Bayesian Model Selection for Reducing Bloat and Overfitting in Genetic Programming for Symbolic Regression

G.F. Bomarito
geoffrey.f.bomarito@nasa.gov
NASA Langley Research Center
Hampton, VA, USA

P.E. Leser
patrick.e.leser@nasa.gov
NASA Langley Research Center
Hampton, VA, USA

N.C.M. Strauss
University of Utah
Salt Lake City, UT, USA

K.M. Garbrecht
University of Utah
Salt Lake City, UT, USA

J.D. Hochhalter
jacob.hochhalter@utah.edu
University of Utah
Salt Lake City, UT, USA

ABSTRACT

When performing symbolic regression using genetic programming, overfitting and bloat can negatively impact generalizability and interpretability of the resulting equations as well as increase computation times. A Bayesian fitness metric is introduced and its impact on bloat and overfitting during population evolution is studied and compared to common alternatives in the literature. The proposed approach was found to be more robust to noise and data sparsity in numerical experiments, guiding evolution to a level of complexity appropriate to the dataset. Further evolution of the population resulted not in overfitting or bloat, but rather in slight simplifications in model form. The ability to identify an equation of complexity appropriate to the scale of noise in the training data was also demonstrated. In general, the Bayesian model selection algorithm was shown to be an effective means of regularization which resulted in less bloat and overfitting when any amount of noise was present in the training data.

ACM Reference Format:
G.F. Bomarito, P.E. Leser, N.C.M. Strauss, K.M. Garbrecht, and J.D. Hochhalter. 2022. Bayesian Model Selection for Reducing Bloat and Overfitting in Genetic Programming for Symbolic Regression. In *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO '22).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The efficacy of a Genetic Programming (GP) [1] solution is often characterized by its (1) fitness, i.e. ability to perform a training task, (2) complexity, and (3) generalizability, i.e. ability to perform its task in an unseen scenario. Bloat is a common phenomenon for GP in which continued training results in significant increases in complexity with minimal improvements in fitness. There are several theories for the prevalence of bloat in GP which postulate possible evolutionary benefits of bloat [2]; however, for most practical purposes bloat is a hindrance rather than a benefit. FOr example, bloated solutions are less interpretable and more computationally expensive. Overfitting is another common phenomena in GP and the broader machine learning field. Overfitting occurs when continued training results in better fitness but reduced generalizability.

GECCO '22, July 9-13, 2022, Boston, MA
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conventional wisdom ties bloat to overfitting, thus many methods have been developed which aim to alleviate both issues [e.g., 2–4]. Alternatively, recent works give evidence that the two phenomena are independent, and that a solution to one does not guarantee a solution of the other [5, 6]. Nevertheless, the current work introduces a Bayesian approach that mitigates both bloat and overfitting in GP.

Symbolic regression is a common application of GP, here referred to as GPSR. In symbolic regression an analytic function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is sought such that $f(\mathbf{x}) = y$ best matches some training data $\mathcal{D}\{(\mathbf{x}_i, y_i)\}_{i=0}^{N}$ with d-dimensional input features $\mathbf{x}$ and label y. Often in GPSR, the training data is considered to be noise-free (or rather noise is not considered at all) and fitness is defined as the ability of the function to closely match every datapoint. Alternatively, explicit consideration of noise in the training data (e.g., $y = f(x) + \epsilon$) during fitness calculation gives two benefits. Firstly, it can reduce the tendency to overfit because the GPSR problem moves away from the goal of simply fitting every datapoint as closely as possible. Secondly, a Bayes factor can be used which can penalize parametric complexity (Bayesian Occam's Razor [7]). In the current work, it will be shown that Bayesian model selection using a variant of the Bayes Factor (Fraction Bayes Factor [8]) can be used in GPSR to achieve two goals: reduced bloat and reduced overfitting.

2 METHODS

The basis of this work is a typical GPSR implementation found in the open-source Python package `Bingo` [9]. Acyclic-graph encodings of equations (see for example Figure 1) are used because they provide increased performance and decreased bloat relative to tree-based encodings [10]. The number of nodes in an acyclic-graph encoding of an equation is a measure of the complexity of the equation. The genetic variation consists of single-point crossover and mutation that is one of the following: point (node) mutation; edge (directed connection) mutation; node and edge mutation; prune mutation; and branch mutation. The results of the current work are thought to be independent of the above GPSR setup, while the treatment of numerical constants and the definition of fitness are more relevant.

Recent works have illustrated that local optimization of numerical constants can greatly enhance the ability of GPSR to build accurate models [11–14]. In this type of approach, numerical constants

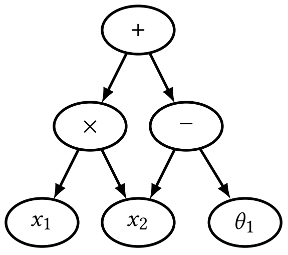

Figure 1: Acyclic graph representation of the equation $f(\mathbf{x}, \boldsymbol{\theta}) = x_1 x_2 + x_2 - \theta_1$ with complexity of 6.

are simply placeholders during genetic operations; i.e., equations are developed with the form $f(\mathbf{x}, \boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^P$. Here, P represents the parametric dimension/complexity of an equation. During fitness evaluation the values of the parameters are chosen such that fitness is locally optimal. In most cases, an error metric such as root mean squared error (RMSE) on the training data is used as a fitness score; e.g., $f(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\theta}^*)$ where $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\ \text{RMSE}(f, \mathcal{D}, \boldsymbol{\theta})$.

Local optimization of this form is deterministic. When the training data are noisy, the conditional probability of the parameters can be estimated using Bayes' Theroem,

$$p(\boldsymbol{\theta}|\mathcal{D}, f) = \frac{p(\mathcal{D}|\boldsymbol{\theta}, f)p(\boldsymbol{\theta}|f)}{p(\mathcal{D}|f)}, \tag{1}$$

where $p(\mathcal{D}|\boldsymbol{\theta}, f)$ is the likelihood of observing the data given a particular model and parameter realization, and $p(\boldsymbol{\theta}|f)$ represents available prior knowledge. $p(\mathcal{D}|f)$ is the marginal likelihood or *model evidence* and can be used for selecting between two models f_1 and f_2 via Bayes factor,

$$B = \frac{p(f_1|\mathcal{D})}{p(f_2|\mathcal{D})} = \frac{p(\mathcal{D}|f_1)p(f_1)}{p(\mathcal{D}|f_2)p(f_2)}. \tag{2}$$

In the GPSR context where models are generated through strategic, but heavily randomized, processes, it is practical to assign equal prior model probabilities such that $p(f_1) = p(f_2)$, resulting in a Bayes factor that is simply the ratio of marginal likelihoods. Following similar logic, it is reasonable to objectively set $p(\boldsymbol{\theta}|f_i) \propto 1$ (i.e., an improper uniform prior) to reflect the lack of knowledge about the parameter values apart from their support on $\mathbb{R}^P$.

The use of improper uniform priors complicates the calculation of Bayes factor due to the appearance of indeterminant constants in the marginal likelihoods. The Fractional Bayes Factor (FBF), B_γ, was developed to address this issue [8]:

$$B_\gamma = \frac{q_1(\gamma)}{q_2(\gamma)} \tag{3}$$

where

$$q_i(\gamma) = \frac{\int_{\mathbb{R}^P} p(\mathcal{D}|\boldsymbol{\theta}, f_i)p(\boldsymbol{\theta}|f_i)d\boldsymbol{\theta}}{\int_{\mathbb{R}^P} p(\mathcal{D}|\boldsymbol{\theta}, f_i)^\gamma p(\boldsymbol{\theta}|f_i)d\boldsymbol{\theta}}. \tag{4}$$

Here, q_i is referred to as the normalized marginal likelihood (NML).[1] The unknown constants in the marginal likelihood are normalized out using the $\gamma \in (0, 1]$ power of the likelihood function, allowing for consistent model selection when using improper uniform priors.

[1]The notational dependence of q_i on γ will be dropped henceforth for clarity.

Following the recommendation of [8], $\gamma = 1\sqrt{N}$ was chosen for improved robustness to prior misspecification.

Sequential Monte Carlo (SMC) [15] as implemented in the open-source Python package SMCPy[16] is used in the present study to estimate q_i. A number of sample-based methods exist for approximating solutions to the inverse problem of Equation (1), including Markov chain Monte Carlo (MCMC) and SMC. However, in contrast with MCMC, SMC enables direct, unbiased estimation of $p(\mathcal{D}|f_i)$. Additionally, the SMC algorithm involves likelihood annealing of the same form as the denominator of Equation (4), which produces the NML as a byproduct.

The NML can be used as a fitness measure in GPSR when paired with an appropriate model selection algorithm. The deterministic crowding (DC) algorithm [17] was designed to promote diversity and avoid premature convergence in evolutionary computation. In DC the population is segregated into pairs. Each pair produces two offspring through a combination of crossover and/or mutation. Each member of the original population then competes against the most similar of its two offspring; the most fit is selected for the next generation. Probabilistic crowding (PC) [18] is an extension of DC wherein selection is a stochastic process depending on the fitness values of the individuals. The probability that f_i is chosen relative to f_j is

$$p(f_i|F_i, F_j) = \frac{F_i}{F_i + F_j} \tag{5}$$

where F_i and F_j are the fitness measures of f_i and f_j, respectively. PC provides a restorative pressure that can further support diversity in the population and reduce the overall tendency for the population to concentrate on a single most-fit species.

In the present study, PC is combined with NML as a fitness measure, resulting in a form of Bayesian model selection using the FBF:

$$p(f_i|q_i, q_j) = \frac{q_i}{q_i + q_j} = \frac{B}{B + 1} \tag{6}$$

For a FBF = 1, this selection probability is equal to 0.5. For cases where FBF > 1, the i^{th} model is more likely to proceed to the next generation, etc. In other words, the equation that is more supported by the data is more probable to be selected, and the extent to which the support is greater influences that probability.

3 RESULTS & DISCUSSION

The goal of the current section is to illustrate the extent to which the use of the fractional Bayes factor during selection in GPSR mitigates bloat and overfitting. To this end, a numerical experiment is performed which quantifies bloat and overfitting in GPSR. In this experiment, datasets are generated using:

$$y_i = 2\sin(x_{0,i}) + 3 + \epsilon_i \tag{7}$$

where $x_{0,i}$ is independently sampled from a uniform distribution, $x_{0,i} \overset{iid}{\sim} U(0, \frac{3\pi}{2})$, and ϵ_i represents additive noise that is Gaussian with standard deviation σ; i.e., $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma)$. For each $\sigma \in \{0, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0\}$ 50 training datasets of $N = 20$ points are generated (each with independent samples of x_0 and ϵ). The operators that are included in the GPSR runs consist only of $[+, -, \times]$, thus GPSR is prevented from recovering the data-generating function. The choice of operators in combination with a sparse dataset are intended to target the GPSR algorithms' natural tendencies toward overfitting.

The use of FBF in selection is a combination of two components: the use of probabilistic crowding and the use of NML as a fitness metric. To illustrate the degree to which each of these components impact results, four GPSR algorithms are tested:

(1) A base GPSR algorithm: DC and RMSE.
(2) A GPSR algorithm using DC and NML.
(3) A GPSR algorithm using PC and RMSE[2].
(4) The FBF-based GPSR algorithm: PC and NML.

All 4 GPSR algorithms use common hyperparameters besides the ones listed above. On each of the training datasets they are run with a population size of 120 for 1000 generations using a mutation rate of 0.4 and crossover rate of 0.4.

The training dynamics for the datasets with $\sigma = 0.5$ are illustrated in Figure 2. In this figure, and all subsequent figures, lines and shaded areas represent mean values and 95% confidence intervals from the 50 repeats, respectively. All of the algorithms see improved fitness with more training, though speed of training slows significantly after approximately 100 generations. Comparing PC+RMSE to DC+RMSE it can be seen that training is slowed slightly for PC due to the lower probability of selecting individuals with only slightly better fitness. A similar, but less pronounced trend is seen for the NML algorithms. Note that trends seen here are representative of other σ values.

On the same sets of data ($\sigma = 0.5$), bloat characteristics are illustrated using complexity and number of parameters during evolution (Figure 3). The characteristic sign of bloat can be seen using the rate of change of these metrics. The use of NML rather than RMSE has a clear effect on bloat in Figure 3. NML identifies an optimal number of parameters, and, after reaching this value, the number of parameters does not deviate significantly with continued evolution. This is in sharp contrast to the RMSE algorithms where the number of parameters continuously increase (i.e. substantial parameter bloat). The complexity has a similar trend wherein RMSE-based evolution sees unbounded complexity growth. Interestingly, NML-based evolution exhibits an initial peak in complexity followed by a steady albeit slight simplification of equations in the population. The results of these experiments also indicate improved bloat control when using PC rather than DC, which likely corresponds to the increased diversity provided by PC.

Figure 4 illustrates complexity and parametric dimension after 1000 generations as a function of σ. While the base algorithm (DC+RMSE) is relatively constant with noise level, the PC and NML algorithms illustrate a dependence on the dataset. For the FBF-based GPSR algorithm (PC+NML) in particular, simpler models are preferred when there appears to be more noise in the data, and the complexity and number of parameters increase when the amount of noise added appears small. Automatic adaptation to the dataset is an attractive property of this bloat control method.

The presence of overfitting is investigated using a group of test datasets. For each level of noise σ, a single test dataset is generated using the data-generating function of Equation (7) with $N = 1000$ data points. Performance on these unseen datasets is a measure of generalizability. Test performance is quantified by RMSE, meaning the optimal test fitness is equal to σ, although it is important to note that available GPSR operators were not sufficient to learn Equation (7) precisely and thus σ should be viewed as a lower bound.

Figure 5 illustrates the typical progression of test fitness during evolution. The base GP algorithm (DC+RMSE) has an initial period of improving generalization followed by a steady increase in test error, ultimately resulting in a large degree of overfitting. While not as significant, PC+RMSE also tends to overfit. On the other hand, the NML-based methods see no trend toward overfitting. After the

[2]Since using RMSE results a minimization problem rather than a maximization problem, a selection probability of $p(f_i|F_i, F_j) = \frac{F_j}{F_i+F_j}$ is used rather than Equation (5)

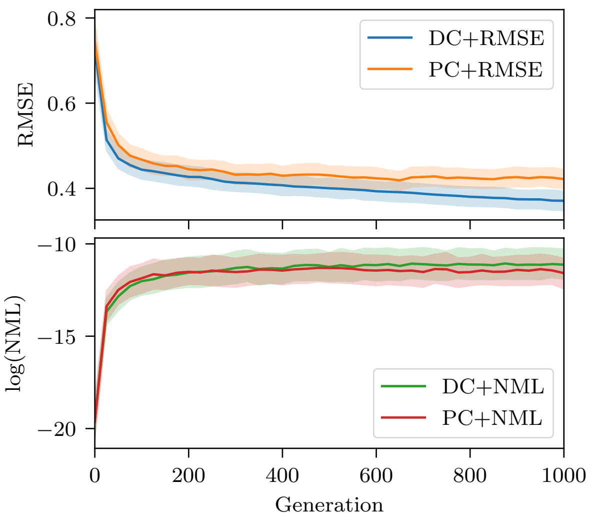

Figure 2: Training fitness as a function of evolution for data with σ = 0.5.

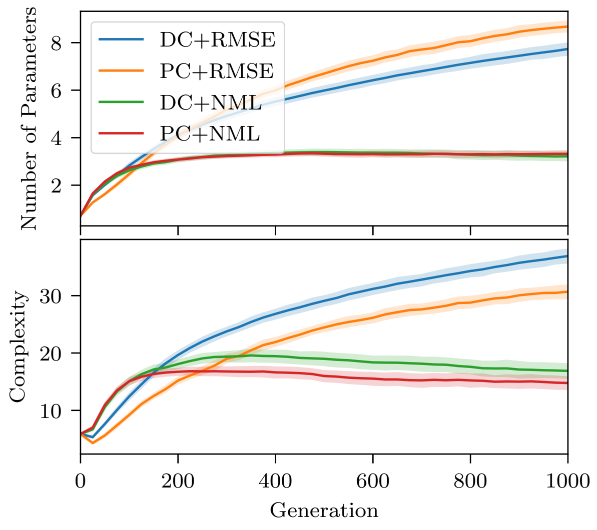

Figure 3: Complexity and parametric dimension as a function of evolution for data with σ = 0.5.

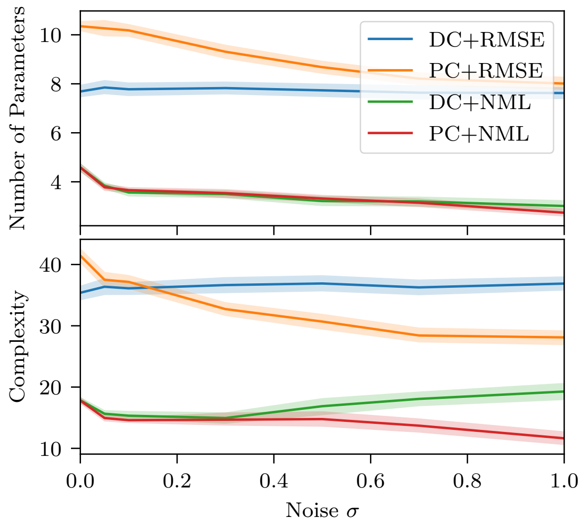

Figure 4: Complexity and parametric dimension after 1000 generations as a function of noise level in dataset.

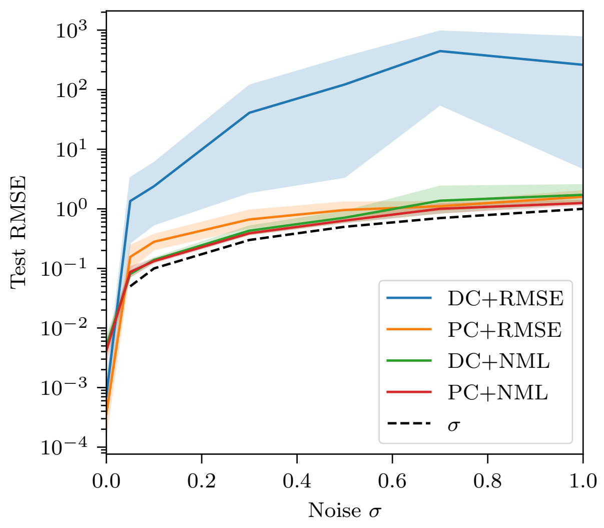

Figure 6: Test fitness after 1000 generations as a function of as a function of noise level.

initial improvements in test fitness, NML-based methods remain relatively constant with further evolution, albeit somewhat noisy.

Figure 6 illustrates the performance of each algorithm as a function of noise level. The base GP algorithm has a clear tendency to overfit. The other algorithms have less tendency to overfit, and the FBF-based GP algorithm (PC+NML) was found to produce the best generalizability across the noise levels. The exception here is that in the absence of noise in the datasets, the RMSE-based algorithms outperform NML-based algorithms; however, if the dataset contains even a small amount of noise then these results suggest that NML-based methods are preferred.

4 CONCLUSIONS & FUTURE WORK

Bayesian model selection was shown to have a substantial impact on the dynamics of GPSR evolution in terms of bloat and overfitting. Specifically, the proposed FBF-based selection algorithm (PC+NML)

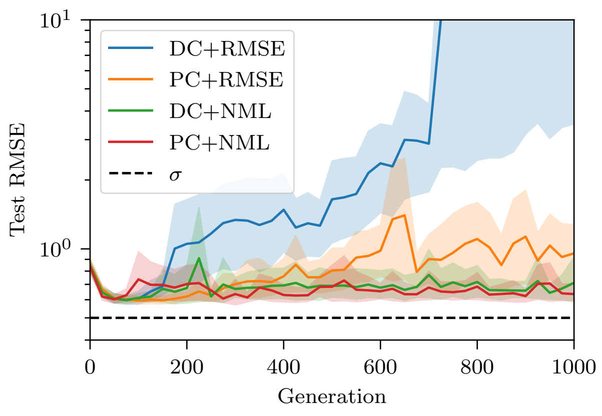

Figure 5: Test fitness as a function of evolution for data with $\sigma = 0.5$.

was shown to be an effective means of regularization which resulted in less bloat and overfitting compared to state-of-the-art RMSE-based algorithms. Figures 3 and 5 illustrate that when the FBF-based algorithm encounters a scenario which is ripe for overfitting, it is able to identify an optimal parametric dimension and a generalizable equation which adapts naturally to the level of noise in the dataset. Moreover, extended evolution of the population resulted not in overfitting or bloat, but rather in slight simplifications in model form which can enhance interpretability. This characteristic of NML-based GPSR evolution dynamics is of great practical significance considering scientists can only evaluate regression results through currently available training and testing data. Specifically, when using the base GPSR implementation, scientist must rely on established best practices to identify the optimum generation to end the analysis and avoid overfitting and bloat. With the proposed NML-based algorithm, this tendency was not observed and therefore active measures are not required to prevent overfitting and bloat. Identification of the optimum generation simply requires monitoring for stability of population dynamics which can be easily incorporated into the algorithm, if so desired.

A noteworthy side effect of using the proposed Bayesian fitness metric is an estimate of the joint distribution of numerical constants and the noise level in the data for each equation in the population. These distributions can be leveraged to produce predictions with quantified uncertainty using Monte Carlo simulation. Further investigation of this ability is forthcoming.

REFERENCES

[1] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.
[2] Vipul K Dabhi and Sanjay Chaudhary. A survey on techniques of improving generalization ability of genetic programming solutions. *arXiv preprint arXiv:1211.1119*, 2012.
[3] Tejashvi R Naik and Vipul K Dabhi. Improving generalization ability of genetic programming: comparative study. *Journal of Bioinformatics and Intelligent Control*, 2(4):243–252, 2013.

[4] Jeannie Fitzgerald, R Muhammad Atif Azad, and Conor Ryan. Bootstrapping to reduce bloat and improve generalisation in genetic programming. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 141–142, 2013.

[5] Ekaterina J Vladislavleva, Guido F Smits, and Dick Den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation*, 13 (2):333–349, 2008.

[6] Sara Silva and Leonardo Vanneschi. Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1115–1122, 2009.

[7] Iain Murray and Zoubin Ghahramani. A note on the evidence and bayesian occam's razor. 2005.

[8] Anthony O'Hagan. Fractional bayes factors for model comparison. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):99–118, 1995.

[9] G.F. Bomarito. Bingo. https://github.com/nasa/bingo, 2022.

[10] Michael Schmidt and Hod Lipson. Comparison of tree and graph encodings as function of problem complexity. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1674–1679, 2007.

[11] Michael Kommenda, Gabriel Kronberger, Stephan Winkler, Michael Affenzeller, and Stefan Wagner. Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1121–1128, 2013.

[12] Z Emigdio, Leonardo Trujillo, Oliver Schütze, Pierrick Legrand, et al. Evaluating the effects of local search in genetic programming. In *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, pages 213–228. Springer, 2014.

[13] Vinicius Veloso De Melo, Benjamin Fowler, and Wolfgang Banzhaf. Evaluating methods for constant optimization of symbolic regression benchmark problems. In *2015 Brazilian conference on intelligent systems (BRACIS)*, pages 25–30. IEEE, 2015.

[14] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351*, 2021.

[15] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (3):411–436, 2006.

[16] P.E. Leser. Smcpy - sequential monte carlo with python. https://github.com/nasa/smcpy, 2022.

[17] Samir W Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[18] Ole J Mengshoel and David E Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. 1999.