Genetic Programming and Evolvable Machines (2020) 21:273–280
https://doi.org/10.1007/s10710-019-09366-0

SOFTWARE REVIEW

Software review: the GPTIPS platform

Amir H. Gandomi[1] · **Ehsan Atefi**[2]

Published online: 29 October 2019

Abstract
GPTIPS is a widely used genetic programming software that was developed in Matlab. The most recent version of this software, GPTIPS 2.0, provides a symbolic multi-gene regression for data analysis, in addition to traditional evolutionary algorithms. We briefly explain the GPTIPS methodology and describe its main features, including its weaknesses and strengths, and give examples of GPTIPS applications.

Keywords GP · MGGP · SMGR

1 Introduction

GPTIPS is a freely available open-source multi-gene genetic programming (MGGP) platform written in Matlab for data mining and model discovery (https://sites.google.com/site/gptips4matlab/). GPTIPS generates explicit predictor models in the form of symbolic equations by searching a numeric dataset for pertinent relationships between the input and output. It uses symbolic multi-gene regression (SMGR) to reduce complexity and improve the functionality, accuracy, and robustness of bio-inspired traditional genetic programming (GP) [1]. A traditional GP approach starts with a population of randomly generated trees termed the initial solutions. Next, to find an accurate model that explains the data, these trees are evolved. Unlike traditional GP, GPTIPS starts with a population of multi-tree solutions (MGGP), constructed from a vector of randomly generated trees. Next, the goodness of fit of each solution is evaluated, and a certain percentage of the population is selected to be parents by using a probabilistic Pareto tournament. To form the next generation, the selected solutions are evolved by mutation and crossover. This process is repeated until a user-defined truncation criterion is satisfied [2]. GPTIPS supports several run termination criteria, including the best

✉ Amir H. Gandomi
gandomi@uts.edu.au

[1] Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia

[2] Faculty of Mechanical Engineering Department, Manhattan College, Bronx, NY, USA

model fitness, maximum number of generations, and maximum runtime. GPTIPS 2.0, the most recent version of GPTIPS, uses a variety of evolutionary schemes including several types of crossover to expedite the evolution of solutions while retaining the performance of the models. Figure 1 contains an example of a user defined GPTIPS configuration file (a configuration file is required to start GPTIPS). In Fig. 1, the GPTIPS parameters are explained by comments on each line.

In Sect. 2 we introduce GPTIPS and explain its main features, including symbolic multi-gene regression (SMGR) and rate-based crossover. The major strengths (Sect. 3) and weaknesses (Sect. 4) of GPTIPS are discussed in-depth. We include an example of a GPTIPS input (configuration) and output file with instructions, plus a few examples of GPTIPS, solving real-world problems from different fields in science and industry.

```
function gp = Sample_Config(gp)
%loading the data
load Sample_Data; % 'Sample Data' includes training, testing and
validation datasets
gp.userdata.xtrain = X; % Input dataset used for training
gp.userdata.ytrain = Y; % Output dataset used for training
gp.userdata.xval = X_val; % Input used for Validation see line 34
gp.userdata.yval = Y_val; % Output used for Validation see line
34
gp.userdata.xtest = X_test; % Input used for testing
gp.userdata.ytest = Y_test; % Output used for testing
%Configurational parameters used in Genes development
gp.genes.max_genes = 4; %Maximum number of trees per solution
%Probability of mutation, crossover etc., used for breeding
gp.operators.mutation.p_mutate = 0.6;
gp.operators.crossover.p_cross = 0.38;
gp.operators.directrepro.p_direct = 0.02;
%The list of functions (funtionset) used in building the trees
gp.nodes.functions.name =
{'times','minus','plus','rdivide','square','tanh','exp','log','mu
lt3','add3','sqrt','cube','negexp','neg','abs','sin','cos'};
%Configurational parameters used for termination of the run
gp.runcontrol.pop_size = 100;    %Population size
gp.runcontrol.timeout = 500;     %Maximum runtime
gp.runcontrol.runs = 30;         %Number of runs
gp.runcontrol.num_gen = 50;      %Number of generations
gp.selection.tournament.p_pareto = 0.4; %Selection procedure
%The definition of fitness for evaluation of the solution
gp.fitness.terminate = true;
gp.fitness.terminate_value = 0.3;
gp.fitness.minimisation = true;
gp.fitness.fitfun = @regressmulti_fitfun;
%Enabling the holding out validation process
gp.userdata.user_fcn = @regressmulti_fitfun_validate;
```

Fig. 1 A sample GPTIPS configuration file written in Matlab. Comments explaining the different variables and sections are shown in green font (starting with % sign)

2 Major features of GPTIPS

One of the key advantages of GPTIPS is using SMGR, as illustrated in Fig. 2. GPTIPS generates a population of solutions, each of which contains randomly generated trees (genes). The trees are combined using different weights. Figure 2a shows a single solution formed of four trees. Their outputs, plus a bias value, are added linearly using weights listed in Fig. 2b to form the solution (see Fig. 2c). In a method like least squares, the weight corresponding to each tree is calculated by minimizing the error of fitting the model to the training dataset.

Another important feature of GPTIPS is that it generates a thorough report on the model properties and configurational parameters. Figure 3 contains an example GPTIPS report, which lists the properties of the best training model, including the input variables, model complexity, and the number of trees used in the model (Fig. 3a). The components of the best training model and the corresponding weights are shown in Fig. 3b, c. The model is explicitly expressed in a symbolic format (Fig. 3d), and the performance of the model is statistically and visually studied using both training and test data sets (Fig. 3d, f). The GPTIPS report contains additional features, including the Pareto Front curve (Fig. 4) and a comparison of the performance of predictive models.

3 GPTIPS strengths

GPTIPS is a freely available GP platform. The majority of GPTIPS features operate automatically, however the user must have a basic understanding of Matlab programming. They must also set configurational variables, such as the maximum number of trees in each solution, the population size and the maximum number of generations.

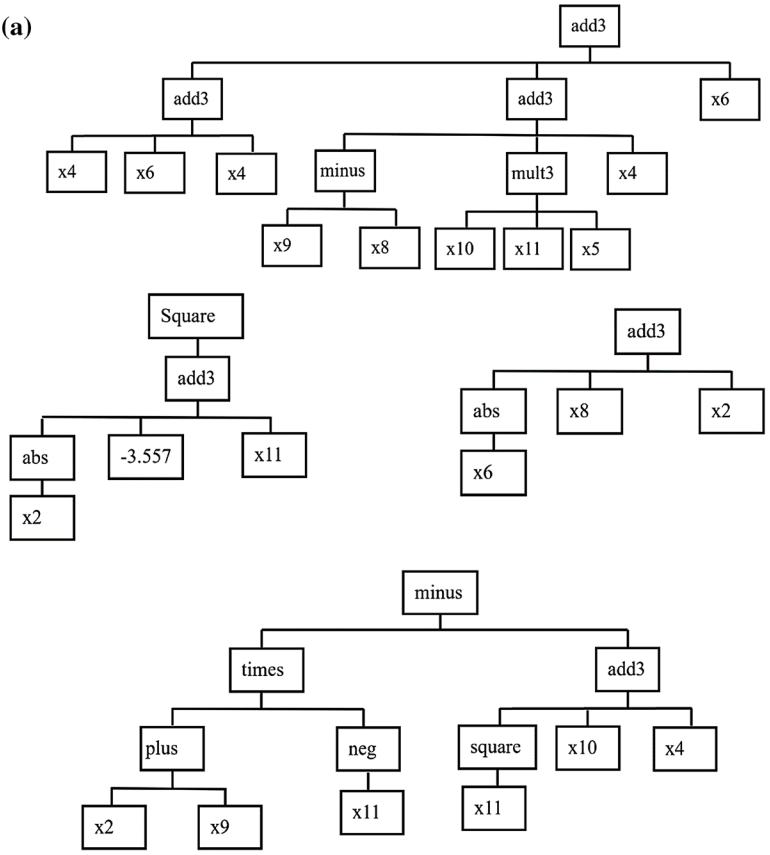
GPTIPS provides three tournament selection options: goodness of fit (RMS error), solution complexity (size), and Pareto Front. Complexity can be measured either by the number of nodes per tree or expressional complexity [1, 3]. The user can choose the probability of each tournament selection method.

GPTIPS produces an HTML report that contains comprehensive data on configuration and run parameters as well as the results of the analysis. The GPTIPS report contains a statistical analysis of the solutions on training, validation and test datasets. The GPTIPS report also provides details of models that lie on the Pareto front (using the training dataset), along with a plot of the Pareto front.

The Pareto curve plots expressional complexity against the goodness of fit (R^2) for the models that are not dominated by other solutions in terms of both complexity and accuracy. The Pareto curve enables the user to visualize the performance of solutions and select a solution that retains a balance between complexity and accuracy. The final solutions can be saved for future use in Matlab as a symbolic function and exported to C [1, 3].

In order to save run-time, in addition to the maximum number of generations, the user can define the maximum run-time and desired fitness score at which

(a)



(b)

Term	Weight	Value
Bias	-	-0.659
Gene 1	0.0556	$0.167\ x_4 + 0.111\ x_6 - 0.0556\ x_8 + 0.0556\ x_9 + 0.0556\ x_5\ x_{10}\ x_{11}$
Gene 2	-0.0433	$-0.0433\ (x_{11} + abs(x_2) - 3.56)^2$
Gene 3	-0.0973	$0.0973\ x_4 + 0.0973\ x_{10} + 0.0973\ x_{11}\ (x_2 + x_9) + 0.0973\ x_{11}^2$
Gene 4	-0.248	$-0.248\ x_2 - 0.248\ x_8 - 0.248\ abs(x_6)$

(c)

$y = 0.264\ x_4 - 0.248\ x_2 + 0.111\ x_6 - 0.303\ x_8 + 0.0556\ x_9 + 0.0973\ x_{10} + 0.308\ x_{11} + 0.308\ abs(x_2) - 0.248\ abs(x_6) - 0.0433\ abs(x_2)^2 + 0.0973\ x_2\ x_{11} + 0.0973\ x_9\ x_{11} - 0.0867\ x_{11}\ abs(x_2) + 0.0539\ x_{11}^2 + 0.0556\ x_5\ x10\ x_{11} - 1.21$

Fig. 2 **a** The parts (4 trees/genes) of a solution in a symbolic tree format. **b** The corresponding weight of each tree. **c** The simplified form of the solution

(a)

Description	Value
Genes	4
Nodes	24
Expressional complexity	67
Depth	4
Inputs	4
Inputs used	x_1 x_2 x_3 x_6

(b)

Term	Value
Bias	-3.21
Gene 1	-0.00166 exp (6.0 x_6)
Gene 2	0.00913 exp(exp(x_6)) exp (-1.0 x_3) exp(x_1)
Gene 3	0.113 exp (3.0 x_6 - 1.0 exp(x_2))
Gene 4	0.595 exp (2.0 1.72^x_6)

(c)

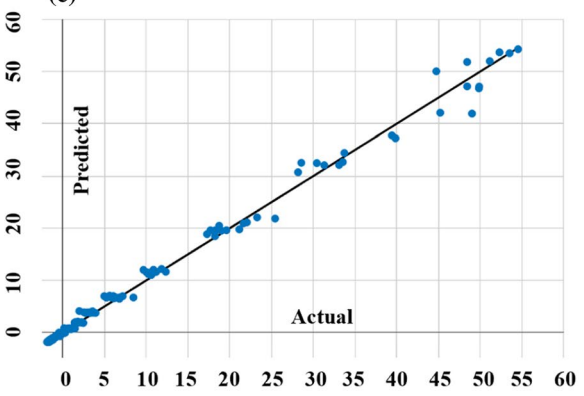
Overall model after simplification

$y = 0.595 \exp(2.0\ 1.72^{x_6}) + 0.113 \exp(3.0\ x_6 - 1.0 \exp(x_2)) - 0.00166 \exp(6.0\ x_6) + 0.00913 \exp(\exp(x_6)) \exp(-1.0\ x_3) \exp(x_1) - 3.21$

(d)

Metric	Value
R2	0.99797
RMSE	0.74638
MAE	0.49455
SSE	68.5206
Max. abs. error	2.1692
MSE	0.55708

(e)



(f)

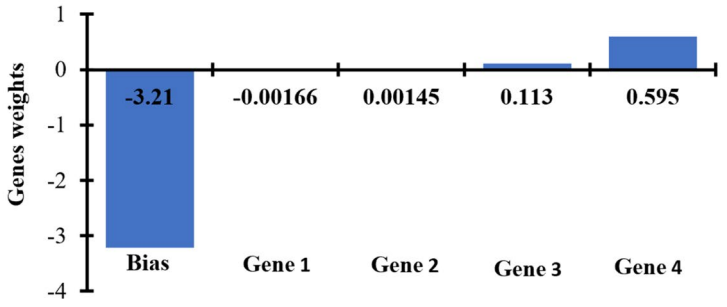


Fig. 3 Selected parts of a sample GPTIPS report. These include: **a** the general properties of the model. **b** The component trees/genes. **c** The symbolic expression of the model. **d** Performance on the training dataset. **e** Plot of performance on the test dataset. **f** Graph of tree/gene weights

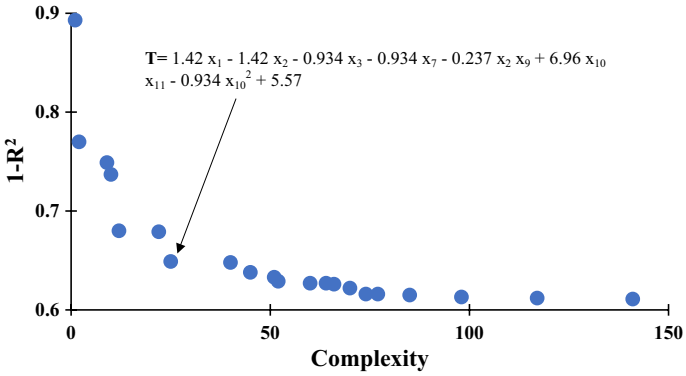


Fig. 4 Example of Pareto tournament generated by GPTIPS on a dataset with 11 dimensions. Pareto front curve shows the models that are dominant in one of the two categories of complexity and goodness of fit ($1\text{-}R^2$). To analyze this dataset GPTIPS generated 3000 models, of which 21 models lie on the Pareto frontier (solid circles)

GPTIPS will stop. By default GPTIPS uses root mean squared error (RMSE) for fitness but GPTIPS also allows the user to define their own fitness function. Also, GPTIPS is compatible with the Matlab Parallel Computing Toolbox, which reduces runtime by using multiple computer cores [1, 3].

To improve robustness, GPTIPS uses a novel crossover technique, includes an extensive set of functions to be the internal nodes of the trees, used to create the evolved models, and forces each tree to be unique in the initial generation.

Evolutionary algorithms have the tendency to evolve bigger programs. This phenomenon is called bloat, which may cause overfitting of the training dataset. Bloat significantly increases the complexity of solutions while providing little to the accuracy of the model. GPTIPS support features that suppress bloat. For instance, the user can limit the depth of the trees and include expression complexity in the fitness of solutions. Since GPTIPS uses multi-gene genetic programming (MGGP), it tends to generate horizontal bloat, meaning extra trees are added with little improvement in model performance. To resolve this issue, GPTIPS permits the user to manually delete low-performing genes by setting the maximum number of genes per solution and by monitoring gene performance, during the runtime, in the evolving populations (i.e. interactive evolution). As shown in Fig. 3a, GPTIPS tracks the input variables used in each predictive model. For instance, in Fig. 3a, only four of the six input variables are used in the best training model. This information can be used to further analyze the importance of each independent variable by measuring the frequency of variable usage.

GPTIPS has been widely used for solving engineering and science problems. In the field of structural engineering (e.g. bridges and pipelines), it is known that studying the geotechnical behavior of structural systems is complex due to its dependency on several variables, such as soil properties. GPTIPS has been used widely in this field to model material and structural problems [4, 5] as well as geotechnical and earthquake engineering problems [6, 7]. Other examples of

GPTIPS applications include solving multi-objective management problems [8], energy forecasting [9], and biotechnology and bioprocess optimization [10, 11].

4 GPTIPS weaknesses

GPTIPS is a Matlab-based platform. Although GPTIPS is freely available software, Matlab needs to be purchased by the user. GPTIPS needs an advanced Matlab library that includes Symbolic Math and Statistics Toolbox. Matlab is not always as fast as other programming languages, such as Python. It is possible that trees in a solution are collinear, i.e. they are not independent, and therefore cannot add to each other. GPTIPS requires the user to define the function set, the maximum number of trees and the population size. Additionally, GPTIPS does not permit seeding the initial population. Finally, it is not simple to access individual parts of the solutions in each population to extract or manipulate them.

Acknowledgement We would like thank William B. Langdon for his careful review and valuable comments on the draft of this paper.

References

1. D.P. Searson, GPTIPS 2: an open-source software platform for symbolic data mining, in *Handbook of Genetic Programming Applications* (Springer, Berlin, 2015), pp. 551–573. https://doi.org/10.1007/978-3-319-20883-1_22
2. D. Searson, M. Willis, G. Montague, Co-evolution of non-linear PLS model components. J. Chemom. (2007). https://doi.org/10.1002/cem.1084
3. A.H. Gandomi, A.H. Alavi, C. Ryan, *Foreword. Handbook of Genetic Programming Applications* (Springer, Berlin, 2015). https://doi.org/10.1007/978-3-319-20883-1
4. A.H. Gandomi, A.H. Alavi, A new multi-gene genetic programming approach to nonlinear system modeling. Part I: materials and structural engineering problems. Neural Comput. Appl. (2012). https://doi.org/10.1007/s00521-011-0734-z
5. H. Bolandi, W. Banzhaf, N. Lajnef, K. Barri, A.H. Alavi, Bond strength prediction of FRP-bar reinforced concrete: a multi-gene genetic programming approach, in *Proceedings of Genetic and Evolutionary Computation Conference Companion,* pp. 364–364 (2019). https://doi.org/10.1145/3319619.3322066
6. A. Garg, A. Garg, K. Tai, S. Sreedeep, Estimation of factor of safety of rooted slope using an evolutionary approach. Ecol. Eng. **64**, 314–324 (2014). https://doi.org/10.1016/j.ecoleng.2013.12.047
7. S. Gharehbaghi, A.H. Gandomi, S. Achakpour, M.N. Omidvar, A hybrid computational approach for seismic energy demand prediction. Expert Syst. Appl. **110**, 335–351 (2018). https://doi.org/10.1016/j.eswa.2018.06.009
8. A.H. Gandomi, A.H. Alavi, A new multi-gene genetic programming approach to non-linear system modeling. Part II: geotechnical and earthquake engineering problems. Neural Comput. Appl. **21**(1), 189–201 (2012). https://doi.org/10.1007/s00521-011-0735-y
9. O. Chikumbo, E. Goodman, K. Deb, Triple bottomline many-objective-based decision making for a land use management problem. J. Multi Criteria Decis. Anal. (2015). https://doi.org/10.1002/mcda.1536

10. A. Tahmassebi, A.H. Gandomi, Building energy consumption forecast using multi-objective genetic programming. Measurement **118**, 164–171 (2018). https://doi.org/10.1016/j.measurement.2018.01.032
11. A. Cankorur-Cetinkaya et al., CamOptimus: a tool for exploiting complex adaptive evolution to optimize experiments and processes in biotechnology. Microbiology **163**(6), 829–839 (2017). https://doi.org/10.1099/mic.0.000477