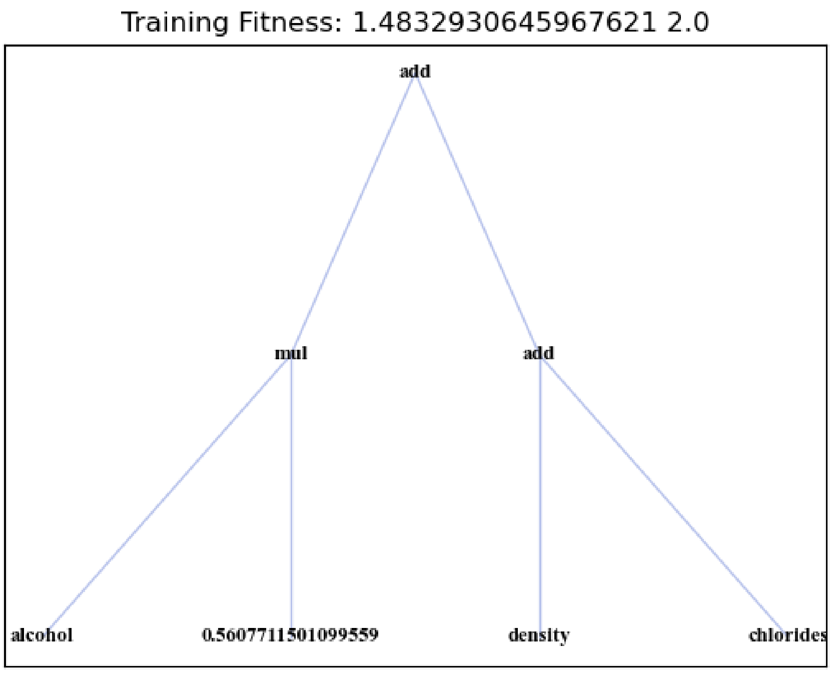


Figure 4: Best individual in TTSGP Evolution, winequality-red



For this analysis the dataset winequality-red was utilized to determine whether any improvement exists in using TTSGP over Standard MOGP. No preprocessing was performed on the data prior to testing. The dataset was split into 70% training instances and 30% testing instances prior to evolution.

The parameters for the experiment were *generations* = 50, *population* = 100, *crossover probability* = 0.5, *mutation probability* = 0.1, and *transient mutation probability* = 0.1. Results for the techniques were averaged across 50 random seeds, and the resulting evolution of the averaged best individual are plotted in **Figures 1 and 2**. **Figure 3** and **Figure 4** are the best individuals for one of the random seeds (661477758), 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 and 2** show no significant improvement in the interpretability of TTSGP solutions compared to those generated during Standard MOGP. It is worth noting that this is in opposition to the initial results (where the experiment was run over 30 different seeds), where it was shown that TTSGP potentially made a slight improvement in solution interpretability. This invites further inquiry into the effectiveness of TTSGP over Standard MOGP. Further experimentation must be performed comparing the techniques over other datasets to gauge the global effectiveness of TTSGP.

4 Intended Future Work

While the majority of the Transient Terminal Set algorithm has been already implemented in Python, 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 research must be done into the effectiveness of the TTSGP approach. This will take the form of investigating the improvement of TTSGP over Standard MOGP on other common regression datasets.

- 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.