Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 8858756, 7 pages
https://doi.org/10.1155/2022/8858756

*Research Article*

# The Train Delay Model Developed by the Genetic Programming Algorithm

**Tomas Brandejsky**

*Department of Software Technologie, Faculty of Electrical Engineering and Informatics University of Pardubice, Pardubice 532 10, Czech Republic*

Correspondence should be addressed to Tomas Brandejsky; tomas.brandejsky@upce.cz

Received 25 September 2020; Revised 30 November 2020; Accepted 17 December 2021; Published 31 January 2022

Academic Editor: Alessandro Severino



The paper discusses the problem of probability distribution category identification of train delay data by a genetic programming algorithm. This train delay frequency function and the probability distribution simply derived from it are significant to train traffic modelling and management. The genetic programming algorithm was used as an uninformed tool to prevent the influence of a priori information, which should be biased. The real traffic data were aggregated into predefined bins and then the frequencies of the individual delays were computed. The genetic programming algorithm was used in the next step as a symbolic regression tool to discover their frequency function in the form of an algebraic expression. The results concluded that although data has no known distribution, their distributions are similar to exponential ones.

## 1. Introduction

Train delays represent inconvenience of the rail transport. They are unwelcome not only in passenger rail transport but also in freight rail transport where delays disrupt logistic chains. The most sensitive is the combination of passenger and freight rail operations for a large difference in the passenger and freight train dynamics which results in complicated train operation scheduling and management.

*1.1. Relevant Research.* Rail transport delay modelling is significant for identification of delay causes which is essential for this delay reduction or even elimination. The delay modelling is also necessary for rail traffic modelling, control, scheduling, and future rail network state prediction. It is also necessary to mention that delays are typically computed for trains, not for passengers [1]. It is significant if we reason that the preservation of the train change possibility spreads a delay to other trains and can increase the magnitude of this delay, e.g., for passengers transferring between lines. A similar situation occurs in freight train transport if the wagons are switched between different trains.

The train delay models gain different forms. They can be based on a precise model of the rail network, train dynamics, and train timetable [2] on one side and a simplified probabilistic model on the other. These simplified probabilistic models are frequently applied for their easier evaluation and maintenance. For their application, it is required to identify probability distribution functions for all variables of the model. These distribution functions or frequency functions can be obtained from the model or from the data measured in real rail traffic. Analysis of the real traffic data represents a task of large or even Big data size problem. There are many variables influencing railway related processes and a huge number of records. To solve the traffic delay problem by probabilistic techniques, it is necessary to compute train delay distribution or frequency functions applicable in train delay management and train delay models.

It is impossible to reason about the development of the "absolutely" precise model for lack of related data and their uncertainty caused, e.g., by measurement errors. For herein

presented research, only the data about delayed trains were available, but not the data about nondelayed trains, their load, weather, large sport or cultural activities, and so on, and especially no data about operational dependencies like prescribed waiting rules between passenger trains in each station. Only the common preferences between different train categories were defined.

The main reason of this research was to find delay frequency function (FF) for delayed trains of a given category. Such a function can be simply transformed into relative frequency and probability density functions. The infrastructure owner was using exponential distribution in its models, but there was suspicion of imprecision of this model and the possibility that different train categories should be described by different distribution models due to different dynamics and operation rules. The genetic programming algorithm was chosen to find FFs describing observed data as objective tools with little a priori information.

The use of genetic programming algorithm (GPA) was decided on the basis of the observation in [3] that no standard probability distribution category FF describes data perfectly and the closest is an exponential distribution.

*1.2. Genetic Programming Algorithm Evolution.* Genetic programming (GP) was developed by J. Koza [4] on the basis of genetic algorithm (GA). Koza continued research of John Holland on "Adaptation in Natural and Artificial Systems" with the idea that programming can be transformed into optimization of a randomly generated population of individuals representing instructions and parameters. The state space of GPA is discrete in view of the change of operators (called functions) like addition, subtraction, or substitution during mutation and crossover operations. Latter many alternatives to GPA were developed like grammar evolution (GE), gene expression programming (GEP), or Cartesian GP (CGP). GE tries to reach better efficiency by application of a user-specified grammar (usually a grammar in Backus–Naur form) [5]. Gene expression programming (GEP) represents a different approach suitable to solving the symbolic regression (SR) problems [6], applying not only replication and mutation operations but also the transposition and recombination ones. A graph-based Cartesian GP (CGP) [7] is a GP algorithm where the candidate solutions are represented as a string of integers of a fixed length that is mapped to a directed graph. CGP can efficiently represent such structures as mathematical equations, computer programs, or neural networks. Another modification of genetic programming is represented by a hybrid single node genetic programming (SNGP) [8] [9]. SNGP is a rather new graph-based GP system that evolves a population of individuals, each consisting of a single program node. Similarly to CGP, the evolution is provided by a hill-climbing algorithm using a single reversible mutation operator. The SNGP represents a very promising development of GPs. Also, the original GPAs were widely studied since its introduction, for example, in ([10–12]), and these works have allowed its expansion. A symbolic regression takes specific position between the GPA applications. The goal of SR is to find a function modelling training data. As a technique of uninformed computer learning, the GP has large application potential.

SR is also easily verifiable, and thus it serves as a test bed frequently. Unfortunately, till now, symbolic regression has had some weaknesses, especially problems with large training data sets, e.g., in Big data applications and low efficiency if human comparative quality results are required. Possible ways of their elimination will be discussed further in this study. According to the paper [13], genetic programming (GP) is a particularly interesting machine learning (ML) algorithm when dealing with symbolic regression.

The GPAs directly evolve mathematical expressions [14], typically represented by a tree structure. While GP is understood to be capable of generating white-box models, i.e., human-interpretable expressions in simple form, the evolved models are often overcomplicated and far from being interpretable [15]. These problems tend to improve, especially in the areas of accurate symbolic regression, hybrid evolutionary algorithms, and some other approaches. Depending on the purpose, any rapidly developed or precise model is searched. While linear regression means findings of the coefficients of the chosen function to best fit the data, symbolic regression searches for a suitable function. If the linear regression result is not good, it is possible to choose a different function. The quality of the regression result depends on the selection of this function. Symbolic regression is capable of going far. Its goal is to find such a function to fit the data, not only its parameters. Since the first application of genetic programming [4], many studies and approaches to solve this problem have been published.

Except the abovementioned problems, many authors published in their works the need to use large populations of thousands or more individuals. This is caused by the need to optimize many constants and the sensitivity of individual selection on them. If the modelled problem is complicated and described by training data containing tens or even hundreds of variables (e.g., Big data problems), it is necessary to reduce this amount to prevent loss of GPA efficiency [16] to keep reasonable processing time. In these cases, the efficiency of machine learning algorithms including GPA significantly decreases and it is better to use a separate algorithm to identify significant features, e.g., by the feature selection algorithm [17].

*1.3. Hybrid Evolutionary Algorithms.* Unpublished work [11] prefigured small but significant research in an area of hybrid evolutionary systems combining GPA with other techniques with the intent to eliminate some of their weaknesses. This work is continued in the paper [18]. Unfortunately, the main research areas of both authors of this work moved out of evolutionary programming and thus these papers were not extended by them. Luckily, they found a continuation in [19]. Raidl assigned multiplication parameters to each node of the parse tree and optimized their magnitudes by the method of least squares. The nonlinear optimization method was used in [18, 20]. The problem of these early works was in large computational requirements

which have allow only a few steps of the time-consuming nonlinear optimization to be applied to each new solution to keep the total running times acceptable. Particle swarm optimization (PSO) was used for arbitrary constant magnitudes optimization in [21]. PSO is a population using a computational optimization method developed by Eberhart and Kennedy in 1995 [22], inspired by the social behaviour of bird flocking. The system is initialized with a population of random solutions and searches space by the coordinated movement of a particle swarm. Accurate SR tries to eliminate well-known problems of original early GPA applications. They were less efficient, overcomplicated, and imprecise. The accurate SR as it was described in [23] applies structural risk minimization based on the Vapnik–Chervonenkis dimension to estimate the difference between the generalization and the empirical error. The problem of these algorithms is that they move GPAs close to overfitting limitation known from the other machine learning algorithms. These works prove that in the hybrid evolutionary algorithm it is possible to apply in the hybrid algorithm any optimization tool in combination with GPA. Thus, there evolutionary strategy of genetic algorithms can be placed too, as in this work.

## 2. Materials and Methods

*2.1. Analysed Train Delay Data Set.* To analyse train delays on the Czech national railway network, the data describing train delays within three months of 2011 was used. Originally, the data were stored in Oracle data warehouse. These data were analysed in [3] with recommendation to use exponential distribution in models.

There were problems with trains free of delays because such trains were not presented in the filtered data provided for the research. Thus, the number of observations of the train passing without delay was smaller than in reality, and they were not used to identify delay FF.

Regardless of the above-described limitations, the amount of processed data was not totally satisfying the attributes of Big data, but it was difficult for processing. Big data are characterised by so called 3 or 4 v's. They are mean volume, variety, velocity, and veracity. The meaning of these terms is described as follows:

Volume—The quantity of generated and stored data. The Big data applications work with large amount of data that are hard to process by standard technology.

Variety—Variety means the type and nature of the data. The data processed by the Big data systems are semi-structured or unstructured. It makes it difficult to process them by standard strongly typed relational database management system (RDBMS).

Velocity—Many Big data applications require high speed reactions and fast processing because they operate in real time. There is also the problem of the continuous incoming of a new data.

Veracity—Referring to data volume and their value, the data can vary too fast for accurate analysis.

Because only a small, time-limited sample of data (3 months) was analysed, the volume of analysed data was relatively small from Big data viewpoint, but its size was still many gigabytes. The data were also well structured. The velocity in this off-line analysis is not significant. Nevertheless, with respect to the future possibility of full data collection analysis, the Big data technology was used. There it is possible to imagine online analytics (with high requirements on velocity) and processing of full data collection representing all train movements in the past 20 years.

For the herein presented analysis, the basic programming model of Big data processing called MapReduce was used (selection of data related to the analysed category and aggregation on the basis of delay magnitude into the relevant bin).

There it is justifiable to expect distinct results for different train categories (Table 1) due to different operational rules like maximal speed limit, typical weight and length of the train limiting maximal acceleration, waiting for connections (and concluding delay propagation) in the case of passenger trains, and priority in access to railway line. Especially, the last two influences strongly determine delays. Figure 1 demonstrates an example of input data for passenger train categories.

*2.2. Data Preprocessing.* The original data set was transformed from the Oracle database into comma-separated values (CSVs) dale file and imported into the Apache Spark application. This application was written in *Python* using Jupyter Notebook and served to select data related to the studied train type and reduce data volume by their transformation into counts of train passes across measurement points with a specified delay. The following step was application of symbolic regression to model FF describing the data.

*2.3. Used GPA-ES Evolutionary Algorithm.* In this research, a modification of the hybrid GPA-ES algorithm for algebraic dependency modelling is used (standard GPA-ES was designed for symbolic regression of ordinary, linear, or nonlinear differential equations—e.g., 10, 24, or 25). The algorithm evolves equations on the basis of the minimization of residual errors—fitness. This algorithm combines the standard GP algorithm for solution structure development and an evolutionary strategy (ES) algorithm for optimization of parameters of each individual in a GPA population. Such a design of the hybrid evolutionary algorithm prevents situations when a well-structured solution (e.g., well-composed equation) but with wrongly estimated constants (coefficients) is eliminated from the population and replaced by an individual of worse structure but better fitted constants, which has worse evolutionary potential. A comprehensive introduction to GPA can be found in [26]. A complex review of the hybrid evolutionary algorithms is in [27], and an explanation of symmetric one-point crossover is described in [28]. Two different variants of crossover increase the efficiency of the Algorithm 1.

The size of the GPA population and the size of the ES populations related to each individual related to the GPA population are the most significant parameters of the GPA-ES algorithm. The influence of the population size was studied in many publications, for example the study conducted in [2].

Table 1: Explanation of the analysed train category abbreviations and the processed amount of data.

| Abbreviation | No. of records in the analysed database | Train category |
|---|---|---|
| EC | 430228 | EuroCity express |
| Ex | 374674 | Express train |
| IC | 211926 | InterCity express |
| Lv | 1118057 | Locomotive train |
| Nex | 612071 | Freight fast express |
| Mn | 377838 | Service train |
| Os | 8220824 | Passenger train |
| Pn | 1271189 | Unit freight train |
| R | 2454901 | Regional train |
| Sp | 550954 | Commuter train |

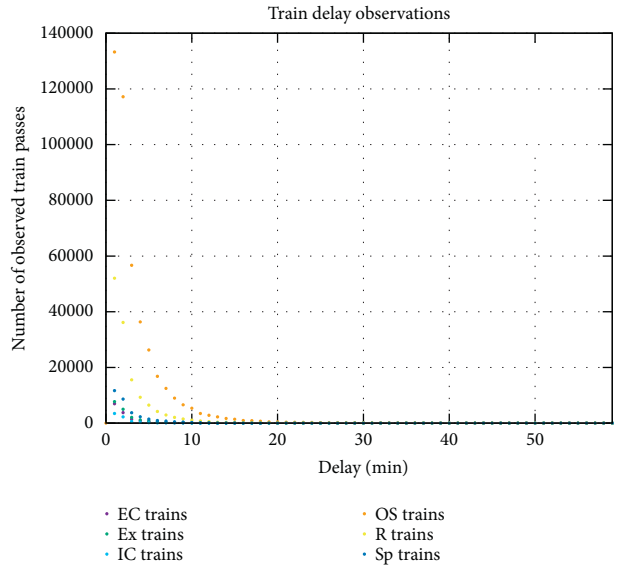


Figure 1: Example of train delay data histograms distinguished by train category.

Small GPA populations forces evolutionary pressure, and in the case of specific conditions it might speed up evolution. On one hand, in small populations there is an increased risk of getting stuck in local optima instead of global ones. Very large populations have problems with low speed and low efficiency of evolution frequently. On the other hand, they bring less dispersion of needed evolutionary cycles and higher reliability of solution discovery. Because extremely small GPA populations bring big dispersion of needed evolutionary cycles, it is difficult to predict the needed computational time. In the case of ES populations, analogical reasons are valid only if there are eliminated random influences of task switching and other sources.

During the initial stage of processing, the records about the delays of a selected type of train were reduced into a vector representing the delay discrete distribution FF by eliminating temporal and spatial information. There are present multiplication, exponential, divide, and factorial functions in the set of GPA functions to allow "discovery" of exponential or Poisson distribution related frequency functions (and similar ones), the candidates to possible solution. The number of reasoned data categories was limited to 20. In the case of large training data vectors, the less significant (outlier) data representing large delays were not reasoned, e.g., in the case of 1-minute delay resolution. Thus, the training data vector was a vector of 20 pairs (delay time and frequency).

Larger delay bins, like the extreme 100 minute one, contain less amount of accumulated noise, but such large bins also lose information. They lose information about delay distribution, and thus they might cause wrong results.

The use of grouping many delays, e.g., from interval <1, 30> minutes, can decrease noise caused especially by small number of samples in the given bin. On the other hand, such grouping represents a significant loss of information. It tends to be a simple function to find and in the extreme case, the linear FF instead of the exponential one can be found.

Because Figure 1 points to functions similar to exponential or Poisson distribution ones, the function set for GPA contains functions that allows to reconstruct them and build a class of similar ones. Thus, there has been a present factorial, but it has a limited scale of argument magnitudes due to the limitations of number representation in computer. Also, the number of distinguished the most significant (the smallest) delays from the interval <1, 20> minutes was reasoned. Exponential or power functions, divide or inverse functions ($1/x$), multiplication, addition, and subtraction are useful in FF regression, and thus they were incorporated

```
(1) FOR ALL individuals DO Initialize() END FOR;
(2) FOR ALL individuals DO Evaluate()=>fitness END FOR;
(3) Sort(individuals according to fitness);
(4) IF terminal condition is met THEN STOP END IF;
(5) FOR ALL individuals DO
       SELECT Rand() OF
   CASE a DO Mutate()=> new_individuals;
   CASE b DO Symmetric_crossover(i-th, (i+1)th individuals) => new_individuals;
   CASE c DO One_point_crossover(i-th, (i+1)th individuals) => new_individuals;
   CASE d DO Re-generating() => new_individuals;
   END SELECT;
END FOR;
(6) FOR ALL new_individuals DO
   New ES_algorithm_object with related ES_individuals and ES_fitness arrays
//for each GPA individual new independent parameter optimizer is created
FOR ALL ES_individuals DO Initialize() END FOR;
    Evaluate() => ES_fitness;
FOR ALL ES_cycles DO
   FOR ALL ES_individuals DO
      Evaluate() => ES_fitness;
   END FOR;
   FOR ALL ES_individuals DO
         Intelligent_crossover() => new_ES_individuals
         Evaluate() => new_ES_fitness;
   END FOR;
   FOR ALL ES_individuals DO
         IF new_ES_fitness < ES_fitness THEN
            ES_individual = new_ES_individual;
            ES_fitness = new_ES_fitness;
      END IF;
   END FOR;
   Sort(ES_individuals, ES_fitness);
END FOR;
new_individual = ES_individual[0]; new_fitness = ES_fitness[0];
END FOR;
(7) FOR ALL individuals DO IF new_fitness < fitness THEN
       individual = new_individual;
       fitness = new_fitness;
       END IF;
    END FOR;
(8) GOTO 3);
```

Algorithm 1: Used GPA-ES algorithm structure.

GPA function set. Herein the presented experiments are based on ungrouped data measured with 1-minute resolution containing the first 20 values (delays 1, 2, ..., 20 minutes).

## 3. Results and Discussion

The GPA-ES algorithm worked with a maximal limit of 10000 GPA evolutionary cycles and 200 ES cycles in each GPA one, with 100 GPA individuals and 200 individuals in each ES population. This algorithm was written in the *C++* language and it was executed as a single-thread task on a single core of the 24 HT cores Intel Xeon processor. Execution times were between 80210 seconds for Sp trains and 274081 seconds for Ex trains due to stochastic character of the algorithm and resulting function shape.

The experiment majority tend to the presence of power function '$^{\circ}$' in the resulting function. Add function '+' is the more useful (the more frequently present) than the divide function '/'. Also, the multiplication function '*' was frequently used. Factorial function '!' was never present in discovered functions. Thus, in future experiments, it can be left out and large data vectors can be used for probability density function learning.

The differences between passenger and freight train behaviour were observed rather in the coefficient magnitudes than in the structure of the discovered functions. All functions described in Table 2 are similar to FF of exponential distribution, but not identical.

Figures 2 and 3 provide examples of results for some train categories and allows comparing of discovered FF with original data frequencies.

Table 2: The FFs discovered by the genetic programming algorithm and residual errors of this estimation.

| Train category | Train category delay frequency function | Sum of error squares |
|---|---|---|
| EC | $2(x-1)+7020.79*0.497426^{x-1}$ | 2.40 E+05 |
| Ex | $(7610.04+(x-1)^2)((2x)^{-0.276312(x-1)})$ | 1.83 E+05 |
| IC | $4966.33(0.686090^x)$ | 1.17 E+05 |
| Lv | $(-3005.93+(3718.59(x-1)))(0.491252^{x-1})$ | 3.63 E+05 |
| Mn | $(1463.47+3(x-1))*(0.846728^{x-1})$ | 1.49 E+05 |
| Nex | $5010.84(x^{-0.130751(x-1)})$ | 5.96 E+05 |
| Os | $134069(x(x-1))^{-0.188082(x-1)}$ | 4.72 E+07 |
| Pn | $(10335.4(x-1))(0.51124^{(x-1)})$ | 1.03 E+07 |
| R | $53062.1*506948^{-0.00965108(x-1)}$ | 4.31 E+07 |
| Sp | $(x^{-0.461526(x-1)})(11695.3+x+(x-1)^2)$ | 3.69 E+05 |

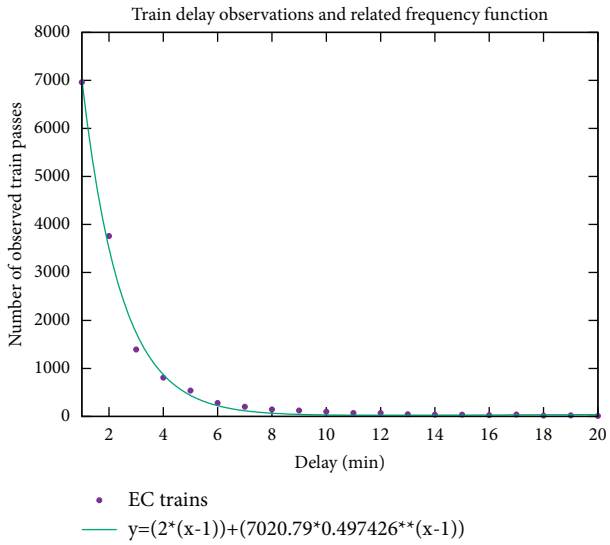


Figure 2: Example of an EC train category delay data histogram and related frequency function drawn as a continuous one to demonstrate its shape.

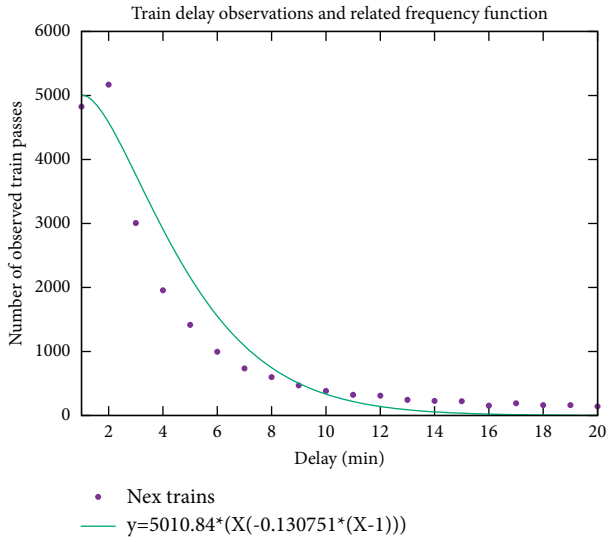


Figure 3: Example of Nex train category delay data histogram and related frequency function drawn as a continuous one to demonstrate its shape.

## 4. Conclusions

The presented study was devoted to the estimation of data distribution class by the application of the hybrid evolutionary algorithm GPA-ES and symbolic regression of frequency function equations.

The experiments described in this paper confirmed the exponential like distribution of train delays on the basis of FF regression from Czech railway network data. The evaluated amount of data was 6 GB and they were preprocessed by the Apache Spark application written in the *Python* language.

Founded FFs are similar to ones related to exponential distribution, but not identical.

The GPA-ES algorithm is capable to discover algebraic relations in applicable form for extremely small data vectors of 20 elements [24, 25].

## Data Availability

The csv data used to support the findings of this study were supplied by the state company: Správa železnic, s.o. (Rail Infrastructure Administration, state organization) under license and are subject of commercial confidentiality, so they cannot be made freely available. Address of the data owner is Správa železnic, s.o., Dlážděná 1003/7, PSČ 110 00 Praha 1, The Czech Republic.

## Conflicts of Interest

The author declares no conflicts of interest.

## Acknowledgments

The work was supported from ERDF/ESF "Cooperation in Applied Research between the University of Pardubice and Companies, in the Field of Positioning, Detection and Simulation Technology for Transport Systems (PosiTrans)" (No. CZ.02.1.01/0.0/0.0/17_049/0008394).

## References

[1] O. A. Nielsen, O. Landex, and R. D. Frederiksen, "Passenger delay models for rail networks, schedule-based modeling of transportation networks," in *Schedule-Based Modeling of*

*Transportation Networks: Theory and Applications*, pp. 1–23, Springer, Boston, MA, USA, 2009.

[2] A. Higgins, E. Kozan, and L. Ferreira, "Modelling delay risks associated with train schedules," *Transportation Planning and Technology*, vol. 19, no. 2, pp. 89–108, 1995.

[3] A. Kavicka, M. Bazant, and V. Zahorova, *Statistic Analysis of Data about Train Delays in the Railway Network*, United Progressive Alliance, Pardubice, Czech Republic, 2014.

[4] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992.

[5] C. Ryan, J. Collins, and M. O. Neill, "Grammatical evolution: evolving programs for an arbitrary language," in *Genetic Programming*, W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogary, Eds., Springer, Berlin, Germany, pp. 83–96, Lecture Notes in Computer Science, 1998.

[6] C. Ferreira, "Gene expression programming: a new adaptive algorithm for solving problems," 2001, https://arxiv.org/abs/cs/0102027.

[7] J. F. Miller and P. Thomson, J. Miller, Edited by R. Poli and W. Banzhaf, Eds., "Cartesian genetic programming," in *Genetic Programming*, P. Nordin and T. C. Fogarty, Eds., Springer, Berlin, Germany, pp. 121–132, Lecture Notes in Computer Science, 2000.

[8] D. Jackson, "Single node genetic programming on problems with side effects," in *Parallel Problem Solving from Nature - PPSN XII*, C. A. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds., Springer, Berlin, Germany, pp. 327–336, Lecture Notes in Computer Science, 2012.

[9] J. Kubalík, E. Alibekov, J. Žegklitz, and R. Babuška, "Hybrid single node genetic programming for symbolic regression," in *Transactions on Computational Collective Intelligence XXIV*, vol. 9770, pp. 61–82, Springer, Berlin, Germany, 2016.

[10] T. Brandejsky and I. Zelinka, Edited by I. Zelinka, O. E. Rössler, and V. Snášel, Eds., "Specific behaviour of GPA-ES evolutionary system observed in deterministic chaos regression," in *Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems*, A. Abraham and E. S. Corchado, Eds., Springer, Berlin, Germany, pp. 73–81, Advances in Intelligent Systems and Computing, 2013.

[11] J. Frohlich and C. Hafner, "Extended and generalized genetic programming for function analysis," *Submitted to Journal of Evolutionary Computation*, 1996.

[12] T. L. Lew, A. B. Spencer, F. Scarpa, K. Worden, A. Rutherford, and F. Hemez, "Identification of response surface models using genetic programming," *Mechanical Systems and Signal Processing*, vol. 20, no. 8, pp. 1819–1831, 2006.

[13] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, "Symbolic regression and feature construction with GP-GOMEAapplied to radiotherapy dose reconstruction of childhood cancer survivors," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018*, H. E. Aguirre and K. Takadama, Eds., pp. 1395–1402, ACM, Kyoto, Japan, July 2018.

[14] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009, https://science.sciencemag.org/content/324/5923/81.

[15] S. Luke and L. Panait, "A comparison of bloat control methods for genetic programming," *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, 2006.

[16] W. B. Langdon and B. F. Buxton, "Genetic programming for mining DNA chip data from cancer patients," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 251–257, 2004.

[17] J. Li, K. Cheng, S. Wang et al., "Feature selection: a data perspective," *ACM Computing Surveys*, vol. 50, no. 6, Article ID 94, 2017.

[18] C. Hafner and J. Frohlich, "Generalized function analysis using hybrid evolutionary algorithms," in *Proceedings of the Congress on Evolutionary Computation*, Washington, DC, USA, July 1996.

[19] G. R. Raidl, "A hybrid GP approach for numerically robust symbolic regression," in *Proceedings of the 3rd Annual Genetic Programming Conference*, J. Koza, Ed., pp. 323–328, Morgan Kaufmann, San Francisco, CA, USA, https://www.ac.tuwien.ac.at/files/pub/raidl.pdf.

[20] B. McKay, "Using a tree structured genetic algorithm to perform symbolic regression," *IET Conference Proceedings*, no. 5, pp. 487–492, 1995, https://digital-library.theiet.org/content/conferences/10.1049/cp_19951096.

[21] F. Qi, Y. Ma, X. Liu, and G. Ji, "A hybrid genetic programming with particle swarm optimization," in *Advances in Swarm Intelligence*, T. Ying, S. Yuhui, and M. Hongwei, Eds., Springer, Berlin, Germany, pp. 11–18, Lecture Notes in Computer Science, 2013.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference*, Perth, Australia, December 1995.

[23] Q. Chen, B. Xue, L. Shang, and M. Zhang, "Improving generalisation of genetic programming for symbolic regression with structural risk minimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '16)*, pp. 709–716, Association for Computing Machinery, New York, NY, USA, July 2016.

[24] T. Brandejsky, "Multi-layered evolutionary system suitable to symbolic model regression," in *Proceedings of the NAUN/IEEE.AM International Conferences. 2nd International Conference on Applied Informatics and Computing Theory*, pp. 222–225, WSEAS Press, Praha, Athens, 2011.

[25] T. Brandejsky, "Symbolic regression of deterministic chaos," in *Proceedings of the 17th International Conference on Soft Computing (MENDEL 2011)*, pp. 90–93, VUT v Brně, Brno, Czech Republic, June 2011, ISSN 1803-3814.

[26] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*, Lulu Enterprises UK Ltd, London, UK, 2018.

[27] T. El-mihoub, A. Hopgood, L. Nolle, and A. Battersby, "Hybrid genetic algorithms: a review," *Engineering Letters*, vol. 3, no. 2, 2006.

[28] R. Poli, W. B. Langdon, P. K. Chawdhry, R. Roy, and R. K. Pant, *Soft Computing in Engineering Design and Manufacturiing*, pp. 180–189, Springer, London, UK, 1998.