Evolving Toxicity Models using Multigene Symbolic Regression and Multiple Objectives

Charles Hii, Dominic P. Searson and Mark J. Willis

***Abstract*— In this contribution a multi-objective genetic programming algorithm (MOGP) is used to perform symbolic regression. The genetic programming (GP) algorithm used is specifically designed to evolve mathematical models of predictor response data that are "multigene" in nature, i.e. linear combinations of low order non-linear transformations of the input variables. The MOGP algorithm simultaneously optimizes the dual (and competing) objectives of maximization of 'goodness-of-fit' to data and minimization of model complexity in order to develop parsimonious data based symbolic models. The functionality of the multigene MOGP algorithm is demonstrated by using it to generate an accurate, compact QSAR (quantitative structure activity relationship) model of existing toxicity data in order to predict the toxicity of chemical compounds.**

***Index Terms*— genetic programming, multi-objective optimization, symbolic regression, QSAR, toxicity, T. pyriformis.**

I. Introduction

Genetic programming [1] is a biologically inspired machine learning method that evolves computer programs to perform a specified task. It does this by randomly generating an initial population of computer programs (represented by tree structures) and then mutating and crossing over the best performing trees to create a new population. This process is iterated until the population contains programs that solve the task well. An excellent, free to download introduction and review of the GP literature is provided by [2].

When GP is used to create empirical mathematical models of data acquired from a process or system, it is commonly referred to as symbolic regression. In contrast to classical regression analysis, in which the user must specify the structure of the model, GP automatically evolves both the structure and the parameters of the mathematical model. Symbolic regression has had successful academic [3] and industrial applications [4].

The purpose of this paper is to describe the use of a multi-objective genetic programming (MOGP) algorithm to perform symbolic regression. The GP algorithm used employs a variant of symbolic regression called multigene symbolic regression [5], [6], [7], [8] that evolves linear combinations of non-linear transformations of the input variables. The multigene approach can often yield more accurate and compact models than those obtained using "standard" GP for symbolic regression. However, the multigene approach has not hitherto explicitly utilized model complexity as a goal of model development. Hence, to optimize the multiple objectives of 'goodness of fit' to data and model complexity simultaneously the well known NSGA-II multiple objective method [9] is incorporated into the existing multigene symbolic regression algorithm.

This paper is structured as follows. Section II provides a brief overview of Genetic Programming. Section III introduces the multigene symbolic regression method that our existing GP software (GPTIPS) implements. Next, in section IV, the NSGA-II algorithm is briefly described. In section V the GPTIPS software is briefly discussed. In sections VII-X, the capabilities of the multigene MOGP algorithm are demonstrated by using it to evolve an accurate, compact mathematical model to predict the toxicity of chemical compounds using a data set from the literature containing over 1000 compounds along with measured toxicity values. Finally, in section XI we provide some concluding remarks.

II. Genetic programming

The evolutionary computational (EC) method of GP evolves populations of symbolic tree expressions to perform a user specified task. A brief description of GP is provided here.

In GP, each tree expression can be thought of as being analogous to the DNA of an individual in natural evolution. The evolution of the expressions occurs over a number of generations (iterations) and each new generation of individuals is created from the existing population by direct copying as well as performing operations on the individuals analogous to the alterations to DNA sequences that naturally occur during sexual reproduction and mutation. This is accomplished by evaluating each individual in the current population to determine its 'fitness' (i.e. its performance on the user specified objective function or functions) and performing probabilistic selection and recombination of individuals biased towards those that are relatively fit compared to the other individuals in the population.

At the beginning of each run, a population of symbolic expressions is randomly generated. This is accomplished using a simple tree building algorithm that randomly selects nodes, with replacement, from a pool comprising primitive functions (e.g. addition, subtraction, the hyperbolic tangent, natural logarithm, exponential, etc.), the input variables as well as randomly generated constants. These nodes are randomly assembled into tree structured symbolic expressions, subject to user-defined tree size and/or depth constraints. After evolving the population for a number of generations by copying, mutation and recombination

Manuscript received March, 22nd, 2011.

Charles Hii is with the School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK.

Dominic P. Searson is with the School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK. (e-mail: d.p.searson@ncl.ac.uk).

Mark J. Willis is with the School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK. (phone +44 191 222 7242; e-mail mark.willis@ncl.ac.uk).

operations, the tree expression with the best fitness is usually selected as the best solution to the problem.

Two principal genetic recombination operators are used in GP: sub-tree crossover and sub-tree mutation. Sub-tree crossover is an operation performed on two parent trees that generates two offspring. For each expression, a sub-tree is randomly selected. These sub-trees are then exchanged to create two new expressions to go into the next generation. Sub-tree mutation operates on a single parent expression and generates a single offspring expression. First, a randomly selected sub-tree of the parent is deleted. Then, a new sub-tree is randomly generated using the same tree building algorithm that was used to build the initial population of expressions. The resulting offspring expression is then inserted into the new population. The mutation operation is used relatively infrequently compared to the crossover operation and its purpose is to maintain genetic diversity over the course of the run and to prevent premature convergence to unsatisfactory solutions.

In GP, the choice of the primitive functions is domain dependent and in symbolic regression, where there is little or no prior knowledge of the underlying relationships, mathematical operators such as those described above are typically employed with a high degree of success [6], [7]. In practice, it is often best to perform some initial runs with a few simple primitives (e.g. addition, multiplication and subtraction) and then incrementally add other non-linear primitives—such as the hyperbolic tangent function—to evaluate whether more accurate and compact symbolic expressions may be evolved.

III. MULTIGENE SYMBOLIC REGRESSION

Symbolic regression is usually performed by using GP to evolve a population of trees, each of which encodes a mathematical equation that predicts a ($N \times 1$) vector of outputs $\mathbf{y}$ using a corresponding ($N \times M$) matrix of inputs $\mathbf{X}$ where N is the number of observations of the response variable and M is the number of input (predictor) variables, i.e. the ith column of $\mathbf{X}$ comprises the N input values for the ith input variable and is referred to as the input variable x_i.

In contrast, in multigene symbolic regression each symbolic model (i.e. each member of the GP population) is a weighted linear combination of the outputs from a number of GP trees, where each tree may be considered to be a “gene” in the overall genome. The mathematical form of the multigene representation is shown below (for N =1).

$$y = \sum_{i=1}^{n} d_i G_i + d_0 \tag{1}$$

where y is the predicted output, G_i is the value of the ith gene and is, in general, a function of one or more of the input variables, d_i is the ith weighting coefficient, n is the number of genes and d_0 is a bias/offset term.

For example, the multigene model shown in Fig. 1 predicts an output variable y using input variables x_1, x_2 and x_3.This model structure contains non-linear terms (e.g. the hyperbolic tangent) but is linear in the parameters with respect to the coefficients d_0, d_1 and d_2. In practice, the user specifies the maximum number of genes $G_{\max}$ a model is allowed to have and the maximum tree depth $D_{\max}$ any gene may have and therefore can exert control over the maximum complexity of the evolved models. In particular, we have found that enforcing stringent tree depth restrictions (i.e. maximum depths of 4-6 nodes) allows the evolution of relatively compact models that are linear combinations of low order non-linear transformations of the input variables.

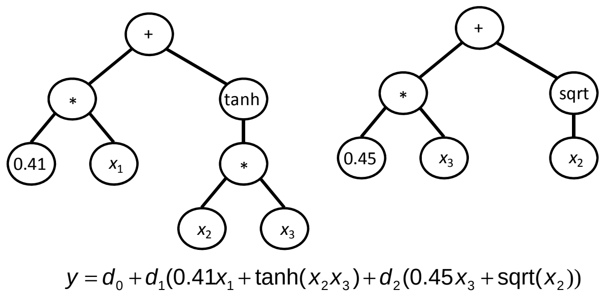

+
*
tanh
0.41
x_1
*
x_2
x_3
+
*
sqrt
0.45
x_3
x_2
$y = d_0 + d_1(0.41x_1 + \tanh(x_2x_3)) + d_2(0.45x_3 + \mathrm{sqrt}(x_2))$

Fig. 1. Example of a multigene symbolic model.

For each model, the linear coefficients are estimated from the training data using ordinary least squares techniques as follows:

The output of the ith gene in any given multigene individual will be an ($N \times 1$) vector $\mathbf{g}_i$. These output vectors can be grouped together (along with a column containing "ones" to represent the bias/offset term) into a ($N \times (n+1)$) matrix $\mathbf{G}$.

The ((n+1) × 1) coefficient vector $\mathbf{d}$ (containing the coefficients d_0, ..., d_n) may then be computed using the least squares normal equation to minimise (in the least squares sense) the prediction error of the output training data $\mathbf{y}$:

$$\mathbf{d} = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{y} \tag{2}$$

Computationally, however, this method of computing $\mathbf{d}$ is unsuitable because duplicate genes in an individual can cause rank deficiency in $\mathbf{G}$. To remedy this, in practice, the Moore-Penrose pseudo-inverse $(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{\#}$ is computed using the singular value decomposition (SVD) and used in place of $(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}$in Equation 2.

Hence, it can be seen that multigene GP combines the power of classical linear regression with the ability to capture non-linear behaviour without needing to pre-specify the structure of the non-linear model. The work in [6] showed multigene symbolic regression can be more accurate and computationally efficient than the standard GP approach for symbolic regression, and [7] showed that the multigene approach can be successfully embedded within a non-linear partial least squares algorithm.

In GPTIPS, there are two types of crossover operators: *high level crossover* and the "standard" GP subtree crossover, which is referred to here as *low level crossover*. In low level crossover, a gene is selected randomly from each parent individual and then standard subtree crossover is performed to create two offspring new genes. The parent genes are discarded and the new offspring genes then replace the parent genes in the otherwise unaltered individual in the new population.

The initial population is constructed by creating individuals that contain randomly generated GP trees with between 1 and G_{max} genes. During a GP run, genes are acquired and deleted using two point *high level crossover*. This allows the exchange of genes between individuals and it is used in addition to the “standard” GP recombination operators. A two point high level crossover is performed as in the following example:

The first parent individual contains the genes (G_1 G_2 G_3) and the second contains the genes (G_4 G_5 G_6 G_7) where G_{max} = 5. Two randomly selected crossover points are created for each individual. The genes enclosed by the crossover points are denoted by < … >.

$(G_1 < \mathbf{G_2} > G_3) \quad (G_4 < \mathbf{G_5}\ \mathbf{G_6}\ \mathbf{G_7} >)$

The genes enclosed by the crossover points are then exchanged resulting in the two new individuals below.

$(G_1\ G_5\ G_6\ G_7\ G_3) \quad (G_4\ G_2)$

Two point high level crossover allows the acquisition of new genes for both individuals but also allows genes to be removed. If an exchange of genes results in any individual containing more genes than G_{max} then genes are randomly selected and deleted until the individual contains G_{max} genes.

IV. Multi-Objective GP

Multi-objective genetic programming (MOGP) [10] is an extension of GP. In standard GP algorithms, typically only one objective is optimized; for symbolic regression this is usually a measure of ‘goodness-of-fit’ to the training data, typically expressed as sum of squared errors (SSE), mean squared error (MSE) or root mean squared error (RMSE). One problem with using the single objective of ‘goodness-of-fit’ is that models will be evolved that fit the training data well at the expense of complexity. Such models are usually not robust, e.g. they may not predict accurately given new data.

Unfortunately, GP is prone to evolving overly complex models due to the tendency (called "bloat") to acquire model terms that provide little or no effect on the final prediction. This effect can be mitigated, to some extent, in multigene regression by restricting tree depths and observing the statistical significances of the genes in the evolved models but even then, the evolution of overly complex models can still be problematic.

However, a MOGP algorithm can be used to simultaneously optimize more than one objective. For symbolic regression, this is useful because a second objective can be specified: i.e. maximize ‘goodness-of-fit’ and minimize model complexity. There have been many multi-objective optimization techniques reported in the literature. NSGA-II [9] and SPEA2 [11] are the two of the more popular algorithms. In this work, the NSGA-II algorithm is incorporated into GPTIPS. The structure of this algorithm is shown in Fig. 2.

The NSGA-II algorithm is used at the end of each generation of the multigene GP algorithm. First, it classifies the individuals from both the new and old population according to their position on the Pareto front. Fig. 3 shows how the Pareto front is categorised using the dual objectives of fitness and complexity (in terms of symbolic regression it desirable to have high fitness and low complexity). Pareto front level 1 consists of a set of Pareto optimal solutions. The solutions that are on the level 1 front are not dominated by any other solution, e.g. in symbolic regression any model that lies on the level 1 front is not outperformed by any other model in the population in terms of both prediction error and complexity. The solutions that comprise Pareto front level 2 are not dominated by any other solutions; apart from those in Pareto front level 1 and so on.

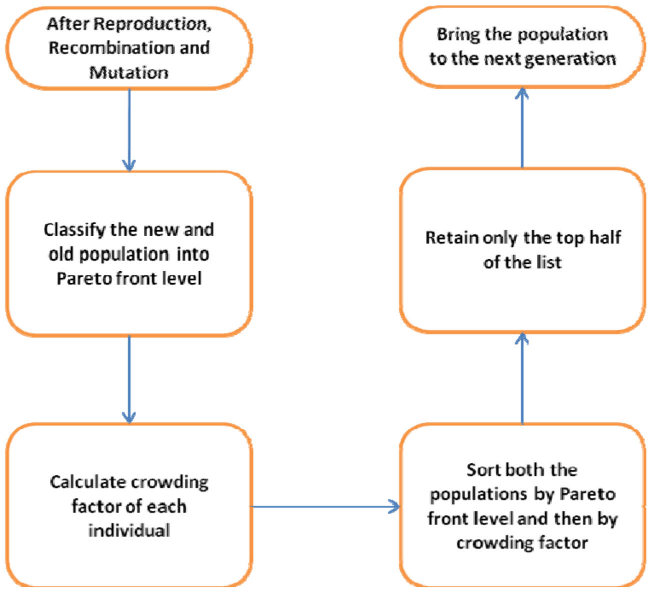

After Reproduction, Recombination and Mutation
Classify the new and old population into Pareto front level
Calculate crowding factor of each individual
Sort both the populations by Pareto front level and then by crowding factor
Retain only the top half of the list
Bring the population to the next generation

Fig. 2. Flowchart of the NSGA-II algorithm.

Next, NSGA-II requires the calculation of a “crowding factor” for each individual. This is the average distance of a solution from the nearest solutions (either side) on the same Pareto front. This is also shown in Fig. 3 for a solution on the level 1 front.

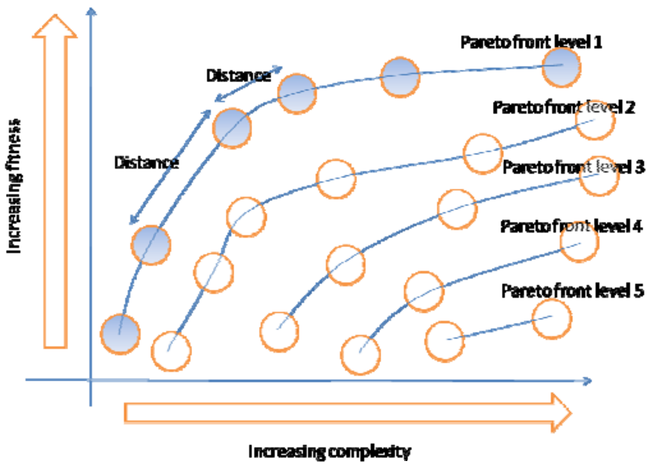


Fig. 3. Pareto optimal fronts (model fitness vs. complexity).

The crowding factor is used to increase the diversity of the population giving less priority to the solutions that are crowded together during the ranking process in NSGA-II. This gives the GP algorithm the ability to attempt to explore other possible solutions within the search space and limit the convergence of the algorithm to local maxima or minima. Once the crowding factor has been calculated for all the solutions, the solutions are then ranked according to their position (those on level 1 are ranked above those on level 2 and so on) and solutions that are on the same Pareto front are ranked according to their crowding factor. The top 50% of the population survive to the next generation while the rest are discarded.

V. GPTIPS

The GP algorithm used in this work is based on a modified version of GPTIPS 1.0 [5], a free, open source MATLAB toolbox for performing genetic programming and symbolic regression. GPTIPS and documentation is available for download[1]. Included with GPTIPS are several multigene symbolic regression demos, of varying complexity, with pre-configured parameter files to allow users to experiment with different parameter settings and some artificial and "real" data sets.

Although users can create their own GP applications and fitness functions in GPTIPS, it was written with the primary intent of make it easy to perform multigene symbolic regression and so a number of MATLAB functions for facilitating this process are included within GPTIPS.

GPTIPS also provides several methods of mutating trees which are described in Section VI.

The user can set a wide variety of GPTIPS parameters, for instance, the relative probabilities of each of the recombinative processes (mutation and crossover). These processes are grouped into categories called events, i.e. the user can specify the probability of crossover events, direct reproduction events and mutation events. These must sum to one. The user can then specify the probabilities of the event subtypes, e.g. the probability of a two point high level crossover taking place once a crossover event has been selected, or the probability of a subtree mutation once a mutation event has been selected. However, GPTIPS provides default values for each of these probabilities so the user does not need to explicitly set them.

VI. GPTIPS FEATURES

GPTIPS is a predominantly command line driven open source toolbox that requires only a basic working knowledge of MATLAB. A run is configured by a simple configuration M file and there are a number of command line functions to facilitate post-run analyses of the results. Whilst not an exhaustive list, GPTIPS currently contains the following configurable GP features: tournament selection & plain lexicographic tournament selection, elitism, three different tree building methods (full, grow and ramped half and half) and six different mutation operators:

(1) subtree mutation
(2) mutation of constants using an additive Gaussian perturbation
(3) substitution of a randomly selected input node with another randomly selected input node
(4) set a randomly selected constant to zero
(5) substitute a randomly selected constant with another randomly generated constant
(6) set a randomly selected constant to one.

[1] http://sites.google.com/site/gptips4matlab/.

Also, GPTIPS can, without modification in the majority of cases, use nearly any built in MATLAB function as part of the function set for a run. The user can also write bespoke function node M files and fitness functions; hence GPTIPS can be used to solve problems other than non-linear modeling/symbolic regression.

In addition, GPTIPS has a number of features that are specifically aimed at the creation, analysis and simplification of multigene symbolic regression models. These include:

(1) use of a 'holdout' validation set during training to mitigate the effects of overfitting
(2) graphical display of the results of symbolic regression for any multigene model in the final population
(3) mathematical simplification of any model
(4) conversion to LaTeX format of any model
(5) conversion to PNG (portable network graphics) file of the simplified equation of any model
(6) conversion of any model to standalone M file for use outside GPTIPS
(7) graphical display of the statistical significance of each gene in a model
(8) functions to reduce the complexity of any model using "gene knockouts" to explore the trade off of model accuracy against complexity
(9) graphical population browser to explore the trade off surface of complexity/accuracy
(10) graphical input frequency analysis of individual models or of a user specified fraction of the population to facilitate the identification of input variables that are relevant to the output.

The Symbolic Math toolbox (a commercial toolbox available from the vendors of MATLAB) is required for the majority of the post run simplification and model conversion features and the Statistics Toolbox (also a commercial toolbox available from the vendors of MATLAB) is required for the display of gene statistical significance. The core functionality of GPTIPS and the ability to evolve multigene models does not, however, require any specific toolboxes.

VII. QSAR

QSAR (Quantitative Structure Activity Relationships) is a well established technique for deriving structure property relationships for chemical compounds that can be used to predict the properties of novel chemical structures. Chemical compounds can be represented by a large number of computed numerical values, called "descriptors", each of which in some way characterises the structure of the compound.

The idea of QSAR is to build empirical or semi-empirical models that relate the descriptors of a compound to some physical, chemical or biological property. Software packages are available to compute descriptor values for compounds with a known structure. Many of these are commercial products (e.g. DRAGON) but there are also free/open source packages (e.g. the Chemical Development Kit (CDK; [12]).

QSAR uses a data set of known chemical compounds and a measured endpoint for each compound. The measured endpoint is the property of interest. Typical properties of interest are those related to pharmaceutical drug development. These include biological activities representing the ability of a compound to perform its desired function (e.g. IC50, the concentration of a compound required to inhibit a particular biological or biochemical function by half) and the ADME properties (adsorption, distribution, metabolism and excretion) which characterise the behaviour of a of a pharmaceutical drug compound within the organism.

The prediction of chemical toxicity is another chemical property that is of vital importance in both pharmaceutical drug development and managing the environmental risk of chemical compounds. In the latter case there are legal regulatory structures (e.g. the REACH regulations in the European Union - EC 1907/2006) that specify that QSAR models should play a part in managing this risk in order to reduce the costs of experimental toxicity measurement.

One method for experimentally evaluating chemical toxicity is the measurement of the growth inhibition of ciliated protozoan *T. pyriformis*. There are freely available aquatic toxicity data for more than 1000 compounds, due to the efforts of Schultz and colleagues [13]. The authors of [14] have used this to compile a data set of 1093 unique compounds and have developed a number of predictive QSAR models using various descriptor packages and modelling methodologies.

Here, the use of the MOGP to evolve a predictive model of chemical toxicity using this data set is demonstrated (using the descriptors from the commercial DRAGON package) and the results compared with those published in [14] and [8].

VIII. Data

The *T. pyriformis* toxicity values (i.e. the response **y** data) are measured as the logarithm of the 50% growth inhibition concentration log(IGC50^{-1}). The data available for training QSAR models contains 644 compounds and 449 compounds are used as an external test/validation data set to verify the predictive ability of the models.

For each compound 1664 DRAGON descriptor values are used as the predictor data (i.e. the input **X** data contains 1664 input variables) - compound structures, toxicity and descriptor values are, at time of writing, available from the EU CADASTER website at http://www.cadaster.eu/node/65. 128 compounds (approximately 20%) in the training data set were randomly selected for use as a holdout validation data set leaving the training data containing 516 compounds.

In GPTIPS, holdout validation is performed as follows: at the end of each generation, the "best" individuals – those on the Pareto front - (as evaluated on the training data) are then evaluated on the holdout validation set.

IX. GPTIPS Run Settings

GPTIPS (version 1.0) was modified as outlined in Section III for multi-objective use. A GPTIPS run with the following settings was performed:

Population size = 500
Number of generations = 500
Tournament size = 10
$D_{\max} = 6$
$G_{\max} = 20$
Function node set = {plus, minus, times, protected divide, square, protected square root, protected log, exponent, sin, cosine}.
Terminal node set = {1664 DRAGON descriptors $x_1 - x_{1664}$, ephemeral random constants in the range [-10 10]}.

The default GPTIPS multigene symbolic regression function was used in order to minimize the root mean squared error (RMSE) on the training data. Model complexity, which was also minimized, was taken as the total number of nodes in a GP model.

The following recombination operator event probabilities were used:

crossover events = 0.85
mutation events = 0.1
direct reproduction = 0.05.

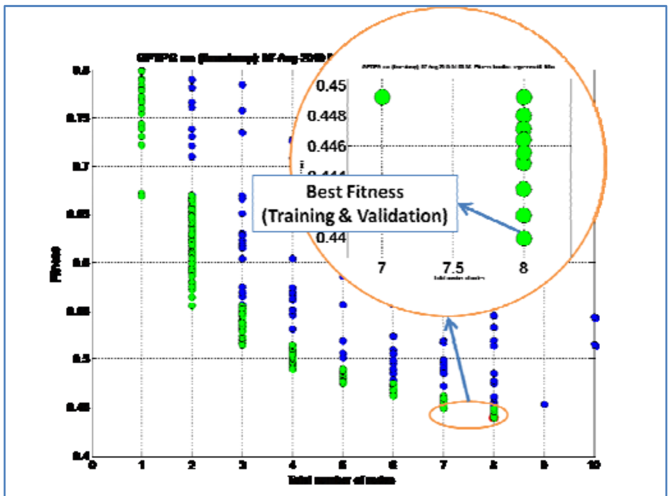
The following sub-event probabilities were used:

high level crossover = 0.2
low level crossover = 0.8
subtree mutation = 0.9
replace input terminal with another random terminal = 0.05,
Gaussian perturbation of random constant = 0.05 (with standard deviation of Gaussian = 0.1).

These settings are based on those used by Searson *et al* in [8].

X. Results

Fig. 4 shows an example of the fitness (RMSE) against complexity plot obtained at the end of a MOGP run and gives an example of a model chosen from the Pareto optimal front. It may be observed that the model chosen, with best fitness for training and validation datasets, is achieved at a higher complexity to others on the front. In general, another selection from the Pareto optimal models (e.g. a simpler model with slightly poorer predictive ability) may in some cases be preferable.



Best Fitness (Training & Validation) Model

$$y = 1.45x_{[illegible]} - 1.566x_{[illegible]} + 0.979x_{[illegible]} + 0.483x_{[illegible]} + 0.7452x_{[illegible]} + 0.01356x_{[illegible]} + 0.658x_{[illegible]} - 0.4811x_{[illegible]} - 0.6427$$

Fig. 4. Fitness (RMSE) against total number of model nodes.

The authors of [14] report their results in terms of MAE (mean absolute error) for two test sets (referred to in their paper as Validation set 1 (339 compounds) and Validation set 2 (110 compounds) that comprise the whole test set used here.

In order to compare with results [14] and [7], the mean absolute error (MAE) is calculated. In terms of MAE, the evolved model has MAE(training) = 0.2934, MAE(holdout) = 0.3376 and MAE(test) = 0.3944. This compares to our earlier results, that used a single objective algorithm minimizing 'goodness of fit' to the data [7] of MAE(training) = 0.3292, MAE(holdout) = 0.3573 and MAE(test) = 0.3518.

While the chosen model has a relatively lower performance on the testing data as compared the model obtained in [7], in terms of the second objective (model complexity) a very compact, linear model has been obtained containing 8 variables automatically selected from 1664 possible variables. Other models in the population contained both linear and non-linear terms.

In [14] the results of a number of individual models (and ensemble models) are reported, built using various descriptor packages and modelling techniques. Some of these models consider the "applicability domain" (AD) of the compounds (i.e. whether the compounds lie in the region of descriptor space deemed to be suitable for generating a prediction) whereas others do not employ AD considerations. In general, models that consider AD give more accurate predictions but only the results of the non AD models using the DRAGON descriptors are repeated here.

The first DRAGON descriptor based model is a support vector machine (SVM; [15]) regression that yields MAE(Validation set 1) = 0.37 and MAE(Validation set 2) = 0.42. This corresponds to an MAE(test) = 0.38. The second DRAGON based model is a *k*- nearest neighbour (*k*-NN) approach that achieves MAE(Validation set 1) = 0.29, MAE(Validation set 2) = 0.43 corresponding to MAE(test) = 0.32. Hence it can be seen that the evolved model has achieved predictive performance of the order of the current state of the art empirical modelling methodologies while ensuring that a low complexity structure is obtained. In this case, a linear model with a small number of descriptors from the 1664 available.

XI. CONCLUSIONS

In this article we have used the multigene symbolic regression capabilities of GPTIPS in conjunction with the NSGA-II [8] multi-objective algorithm and demonstrated it with an application in which a population of predictive symbolic QSAR models of *T. pyriformis* aqueous toxicity was evolved. This population contained a well defined Pareto front of models that trade off complexity against accuracy. For the best performing model (in terms of RMSE) it was demonstrated that the model is still very compact and offers similar high performance to recently published QSAR models of the same data.

We hope we have shown that multigene symbolic regression is an alternative and complementary approach to existing empirical modelling and data analysis techniques and that the multi-objective extension allows the simultaneous optimization of two competing objectives of model 'goodness of fit' to data and model complexity. It is also an approach that is facilitated by the free GPTIPS toolbox for MATLAB, a program that is used widely in academia and industry.

As a result of the work described in this paper, we hope to eventually release an updated version of GPTIPS, with support for multi-objective symbolic regression.

REFERENCES

[1] Koza J.R., Genetic programming: on the programming of computers by means of natural selection, The MIT Press, USA, 1992.

[2] Poli R., Langdon W.B. & McPhee N.F., A field guide to genetic programming, Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008.

[3] Alfaro-Cid E., Esparcia-Alcázar A.I., Moya P., Femenia-Ferrer B., Sharman K., Merelo J.J., Modeling pheromone dispensers using genetic programming. In Lecture Notes in Computer Science, Vol. 5484/2009, 635-644, 2009.

[4] Kordon, A.K., Future trends in soft computing industrial applications, Proceedings of the 2006 IEEE Congress on Evolutionary Computation, 7854-7861, 2006

[5] Searson, D., GPTIPS: Genetic programming & symbolic regression for MATLAB, http://gptips.sourceforge.net, 2009.

[6] Hinchliffe M.P., Willis M.J., Hiden H., Tham M.T., McKay B., Barton, G.W., Modelling chemical process systems using a multi-gene genetic programming algorithm. Genetic Programming: Proceedings of the First Annual Conference (late breaking papers), 56-65, 1996.

[7] Searson D.P., Willis M.J., Montague G.A., Co-evolution of non-linear PLS model components, Journal of Chemometrics, 2, 592-603, 2007.

[8] Searson, D., Leahy, D.E., Willis, M., GPTIPS:An open source genetic programming toolbox for multigene symbolic regression, International MultiConference of Engineers and Computer Scientists (IMEC 2010), Vol I., Hong Kong, 2010.

[9] Deb, K., Pratap, A., Argawal, S., Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 10(4) , 181-197, 2002.

[10] Rodriguez-Vazquez , K., Fonseca, C.M., Fleming P.J., Multiobjective genetic programming: a nonlinear system identification application, Proceedings of the Genetic Programming 1997 Conference (late breaking papers), 207-212, 1997.

[11] Zitzler, E., Laumanns, M., Thiele, L., SPEA2: Improving the strength pareto evolutionary algorithm., Swiss Federal Institute of Technology: Technical Report TIK-103, 2001.

[12] Steinbeck C., Han Y., Kuhn S., Horlacher O., Luttmann E., Willighagen E., The Chemistry Development Kit (CDK): An open-source java library for chemo- and bioinformatics, J. Chem. Inf. Comput. Sci., 43, 493 - 500, 2003.

[13] Schultz T.W.; Yarbrough J.W., Woldemeskel M., Toxicity to Tetrahymena and abiotic thiol reactivity of aromatic isothiocyanates, Cell Biol. Toxicol., 21, 181-189, 2005.

[14] Zhu H., Tropsha A., Fourches D., Varnek A., Papa E., Gramatica P., Oberg T., Dao P., Cherkasov A., Tetko I.V., Combinatorial QSAR modeling of chemical toxicants tested against Tetrahymena pyriformis, J. Chem. Inf. Model.,48, 766 -784, 2008.

[15] Vapnik, V.N., The nature of statistical learning theory, second edition, Springer-Verlag, New York, 2000.