Predicting the Toxicity of Chemical Compounds Using GPTIPS: A Free Genetic Programming Toolbox for MATLAB

Dominic P. Searson, David E. Leahy, and Mark J. Willis

Abstract In this contribution GPTIPS, a free, open source MATLAB toolbox for performing symbolic regression by genetic programming (GP) is introduced. GPTIPS 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 functionality of GPTIPS 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. It is shown that the low-order "multigene" GP methods implemented by GPTIPS can provide a useful alternative, as well as a complementary approach, to currently accepted empirical modelling and data analysis techniques. GPTIPS and documentation is available for download at http://sites.google.com/site/gptips4matlab/.

Keywords Genetic programming · Symbolic regression · QSAR · Toxicity

1 Introduction

Genetic programming [6] is a biologically inspired machine learning method that evolves computer programs to perform a task. It does this by randomly generating a 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 (hopefully) solve the task well.

When the task is building an empirical mathematical model of data acquired from a process or system, the GP is often known as symbolic regression. Unlike traditional regression analysis (in which the user must specify the structure of the

D.P. Searson (✉)
Northern Institute for Cancer Research, Newcastle University, Newcastle upon Tyne, UK
e-mail: d.p.searson@ncl.ac.uk

S.-I. Ao et al. (eds.), *Intelligent Control and Computer Engineering*,
Lecture Notes in Electrical Engineering 70,
DOI 10.1007/978-94-007-0286-8_8,

Table 1 Selected chemical engineering applications of GP

Authors	Application area	Model/application
Greeff and Aldrich [2]	Acid pressure leaching	Extent of dissolution model
McKay et al. [9]	Distillation column	Inferential Sensor
Grosman and Lewin [3]	Catalytic reaction	Reaction rate model
Hinchliffe and Willis [4]	Cooking extruder	Dynamic process model
Madar et al. [8]	Polymerisation	Dynamic process model
Wang and Li [17]	Distillation column	Optimise sequence
Seavy et al. [14]	Vapour liquid equilibrium	Hybrid model

model), GP automatically evolves both the structure and the parameters of the mathematical model. Symbolic regression has had both successful academic [1] and industrial applications in a variety of disciplines. For instance, in the discipline of Chemical Engineering, an overview of selected applications of GP is given in the table above.

In all the applications cited in Table 1 there is a common theme; GP is used for symbolic regression and, when using industrial data, the evolved models are shown to have better or comparable accuracy to alternative nonlinear modeling approaches such as neural networks.

The purpose of this chapter is to introduce a free open source MATLAB toolbox called GPTIPS [12, 13] that was written for the specific purpose of performing symbolic regression. GPTIPS employs a unique type of symbolic regression called "multigene" symbolic regression [5] that evolves linear combinations of non-linear transformations of the input variables. When the transformations are forced to be low order (by restricting the GP tree depth) this, in contrast to "standard" symbolic regression, allows the evolution of accurate, relatively compact mathematical models of predictor – response (input – output) data sets, even when there are a large number of input variables. Hence, the authors believe that GPTIPS provides a useful, free and complementary alternative to current data analysis techniques and has a broad spectrum of applicability across many scientific and engineering disciplines.

This chapter is structured as follows. Section 2 provides a brief overview of GP. Next, Sect. 3 discusses the multigene low order GP approach that GPTIPS implements. In Sect. 4, some of the features of GPTIPS are described. In Sects. 5–8, the capabilities of GPTIPS are demonstrated by using it to evolve an accurate, relatively 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 Sect. 9 we provide some concluding remarks.

2 Genetic Programming

The evolutionary computational (EC) method of GP evolves populations of symbolic tree expressions to perform a user specified task. A comprehensive, free to download introduction to GP and review of the literature is provided by Poli et al. [10] but 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 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 [9, 13]. 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.

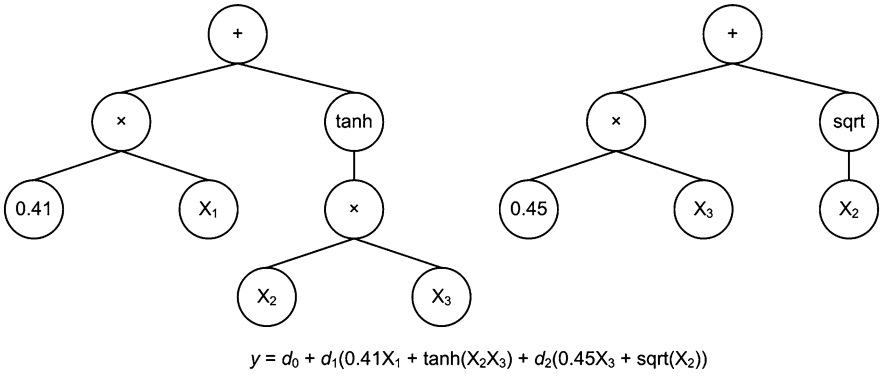

Fig. 1 Example of a multigene symbolic model

3 Multigene Symbolic Regression

Typically, symbolic regression is 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 may be designated as the input variable x_i.

In contrast, in multigene symbolic regression each symbolic model (and 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". 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 or 5 nodes) often allows the evolution of relatively compact models that are linear combinations of low order non-linear transformations of the input variables.

For each model, the linear coefficients are estimated from the training data using ordinary least squares techniques. Hence, 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. In Hinchliffe et al. [5] it was shown that multigene symbolic regression can be more accurate and computationally efficient than the standard GP approach for symbolic regression and Searson et al. [13] demonstrated that the multigene approach could be successfully embedded within a non-linear partial least squares algorithm.

In GPTIPS, the initial population is constructed by creating individuals that contain randomly generated GP trees with between 1 and $G_{\max}$ genes. During a GPTIPS

run, genes are acquired and deleted using a tree crossover operator called 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. If the ith gene in an individual is labelled G_i then a two point high level crossover is performed as in the following example. Here, the first parent individual contains the genes $(G_1G_2G_3)$ and the second contains the genes $(G_4G_5G_6G_7)$ where $G_{\max} = 5$. Two randomly selected crossover points are created for each individual. The genes enclosed by the crossover points are denoted by $< \ldots >$.

$$(G_1 < \boldsymbol{G}_2 > G_3) \quad (G_4 < \boldsymbol{G}_5\boldsymbol{G}_6\boldsymbol{G}_7 >)$$

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

$$(G_1G_5G_6G_7G_3) \quad (G_4G_2)$$

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 an individual containing more genes than $G_{\max}$ then genes are randomly selected and deleted until the individual contains $G_{\max}$ genes.

In GPTIPS, standard GP subtree crossover is referred to as low level crossover. In this case, a gene is selected randomly from each parent individual, standard subtree crossover is performed and the resulting trees replace the parent trees in the otherwise unaltered individual in the next generation. GPTIPS also provides several methods of mutating trees.

The user can set the relative probabilities of each of these recombinative processes. These processes are grouped into categories called events. The user can then specify the probability of crossover events, direct reproduction events and mutation events. These must sum to one. The user can also specify the probabilities of 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.

4 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 [7], 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. In addition, 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 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.

4.1 Using GPTIPS for Symbolic Regression

An example of a simple configuration file for multiple gene symbolic regression is shown in Fig. 2. It is assumed that data is located in the current directory in the file mydata.mat and comprises the training input data variable (xtrain), the training output variable (ytrain) as well as a testing data set (xtest and ytest). The data should be arranged by columns, e.g. the nth column of xtrain should contain the observations of the nth input variable.

In this example, only a few GPTIPS settings are specified. Any user parameters not explicitly set automatically use the default values. However, the user must at least specify the fitness function, the input and output data and the function nodes to be used in their configuration file. This configuration file first sets population size $= 100$ and number of generations $= 100$. The fitness function is then specified (i.e. the name of the M file) as is the fact that this is an error minimisation problem.

Next, the user data file mydata.mat is loaded and the variables within this file (here called xtrain, ytrain, xtest and ytest but they could be called anything) are assigned. After this is the number of input variables is set as the number of columns in the inputs training data matrix. Next, multigene mode is enabled and the maximum

```
function gp = my_config(gp);

gp.runcontrol.pop_size = 100;
gp.runcontrol.num_gen = 100;
gp.fitness.fitfun = @regressmulti_fitfun;
gp.fitness.minimisation = true;

load mydata
gp.userdata.xtrain = xtrain;
gp.userdata.ytrain = ytrain;
gp.userdata.xtest = xtest;
gp.userdata.ytest = ytest;

gp.nodes.inputs.num_inp = size(gp.userdata.xtrain,2);

gp.genes.multigene = true;
gp.genes.max_genes = 4;
gp.treedef.max_depth = 5;
gp.nodes.functions.name = {'times','minus','plus'};
```

Fig. 2 Example GPTIPS configuration file for multigene symbolic regression

number of genes per individual is set to 4. The maximum tree depth is then set to 5. Finally, the function nodes times, minus and plus are specified.

5 Evolution of a Predictive Model of Aqueous Chemical Toxicity Using GPTIPS

In the remainder of this article we will demonstrate how we have used GPTIPS to evolve a predictive QSAR model of aquatic toxicity for chemical compounds, based on their molecular structure.

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 or behaviour 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. A number of 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 [15].

A QSAR modelling scenario involves 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 drug candidate 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. Hence, the development of effective QSAR modelling methods continues to present a very real and relevant challenge.

There are a number of strategies & protocols for experimentally evaluating chemical toxicity. One commonly accepted method is the measurement of the growth inhibition of ciliated protozoan *T. pyriformis* [18]. There are freely available aquatic toxicity data for more than 1000 compounds, due to the efforts of Schultz and colleagues [11]. Zhu et al. [18] 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 GPTIPS 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 Zhu et al. [18].

6 Data

The *T. pyriformis* toxicity values (i.e. the response **y** data) are measured as the logarithm of the 50% growth inhibition concentration $\log(\mathrm{IGC50}^{-1})$. The data available for training QSAR models contains 644 compounds and another 449 compounds are used 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 available from the EU CADASTER website at http://www.cadaster.eu/node/65. To mitigate against the effects of overfitting, 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" individual (as evaluated on the training data) is then evaluated on the holdout validation set. The individual that performs best on the holdout set (over the course of the run) is stored and may be accessed after the run.

7 GPTIPS Run Settings

A GPTIPS run with the following settings was performed: Population size = 500, Number of generations = 500, Tournament size = 12 (with lexicographic selection

pressure), $D_{\max} = 4$, $G_{\max} = 8$, Elitism = 0.01% of population, function node set = {plus, minus, times, tanh, sin}, terminal node set = {1664 DRAGON descriptors $x_1 - x_{1664}$, ephemeral random constants in the range [−1010]}. The default GPTIPS multigene symbolic regression function was used in order to minimise the root mean squared prediction error on the training data.

The following (default) 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 randomly selected constant = 0.05 (with standard deviation of Gaussian = 0.1). These settings are not considered 'optimal' in any sense but were based on experience with modelling other data sets of similar size. The run took approximately 15 minutes on a PC with a dual core processor running at 2.2 GHz with 3.5 GB of RAM.

8 Results

The model that performed best on the holdout validation data was chosen. This model has coefficients of determination (i.e. proportion of the variation in the response explained by the model) of R^2 (training) = 0.83, R^2 (holdout) = 0.78 and R^2 (test) = 0.78. In Zhu et al. [18] the results are reported in terms of MAE (mean absolute error) for two test tests referred to in the paper as Validation set 1 (339 compounds) and Validation set 2 (110 compounds) that comprise the whole test set used here. In terms of MAE, the evolved GPTIPS model has MAE(training) = 0.3292, MAE(holdout) = 0.3573 and MAE(test) = 0.3518.

Zhu et al. [18] report the results of a number of individual models, 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 [16] 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 GPTIPS model lies between the SVM and the k-NN approaches, i.e. GPTIPS can achieve predictive performance of the order of the current state of the art empirical modelling methodologies.

GPTIPS was used to mathematically simplify and export the evolved model as a PNG graphics file. This is shown in Fig. 3.

It can be seen that the evolved model is reasonably compact, consists of both linear terms and low order non-linear transformations of the inputs and has selected a small number of descriptors from the 1664 available.

$$
\begin{aligned}
y = {} & -2.092 - 0.7548x_{911} + 0.7548x_{1558} - 0.8997\tanh(\tanh(x_{1426})) \\
& + 0.09443(x_{911} - x_{1558})x_{654} - 0.1481x_{1552} \\
& + 0.1481x_{391} - 0.2489x_{1429} - 0.2489\sin(x_{967} - x_{709}) \\
& + 0.7143x_{1245} - 0.5978x_{1662} - 0.5978\tanh(x_{1429} + x_{525}) \\
& + 0.7802x_{1426} + 0.7802x_{1563}
\end{aligned}
$$

Fig. 3 Graphical rendering of evolved symbolic *T. pyriformis* toxicity model

9 Conclusions

In this article we have introduced the multigene symbolic regression capabilities of GPTIPS and demonstrated it with an application in which a predictive symbolic QSAR model of *T. pyriformis* aqueous toxicity was evolved. It was demonstrated that the evolved model is compact and offers similar high performance to recently published QSAR models of the same data. The point of this article is not to assert that multigene symbolic regression (using low order non-linear transforms of the inputs) is better or worse than other methods, but that it is an alternative and complementary approach to existing empirical modelling and data analysis techniques. 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.

References

1. 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, pp. 635–644. Springer, Berlin/Heidelberg (2009)
2. Greeff, D.J., Aldrich, C.: Empirical modeling of chemical process systems with evolutionary programming. Comp. Chem. Eng. **22**, 995–1005 (1998)
3. Grosman, B., Lewin, D.R.: Automated nonlinear model predictive control using genetic programming. Comp. Chem. Eng. **26**, 631–640 (2002)
4. Hinchliffe, M.P., Willis, M.J.: Dynamic systems modelling using genetic programming. Comp. Chem. Eng. **27**(12), 1841–1854 (2003)
5. 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. In: Genetic Programming: Proceedings of the First Annual Conference (late breaking papers), pp. 56–65. MIT Press, Cambridge (1996)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
7. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), 2002
8. Madar, J., Abonyi, J., Sziefert, F.: Genetic programming for the identification of nonlinear input-output models. Ind. Eng. Chem. Res. **44**, 3178–3186 (2005)
9. McKay, B., Willis, M.J., Barton, G.W.: Steady-state modeling of chemical process systems using genetic programming. Comp. Chem. Eng. **21**, 981–996 (1997)
10. 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)
11. 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)

12. Searson, D.P., Leahy, D.E., Willis, M.J.: GPTIPS: An open source genetic programming toolbox for multigene symbolic regression. In: Lecture Notes in Engineering and Computer Science: Proceedings of the International Multiconference of Engineers and Computer Scientists, IMECS 2010, Hong Kong, 17–19 March 2010
13. Searson, D.P., Willis, M.J., Montague, G.A.: Co-evolution of non-linear PLS model components. J. Chemom. **2**, 592–603 (2007)
14. Seavey, K.C., Jones, A.T., Kordon, A.K.: Hybrid genetic programming – First-principles approach to process and product modeling. Ind. Eng., Chem. Res. **49**, 2273–2285 (2010)
15. 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)
16. Vapnik, V.N.: The Nature of Statistical Learning Theory, second edn. Springer, New York (2000)
17. Wang, X., Li, Y.: Synthesis of multicomponent product separation sequences via stochastic GP method. Ind. Eng. Chem. Res. **47**, 8815–8822 (2008)
18. 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)