DEC-GEP Tool: Differential Evolutionary Chromosomal Gene Expression Programming

Brief Introduction:

This document introduces the Differential Evolutionary Chromosomal Gene Expression Programming tool (DEC-GEP). This tool was developed for data-driven symbolic regression applications. The DEC-GEP optimizes a complete genotype of expression trees for mathematical model synthesis with their constant coefficients from a training dataset. To illustrate an application of the DEC-GEP algorithm in a real-world data-driven modeling application, a symbolic regression model for the soft calibration of e-nose (the low-cost, solid state multisensor array measurements) is shown. You can define your custom functions and operators in DEC-GEP.

More details and citation for this tool:

Citation for this tool:

[1] Ari, D., Alagoz, B. B. (2023). A differential evolutionary chromosomal gene expression programming technique for electronic nose applications. Applied Soft Computing, 136, 110093. https://doi.org/10.1016/j.asoc.2023.110093

What is GEP and Symbolic Regression?

Gene expression programming (GEP) is an evolutionary algorithm that aims to produce computer programs or mathematical models from data sets. GEP is a genotype–phenotype evolutionary scheme that benefits from a linear, symbolic string (chromosome code) of fixed length in order to encode a genome of the program or mathematical model. Although its fixed length chromosome code, it can express variable, complex phenotypes of symbolic models (mathematical models, computer programs etc.)

One of the prominent applications of GEP is the symbolic regression that yields mathematical models for mathematical relation between input and output parameters. It directly synthesizes

equations that formulize mathematical relationships between input and output data from the real-world systems. Therefore, the GEP is capable of exploring mathematical relations between parameters from datasets, and this task is commonly known as symbolic regression. Symbolic regression models express the solution with mathematical symbols and structures. Figure 1 shows a GEP chromosome decoding example.

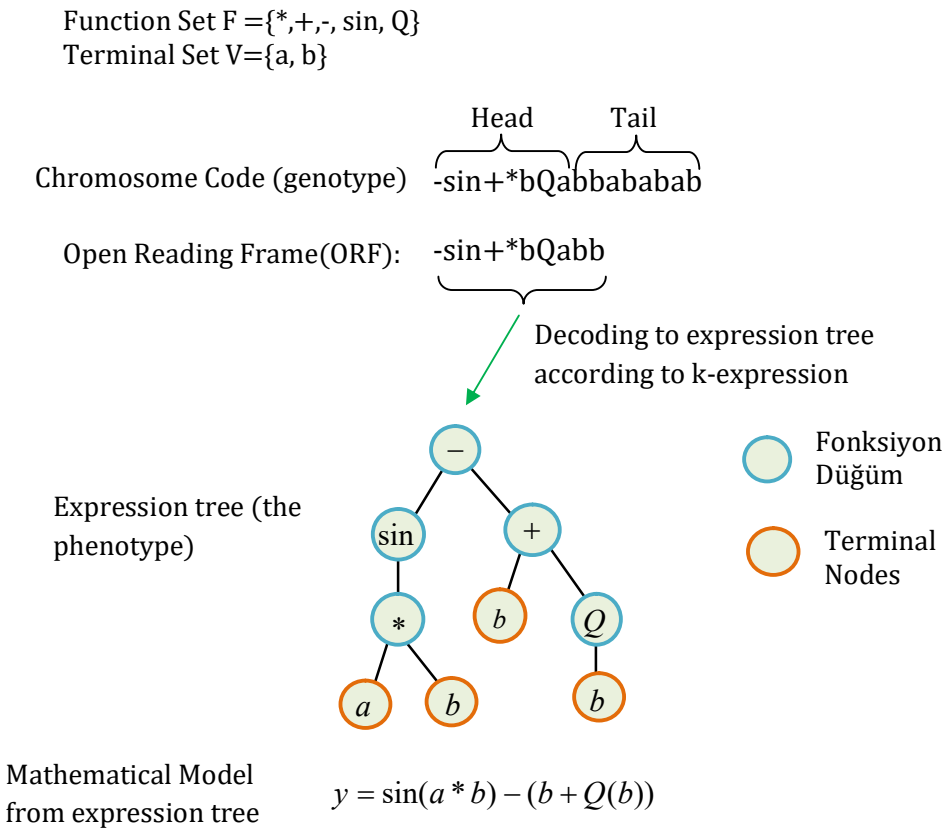


Figure 1. A DEC-GEP chromosome decoding example

DEC-GEP Algorithm:

DEC-GEP Chromosomes:

GEP requires specification of three descriptive sets by users. These are the function set (F), which expresses a collection of functions (operators, functional structures), a variable terminal set (V), which expresses a collection of input variables (input parameters of the symbolic model) and a constant terminal set (C), which defines constants of the symbolic model. Figure 1 shows

composition of a DEC-GEP chromosome frame. Table 1 describes compositions of each part of the DEC-DEP chromosome. The head root part involves codes only from the function set (F) that can ensure the start of the expression tree. The head body can involve codes from the function set, variable and constant terminal sets (F,V,C). Table 4 shows basic formulas to calculate length of the DEC-GEP chromosome.

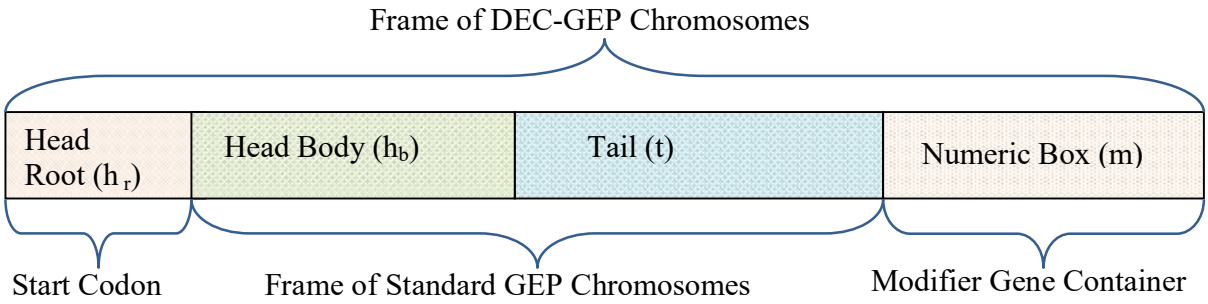


Figure 2. A DEC-GEP chromosome frame [1]

Table 1. Composition of the DEC-GEP chromosome parts [1]

Parts of Chromosome	**Compositions**	**Code Type**
Head Root (h_r)	A code from function set (F)	Integer Number
Head Body (h_b)	h_b number of codes from function and terminals (variables and constants) set group (F, V, C)	Integer Number
Tail (t)	t number of codes from terminals (variables and constants) sets (V, C)	Integer Number
Numeric Box (m)	$m = \|C\|$ number of numerical values in order to assign each constant in the constant set (C)	Real Number

Table 4. Length of the DEC-GEP chromosome[1]

Parameters of DEC-GEP Chromosomes	Values
Size of function set (F)	$\|F\|$
Size of variable set (V)	$\|V\|$
Size of constant set (C)	$\|C\|$
Length of head section (h) in GEP Chromosome	$h = h_r + h_b$
Length of tail section (t) in GEP Chromosome	$t = h(n_{ar} - 1) + 1$

Length of numerical box (m) in GEP Chromosome	$m = \|C\|$
Total length of the DEC-GEP Chromosome (c_L)	$c_L = h + t + m$

DEC-GEP Algorithm:

Figure 6 shows a flowchart of the DEC-GEP algorithm that describes interactions between the DE and the GEP[1]. The objective function of the DE algorithm was designed to perform the decoding of GEP chromosomes, and the performance evaluation of the evolved mathematical expressions for the dataset. The chromosomes of GEPs are optimized via the DE algorithm, and the objective function returns the Mean Square Error (MSE) performance that measures the fitness of resulting mathematical expressions (programs) to the training dataset. This scheme progressively enhances DEC-GEP chromosomes in order to find a better symbolic model from datasets.

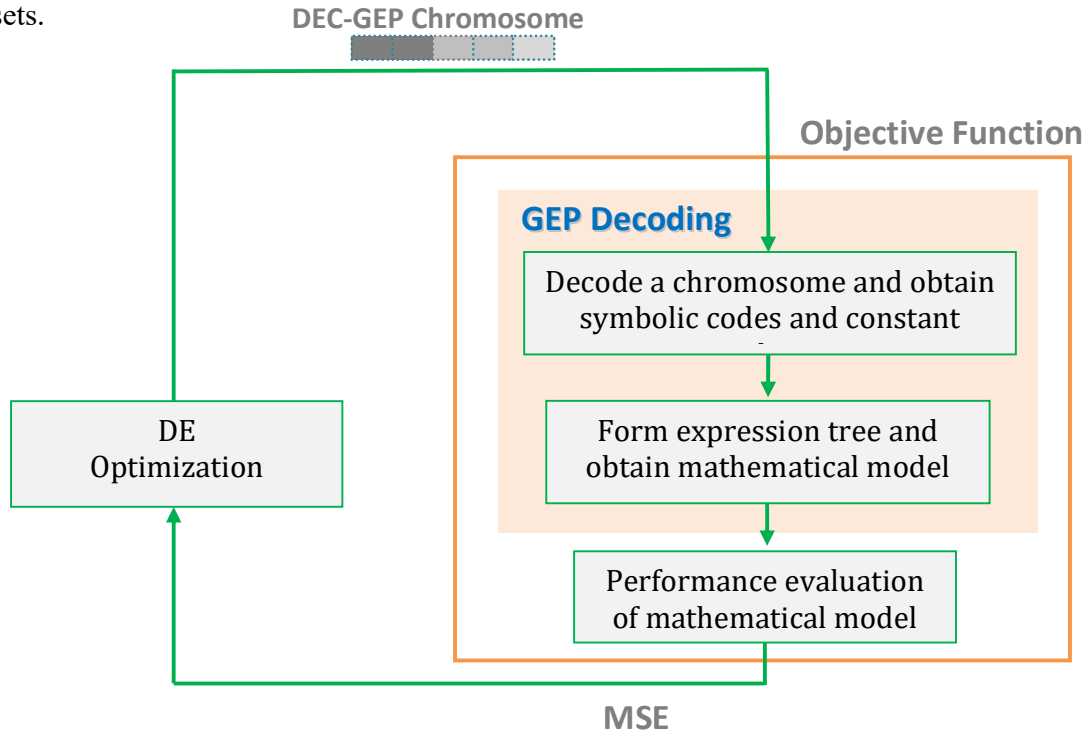


Figure 3. Flowchart of a DEC-GEP algorithm [1]

Getting Started:

To implement the DEC-GEP algorithm for e-nose calibration application, we used the DEC-GEP tool in order to obtain a symbolic regression model from the air quality dataset. In this application, measurement data from a solid state sensor array was used as inputs of the model to estimate the CO concentration measurements of the chemical analyzers. (The chemical analyzers are assumed to yield the ground truth data) Data from 8 sensors was utilized as input terminals, and the symbolic regression model estimates the CO concentration as the output of the model[1].

To perform this application, open **gep_main_enose.m** file and run it. It runs the DEC-GEP tool. After the DEC-GEP algorithm is completed, it yields an optimal symbolic regression model object (**BestModel**) and the **BestModel** object (symbolic model) is used in order to perform estimation of CO concentration.

Codes use some global parameters for configuration of the DEC-GEP tool. (Please do not change global variable declarations) After configuring some necessary user parameters, the *DEC-GEP function* is executed to obtain a data-driven symbolic regression model.

```
DEC_GEP(ConsValLow,ConsValMax,max_arity,Max_iteration,SearchAgents_no,
beta_min,beta_max,pCR,MesFlag,IsDraw)
```

To implement the training dataset, input data (features) and output data (predictions) should be assigned to the global variable **inputs** and **outputs**. For instance;

```
% Assigns input data from the loaded dataset
inputs = x';
% Assigns output data from the loaded dataset
outputs = yCO';
```

To define the function set F, one should compose the global variable **F** in the form of {'Function name',number of arguments} as follows:

```
F={'div',2;'ReelSqrt',1;'ReelLog',1;'cos',1;'tan',1;'cot',1;'sin',1;'*'
,2;'+',2;'-',2};
```

Also the variable terminal set (V) should be defined by using the global variable **x_var** and the constant terminal set (C) should be defined by using using the global variable **c_var**:

```
x_var = {'x1','x2','x3','x4','x5','x6','x7','x8'};  % Defines 8 input
terminals from x1 to x8
c_var = {'c1','c2','c3','c4','c5','c6','c7','c8'};% Defines 8 constant
terminals from x1 to x8
```

To calculate the predicted data and evaluate performance of the symbolic regression model, the *gep_compare function* is run.

```
[MSE, MAE, RAE, gep_model_data] = gep_compare(BestModel, inputs, outputs, best_constants);
```

Some important parameter settings and calling *DEC-GEP function* and *gep_compare function*:

```
clear all;close all;
clc;
% Global variable declaration of the DEC-GEP tool
% Please do not change global variable declarations
global inputs;            % Input data for symbolic regression model
global outputs;           % Output data for symbolic regression model
global head_length;       % Length of Head section in the GEP
chromosome
global best_constants;    % The optimal set of constants after DEC-GEP
algorithm is completed.
global BestModel;         % The optimal symbolic model after DEC-GEP
algorithm is completed.
global F;                 % Function set (F) to defines functions and
operators
global isPrint;           % Flag to enable to print report of DEC_GEP
at each iteration
global x_var;             % Variable set to define input terminals
global c_var;             % Constant set to define constant terminals
% Sets isPrint flag to report significant parameters at each iteration
('Yes' or 'No')
isPrint = 'No';
% Loads data from the specified file
load('./data/AirQualitySensor');
% Assigns input data from the loaded dataset
inputs = x';
% Assigns output data from the loaded dataset
outputs = yCO';
% Sets the head length of the GEP chromosome
head_length = 16;
% Defines the lower bound for constant values
ConsValLow = -1000;
% Defines the upper bound for constant values
ConsValMax = 1000;

% Defines the function set (F) for the GEP algorithm
% Format of set is the {'Function',number of arguments}
% Each row represents a function and its arity (number of arguments
(operands))
% Example:
% F = {'*', 2;   % Multiplication operator with arity 2
%      '+', 2;   % Addition operator with arity 2
%      '-', 2;   % Subtraction operator with arity 2
```

```
%      'sin', 1}; % Sinus function with the arity 1

F={'div',2;'ReelSqrt',1;'ReelLog',1;'cos',1;'tan',1;'cot',1;'sin',1;'*'
,2;'+',2;'-',2};

% Defines the variable terminal set (V)
% This set includes the input variables for the GEP model
x_var = {'x1','x2','x3','x4','x5','x6','x7','x8'};  % Terminal
variables vector
% Defines the constant terminal set (C)
% This set includes the constant values that can be used in the GEP
model
c_var = {'c1','c2','c3','c4','c5','c6','c7','c8'};

%In Genetic Programming (GP), the term "arity" refers to the number of
arguments
% or operands that a function or operator takes.
% max_arity: the maximum arity (number of arguments) of the functions
in the function set
max_arity=2;

%-----------Differential Evolution(DE) Parameters-------------------
-------------
% Maximum Number of Iterations for the DE algorithm
Max_iteration = 5000;
% Population Size for the DE algorithm
SearchAgents_no = 40;
% Lower Bound of the Scaling Factor in DE
beta_min = 0.2;
% Upper Bound of the Scaling Factor in DE
beta_max = 1.5;
% Crossover Probability for the DE algorithm
pCR = 0.2;
% Message flag for optimization process
% 0: stop messages
% 1: enables messages from the optimization process
MesFlag = 1;
% Enables to plot convergence characteristic (Cost-Iteration curve)
% 0: Does not draw Cost-Iteration plot
% 1: Draws Cost-Iteration plot
IsDraw = 1;

% Run the DEC_DE algorithm
DEC_GEP(ConsValLow,ConsValMax,max_arity,Max_iteration,SearchAgents_no,
beta_min,beta_max,pCR,MesFlag,IsDraw)

% Prints a separator line for better readability in the command window
fprintf('**********************************************************
******\n');
```

```
% Calculates the predicted data and evaluate performance of the
symbolic regression model
[MSE, MAE, RAE, gep_model_data] = gep_compare(BestModel, inputs,
outputs, best_constants);
% Prints the Mean Absolute Error (MAE)
fprintf('MAE: %d \n', MAE);
% Prints the Mean Squared Error (MSE)
fprintf('MSE: %d \n', MSE);
% Prints the Relative Absolute Error (RAE)
fprintf('RAE: %d \n', RAE);
% Prints the symbolic regression model
fprintf('Best Model:');
disp(BestModel);
```

To consider later, all workspace parameters are saved in "Workspace_Date_CurrentTime.mat" file.

You can freely use or modify these codes by citing articles [1]. (These codes and the algorithm are designed only for research purposes. Users accept responsibility for any problem that is associated with these codes and the algorithm.)
When you use these codes in your research work, you can cite the following three articles.

Citation: Ari, D., Alagoz, B. B. (2023). A differential evolutionary chromosomal gene expression programming technique for electronic nose applications. Applied Soft Computing, 136, 110093. https://doi.org/10.1016/j.asoc.2023.110093

Adding User-defined Function and Operators:

Construction of the function set (F) is a very important stage that is effective in symbolic regression performance of DEC-GEP. Two types of functions and operators can be used [1]:

1- Matlab Functions: Users can directly use Matlab functions in the function list by using the function name and arity (number of arguments (operands, input)). Format of the F set is the {'function name', number of arguments (operands);}

Example:

F = {'*', 2; % Multiplication operator with arity 2 in Matlab is added to the function set F.

'+', 2; % Addition operator with arity 2 in Matlab is added

'-', 2; % Subtraction operator with arity 2 in Matlab is added

'sin', 1}; % Sinus function with the arity 1 in Matlab is added

2- User-Defined Functions(Custom Functions): Users can define their custom functions and add them into `\functions` folder. Thus, the DEC-GEP algorithm can use user-defined functions in this folder when they are listed in the set F.

Definitions of some custom functions in Ref[1]:

1) User-defined robust division function:

$$div(x,y)=\begin{cases}\dfrac{x}{0.001+y} & if \quad y==0\\ \dfrac{x}{y} & else\end{cases}$$

Custom function definition:

```
function [result] = div(x, y)
    epsilon = 0.001;
    % Check if the denominator y is zero
    if y == 0
        % If y is zero, perform division by (y + epsilon) to
avoid division by zero errors
        result = x ./ (y + epsilon);
    else
        % If y is not zero, perform division
        result = x ./ y;
    end
end
```

2) User-defined real valued square function:

$$sqrt(x)=\sqrt{|x|}$$

Custom function definition:

```
function [result]=ReelLog(x)
     result=log(abs(x));
end
```

3) User-defined real valued log function:

$$\log(x)=\log(|x|)$$

Custom function definition:

```
function [result]=ReelSqrt(x)
```

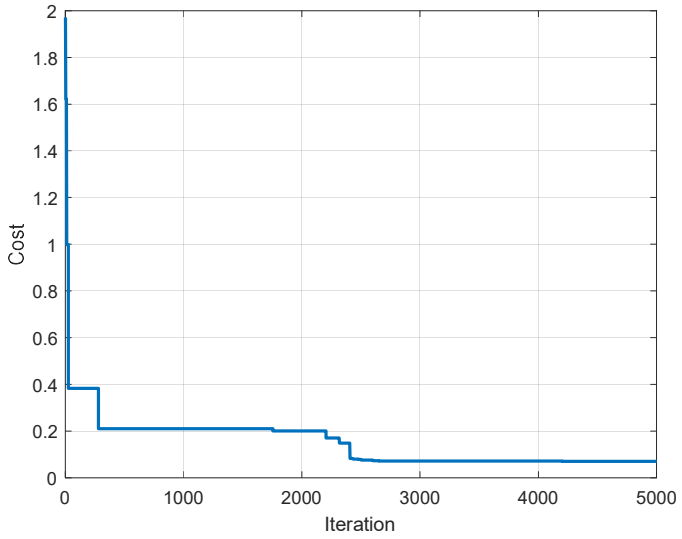
```
    result=sqrt(abs(x));
end
```

Example: To use user-defined functions and other matlab functions, an example function set F is defined below[1].

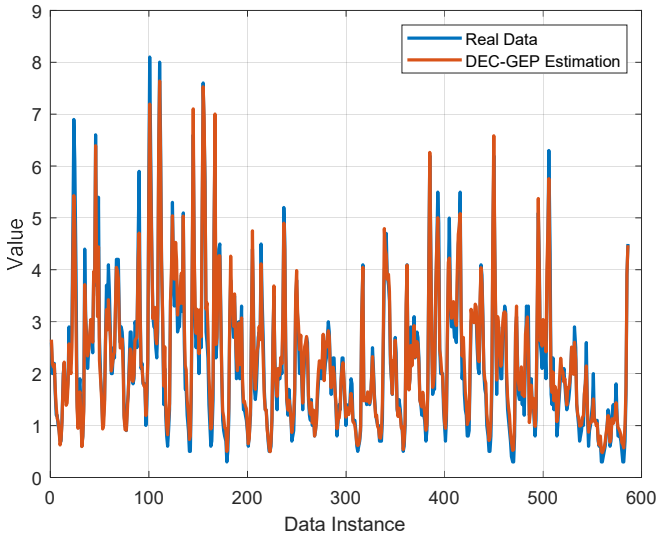
```
F={'div',2;'ReelSqrt',1;'ReelLog',1;'cos',1;'tan',1;'cot',1;'sin'
,1;'*',2;'+',2;'-',2};
```

Some Results for Running DEC-GEP tool:

The convergence characteristic (cost-iteration curve) shows improvement of MSE performance during DE optimization, below.



CO concentration data and estimates of DEC-GEP symbolic model are shown below.



Some performance data and symbolic regression model:

MAE: 1.967623e-01

MSE: 7.061852e-02

RAE: 1.137791e-01

Best Model: div((div((x2),((c7)+(div((x5),(c8))))))),(div((c4),(x2))))

c1=-1000.000000; c2=-1000.000000; c3=1000.000000; c4=582.877622; c5=-488.535587; c6=-1000.000000; c7=730.938119; c8=-53.561690;

This symbolic model can be interpreted as the mathematical model as follows:

$$y = \frac{\frac{x_2}{c_7 + \frac{x_5}{c_8}}}{\frac{c_4}{x_2}} \Rightarrow y = \frac{\frac{x_2}{730.93811 + \frac{x_5}{-53.561690}}}{\frac{582.877622}{x_2}}$$

Multi-sensor training dataset and parameters[1]

	Sensor Type	Input-Output Parameters	Explanation
Input	PT08.S1	x_1	Tin oxide gas sensor (CO sensitive)
	PT08.S2	x_2	Titania gas sensor (NMHC sensitive)
	PT08.S3	x_3	Tungsten oxide gas sensor (NOx sensitive)
	PT08.S4	x_4	Tungsten oxide gas sensor (NO_2 sensitive)
	PT08.S5	x_5	Indium oxide gas sensor (O_3 sensitive)
	Temperature	x_6	Temperature Measurement
	Relative Humidity	x_7	Relative Humidity Measurement
	Absolute Humidity	x_8	Absolute Humidity
Output	Reference Analyzer	y_r	Precise concentration measurements (ground truth)for CO concentration
	Output of the DEC-GEP based models	y	CO concentration estimates (mg/m^3) from the DEC-GEP based estimation model

* Dataset was collected and shared by S. De Vito et al. (S. De Vito, E. Massera, M. Piga, L. Martinotto, G. Di Francia, On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario, Sensors Actuators B Chem. 129 (2008) 750–757. https://doi.org/10.1016/j.snb.2007.09.060)