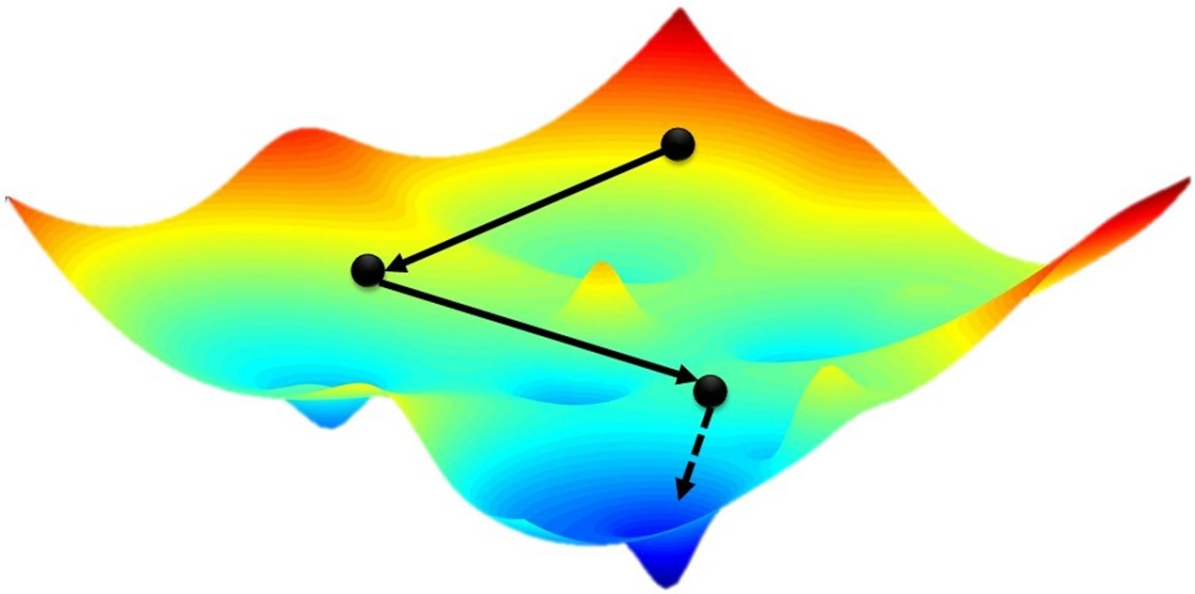
Center for Hydrometeorology and Remote Sensing,
Department of Civil and Environmental Engineering,
University of California, Irvine, CA, USA.

Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) Optimization Framework

User Guide



Author: Matin Rahnamay Naeini
Email: rahnamam@uci.edu

Disclaimer

The Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) optimization toolbox is provided 'as is' without any warranty of any kind, express or implied. While we strive to provide accurate codes for SC-SAHEL toolbox, we cannot guarantee the accuracy of the codes, figures and examples. The results generated by the SC-SAHEL framework can be used at your own discretion and risk, and with agreement that you will be solely responsible for errors or omissions of the SC-SAHEL codes, outputs, documents and figures. In no event shall the authors, developers or their affiliate institutions be liable to you or any third parties for any special, direct, indirect or consequential damages and financial risks of any kind, or any damages whatsoever, resulting from, arising out of or in connection with the use of the SC-SAHEL toolbox. The code is subject to change without notice.

To download the latest version of the code please visit:
http://chrs.web.uci.edu/resources.php or visit MathWorks website.

For any question or technical issue, please contact:
Matin Rahnamay Naeini
rahnamam@uci.edu

Please reference to:

Matin Rahnamay Naeini, Tiantian Yang, Mojtaba Sadegh, Amir AghaKouchak, Kuo-lin Hsu, Soroosh Sorooshian, Qingyun Duan, and Xiaohui Lei. Shuffled Complex-Self Adaptive Hybrid Evolution (SC-SAHEL) optimization framework. *Environmental Modelling & Software*, 104:215-235, 2018.

Contents

1 Introduction

The Shuffled Complex-Self Adaptive Hybrid Evolution (SC-SAHEL) optimization framework is developed based on the Shuffled Complex Evolution-Developed at University of Arizona (SCE-UA) algorithm. SC-SAHEL provides a hybrid optimization framework for solving single-objective optimization problems, which allows employing multiple Evolutionary Algorithms(EAs) as search mechanisms. In this framework, the most suitable EAs are selected and employed to evolve the population at each optimization step. The algorithm follows an "Award and Punishment" logic for selection EAs. Hence, the best performing algorithms contribute more to evolution of the population. This feature provide detailed information regarding the performance of the EAs during the optimization process. In this user guide, settings and features of the SC-SAHEL framework is detailed. Examples are provided to help user with preparing the algorithm.

2 SC-SAHEL settings

SC-SAHEL algorithm provide an arsenal of tools and feature for solving single objective optimization problems, and evaluating different EAs. The setting of the algorithm include *required* and *optional* settings. The *required* settings include range of the parameters, objective function, and maximum number of function evaluation. Providing these settings is necessary and user need to provide them with the proper format. Format of the *required* inputs are as follow:

Required inputs:

Range of parameters: The range of parameters is the first and second input for the SC-SAHEL algorithm. The range should be provided as two separate variable as *lb* and *ub*, which are the lower bound and upper bound of the parameters range, respectively.

Maximum number of function evaluation allowed: The maximum number of function evaluation is the third input for the SC-SAHEL toolbox.

Objective function: The objective function name, is the fourth input to the algorithm. Please note if the objective function requires any input, such as specific setting, input data, or any other data, these information should be provided in another variable, referred as *Data*. The *Data* variable will be discussed subsequently. It is necessary to use *varargin* for objective function. Figure 2.1 shows the details of objective function structure using *varargin*.Using the inputs described above, user can run the SC-SAHEL algorithm as shown in Figure 2.2. The *'main.m'* file is an example of main file for running SC-SAHEL algorithm.

```
function [ObjectiveFunctionValue] = objectivefunction(varargin)
params = varargin{1};              % Parameters
OtherData = varargin{2};           % Any other Data
```

Figure 2.1: Example of objective function structure.

```
lb = [-100,-100];                  % Parameters lower bounds
ub = [100,100];                    % Parameters upper bounds
MaxFcal = 100000;                  % Maximum number of function evaluation
ObjFun = 'F1';                     % Objective function name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Call SC-SAHEL algorithm
[X_optimum,F_optimum,misc] = SC_SAHEL(lb,ub,MaxFcal,ObjFun);
```

Figure 2.2: Setting up SC-SAHEL with only 4 inputs.

The SC-SAHEL algorithm will use default settings if only 4 inputs are provided. To

apply different settings for the SC-SAHEL framework, user can tune the optional settings by tuning the parameters. The optional settings follow specific format and should be provided with careful consideration. Providing wrong inputs may lead to errors and unexpected results. The optional settings are detailed in Table 2.1.

Table 2.1: SC-SAHEL setting, parameters and their default values.

Setting	Parameter	Default	Description
Number of Complexes	NumOfComplex	4	Specify the number of complexes used within the framework. ***Note the number of complexes should be proportional to the number of EAs.
Size of Complexes	ComplexSize	$2 \times d + 1$ (d is dimension of the problem)	Specify the number of points within each complex. ***Note the population is equal to $NumOfComplex \times ComplexSize$.
Number of evolution steps	EvolStep	$max(d + 1, 10)$ (d is dimension of the problem)	Specify the number of steps each complex will be evolved before shuffling.
Evolutionary algorithms	EAs	{'MCCE','CCE'} (Employed in SP-UCI and SCE-UA algorithms, respectively.)	Specify the evolutionary algorithms. ***Note the name of the evolutionary algorithms should be provided in a cell variable.
Initial sampling method	SampMethod	'LHsamp' (Latin Hypercube Sampling)	Specify the method for generating the initial population.
Boundary handling method	BoundHand	'ReflectBH' (Reflection boundary handling method)	Specify the algorithm for boundary handling.
Parallel computation	Parallel	false	Enable/Disable parallel computation.
Dimension restoration component	DimRest	true	Enable/Disable dimension restoration for finding and restoring the missing dimensions.
Gaussian resampling	ReSamp	false	Enable/Disable Gaussian resampling.
Stopping criteria	StopSP	10^{-7}	The minimum range of parameters that span the search space.
Stopping criteria	StopStep	50	Specify the number of steps to be considered for evaluating the objective function improvement.
Stopping criteria	StopIMP	0.1	The minimum percent improvement in objective function value in the last m steps. (m specified by StopStep)
Objective function data	Data	empty	Data required for calculating the objective function should be store in this variable.

The information provided in Table 2.1 should be used as reference for tuning the SC-SAHEL algorithm settings. Employing these settings requires a good understanding of

the problem and SC-SAHEL features. Available choices for some of the settings depend on the availability of the functions. The available choices for EAs, initial sampling and boundary handling are detailed below. Interested users are referred to the SC-SAHEL manuscript [5] for further details on the algorithms.

2.1 Evolutionary algorithms

The current version of the SC-SAHEL toolbox provides multiple choices for EAs. These choices are listed in Table 2.2. Any user defined evolutionary algorithm can be used within the SC-SAHEL framework. Figure 2.3 shows an example of the function structure for defining EA for SC-SAHEL framework.

Table 2.2: List of EAs.

EA name	Function Name	Description
Competitive Complex Evolution (CCE)	'CCE'	The evolutionary method used in the SCE-UA algorithm [2].
Modified Competitive Complex Evolution (MCCE)	'MCCE'	The evolutionary method used in SP-UCI algorithm [1].
Modified Differential Evolution	'DEF'	The modified version of Differential Evolution [6].
Modified Frog Leaping	'FL'	The modified version of Frog Leaping algorithm [3].
Modified Grey Wolf Optimizer	'GWO'	The modified version of Grey Wolf Optimizer [4].

2.2 Initial sampling

SC-SAHEL uses the Latin Hypercube sampling (LHS) as default method for random sampling however any user defined initial sampling method can be employed within the framework. Another initial sampling method which is provided within the SC-SAHEL framework is Uniform Random sampling (URsamp), which can be enabled by setting the 'SampMethod' parameter to 'URsamp'. Figure2.4 show the structure of the random sampling function for interested users.

2.3 Boundary handling

SC-SAHEL uses Reflection as default method for boundary handling. The algorithm also provide SetToBound method for handling the points which are out of the parameters feasible range. To enable SetToBound method, set 'BoundHand' to 'SetToBound'.

```
function [X,F,fcal] = FL(varargin)
% Outputs
% - X is the evolved complex
% - F is the evolved complex objective function values
% - fcal is the function evaluation counter
% Input variables
X = varargin{1};            % Samples within the complex
F = varargin{2};            % Samples objective function values
fobj = varargin{3};         % Objective function, function handle
EvolStep = varargin{4};     % Maximum number of evolution allowed
bh = varargin{5};           % Boundary handling structure
X_best = varargin{6};       % Best point achieved so far
F_best = varargin{7};       % Best objective function value achieved so far
Data = varargin{8};         % Required data for objective function
fcal = varargin{9};         % Function evaluation counter
```

Figure 2.3: Example of EAs structure.

```
function X = URsamp(N,lb,ub)
% Input
% N is the number of samples
% lb is the parameters lower bound
% ub is the parameters upper bound
% Output
% X is the sampled data
```

Figure 2.4: Example of initial sampling function structure.

Figure2.5 shows the structure of the boundary handling functions for interested users. Boundary handling settings are transferred as a structure variable within the SC-SAHEL

```
function [X] = ReflectBH(X,lb,ub)
% Input
% X is the sample
% lb is the parameter lower bound
% ub is the parameter upper bound
% Output
% X is the Adjusted sample
```

Figure 2.5: Example of boundary handling function structure.

framework. The boundary handling structure variable, which is referred as 'bh' store the range of parameters ('bh.lb','bh.ub') and the boundary handling function ('bh.fun'). This should be considered for defining new EAs for the SC-SAHEL framework.

3 SC-SAHEL output

The SC-SAHEL toolbox return three outputs including the optimum solution, the optimum objective function values, and an structure array called misc. The misc array store the information regarding the performance of the framework. This variable stores different information. The fields of the misc structure are listed in Table3.1.

Table 3.1: List of misc variables.

Field Name	Description
FunCal	Store the number of function calls at each shuffling step. It can be used for tracking the changes in the objective function values with the number of function calls.
BestF	Store the best objective function value achieved at the end of each shuffling step. It can be used for monitoring the changes in objective function values with the number of function evaluations.
BestX	Store the best points achieved at the end of each shuffling step.
EAselect	Store the selected EAs at each shuffling step. It can be used for evaluating the performance of each EA. This variable has m columns (m is the number of complexes), and each element of the matrix shows a number which is corresponding to the EA. For instance if user employ 'MCCE','DEF' as EAs, these EAs are shown as 1 and 2 in the EAselect matrix, respectively.

4 SC-SAHEL plotting

The SC-SAHEL plotting function provide 2 figures. First figure shows the changes in the objective function value vs. the number of function calls, and second figure shows the number of complexes assigned to EAs at each shuffling steps. Figure 2.5, and 4.1 show an example for these figures. To plot these information, user can call the function (Plot_SC_SAHEL) and pass the variable misc to the function (Plot_SC_SAHEL(misc)).

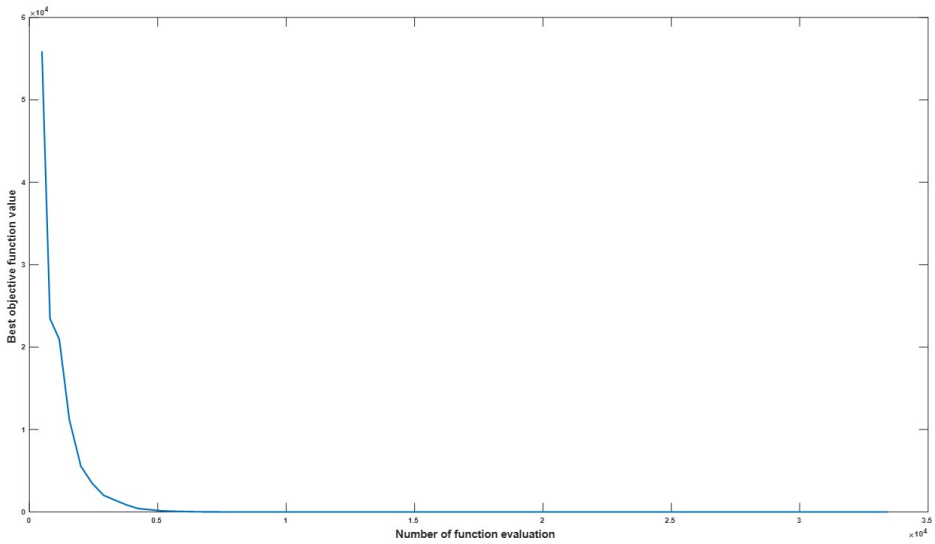


Figure 4.1: Example of Plot_SC_SAHEL figures

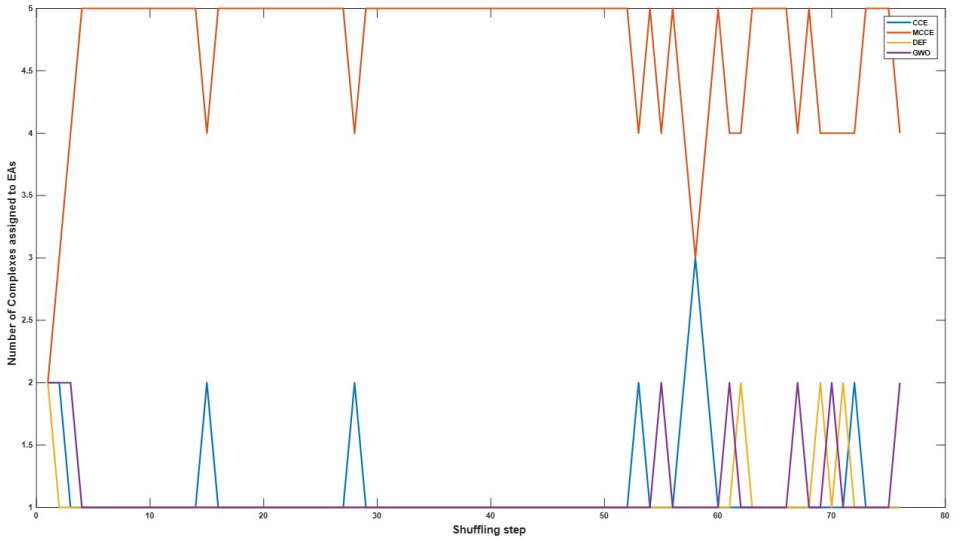


Figure 4.2: Example of Plot_SC_SAHEL figures

5 SC-SAHEL GUI

SC-SAHEL GUI can be executed by running command *'GUI_SC_SAHEL'* in MATLAB. Before executing this command make sure you are in the correct directory. Figure 5.1 shows the main window of the SC-SAHEL GUI. For more details on the technical description of the algorithm settings, please refer to [5], or section 2 of this document. The setting of the GUI is detailed subsequently.

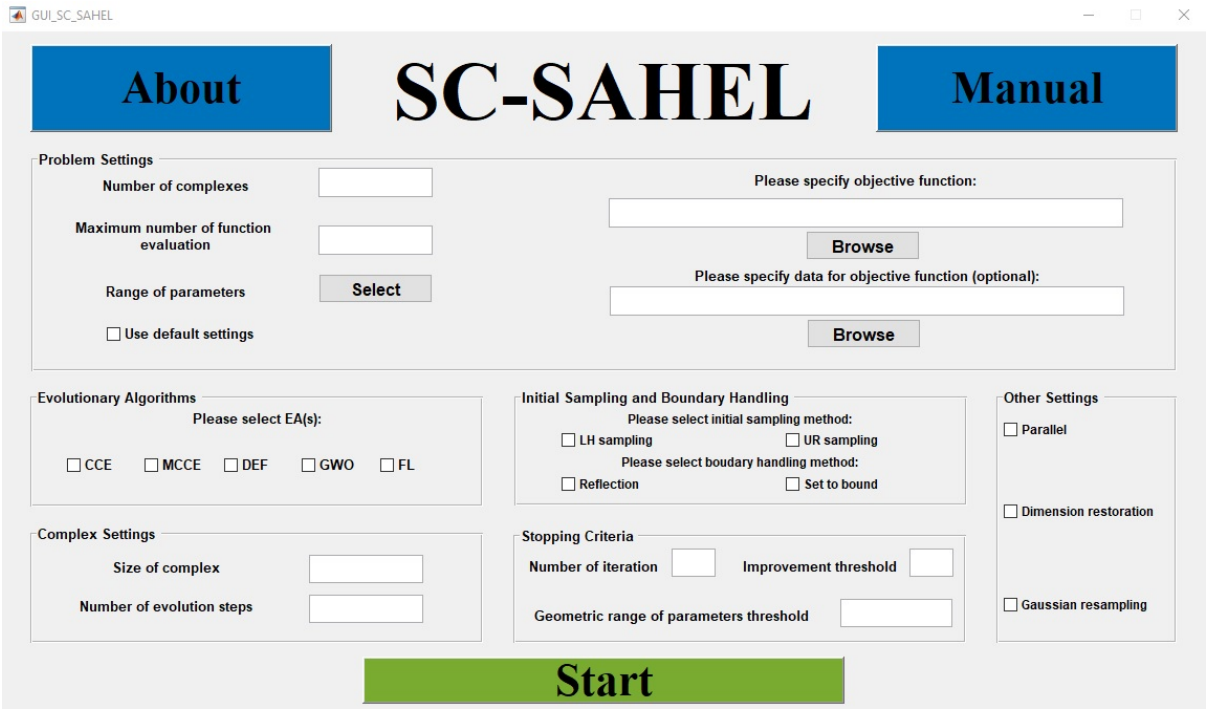


Figure 5.1: SC-SAHEL GUI.

The setting of the GUI is detailed here according to the box numbers in Figure 5.2. All the

settings in the panel **Problem Settings** is required for running SC-SAHEL algorithm. Details of the settings are as follow:

- **Box 1** is **Number of complexes**. This field has default value of 4, however, if user employs more than one EA, the number of complexes should be proportional to the number of EAs.
- **Box 2** is the **Maximum number of function evaluation** which is one of the stopping criteria for SC-SAHEL. This value should be specified according to the dimension and complexity of the problem.
- **Box 3** opens a new window for specifying the range of parameters. This settings is detailed subsequently.
- **Box 4** enables/disables default settings for the algorithm. If this box is checked, user cannot specify other settings. For details of the SC-SAHEL default settings please refer to Table 2.1.
- **Box 5** is the address for the objective function. The objective function should be in MATLAB language. If the objective function is in the same directory as the SC-SAHEL code, there is no need to browse to the location of file, and providing the name of function suffices.
- **Box 6** is the data file or additional settings required for calculating objective function value. SC-SAHEL only supports ***.mat** files in the current version.

The aforementioned settings are required settings for running SC-SAHEL algorithm.

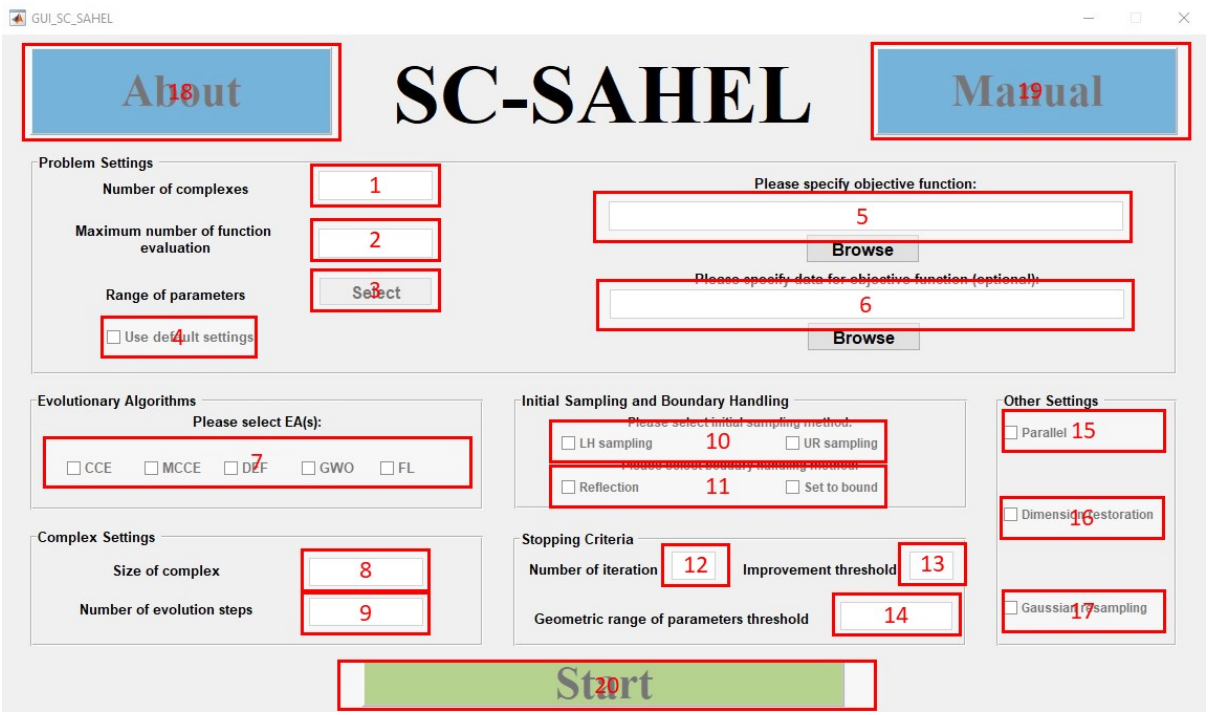


Figure 5.2: SC-SAHEL GUI with boxes for reference.

If user is interested in employing other settings rather than default settings, careful consideration should be devoted to the details and functionality of the settings. User can

leave any of the boxes 7-17 blank to use default settings. Details of these settings are as follow:

- **Box 7** specify the EAs. Multiple algorithm can be used within SC-SAHEL framework, however, number of complexes should be proportional to the number of EAs.
- **Box 8** specify the size of the complexes.
- **Box 9** define the number of steps each complex will be evolved before shuffling the complexes.
- **Box 10** tune the settings for initial sampling methods.
- **Box 11** tune the settings for boundary handling methods.
- **Box 12** is the number of steps which should be considered for measuring improvement in the objective function value.
- **Box 13** is the threshold for measuring the improvement in selected previous steps.
- **Box 14** is the minimum range of parameters which span the search space.
- **Box 15** enables/disables the parallel computation.
- **Box 16** enables/disables dimension restoration component.
- **Box 17** enables/disables Gaussian resampling feature.
- **Box 18** shows information about the current version of the algorithm.
- **Box 19** opens the SC-SAHEL manual PDF file.
- **Box 20** starts the SC-SAHEL optimization process.

Figure 5.3 shows the window for selecting the range of parameters. First user need to specify the dimension of the problem in the top right field. User can also browse and load the ***.mat** file containing the parameters range. If all the parameters have same range, user can enable the setting **Parameters have same range** and specify one lower and upper bound for the parameters in the table. After specifying the range of parameters, the code will save the range of parameters in a **par_range.mat** by pressing the save button. If this file exist in the code directory, there is not need to specify the parameters range again.

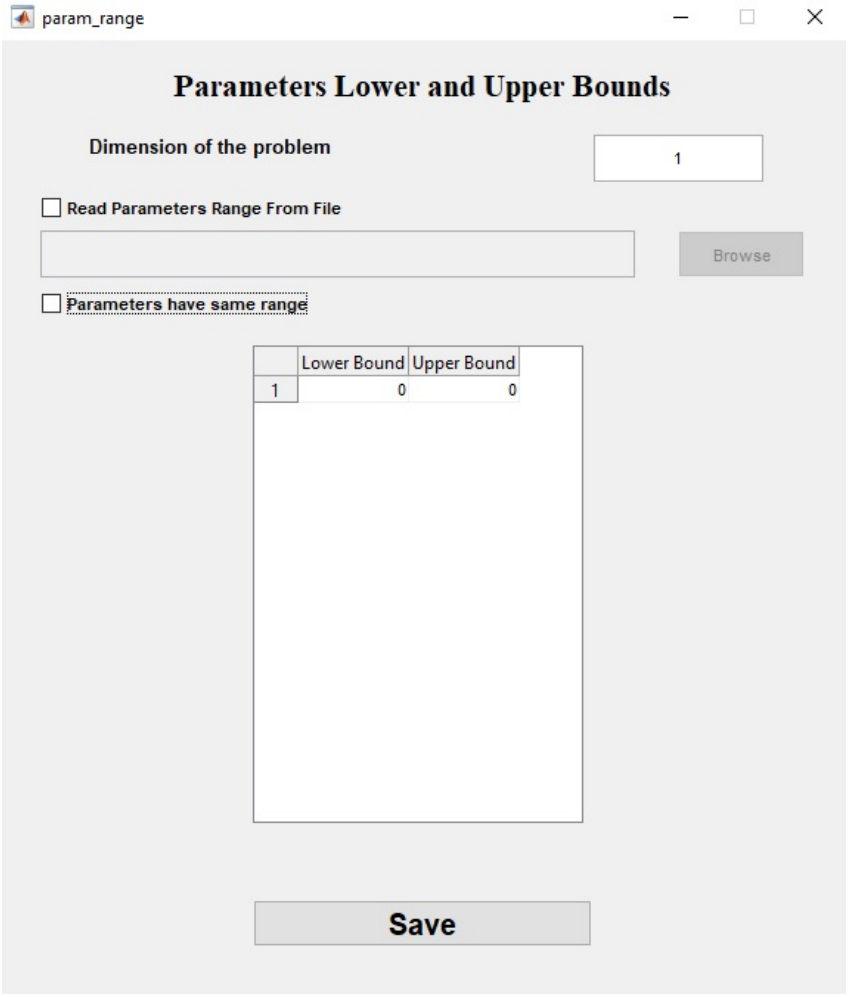


Figure 5.3: Range of parameters window.

6 Examples

6.1 Using script

Here is an example for setting up the SC-SAHEL without the GUI. For instance, the objective function is 'F3', maximum number of function evaluation is 100000, and the range of parameters are [-600,600] with 5 dimensions. Figure 4.2 shows this example with default settings. However, if user is interested in using other settings rather than the default settings, the script will change as follow. For instance, user is interested in using DEF as the evolutionary algorithm, and would like to use uniform random sampling. Figure 6.1 shows an example for this case.

```
addpath('TestCases\');
lb = [-100,-100,-100,-100,-100];    % Parameters lower bounds
ub = [100,100,100,100,100];         % Parameters upper bounds
MaxFcal = 100000;                   % Maximum number of function evaluation
ObjFun = 'F3';                      % Objective function name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Call SC-SAHEL algorithm
[X_optimum,F_optimum,misc] = SC_SAHEL(lb,ub,MaxFcal,ObjFun);
% Plot the results
Plot_SC_SAHEL(misc)
```

Figure 6.1: Example of SC-SAHEL setup with default settings.

```
addpath('TestCases\');
lb = [-100,-100,-100,-100,-100];    % Parameters lower bounds
ub = [100,100,100,100,100];         % Parameters upper bounds
MaxFcal = 100000;                   % Maximum number of function evaluation
ObjFun = 'F3';                      % Objective function name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Call SC-SAHEL algorithm
[X_optimum,F_optimum,misc] = SC_SAHEL(lb,ub,MaxFcal,ObjFun,...
    'EAs',{'DEF'},'SampMethod','URsamp');
% Plot the results
Plot_SC_SAHEL(misc)
```

Figure 6.2: Example of SC-SAHEL setup without default settings for EAs and initial sampling.

6.2 Using GUI

First the GUI is executed by *'GUI_SC_SAHEL'* command in MATLAB. Figure 6.3 shows the SC-SAHEL window with the setting for function 'F1'. In this example, user employ 8 complexes with 100000 as maximum number of function evaluation. User employ 4 different EAs for evolving the complexes and use Set to Bound as boundary handling method. Dimension restoration is also enabled in this example. Figure 6.4 shows the parameter range panel. The problem has 30 dimensions and the range of the parameters are all the same.

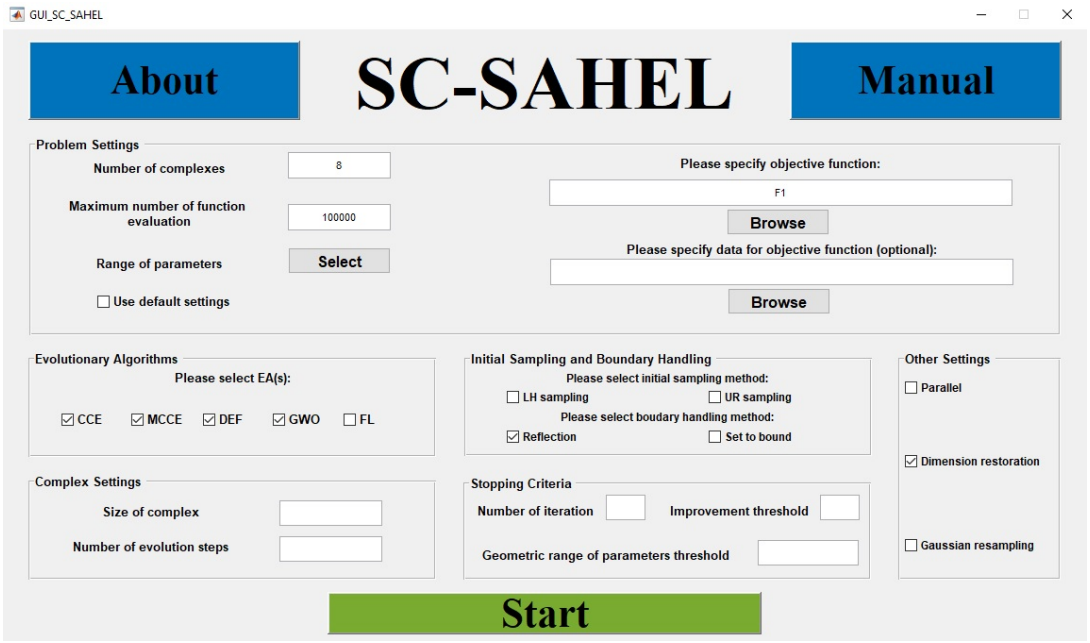


Figure 6.3: SC-SAHEL GUI settings for example F1.

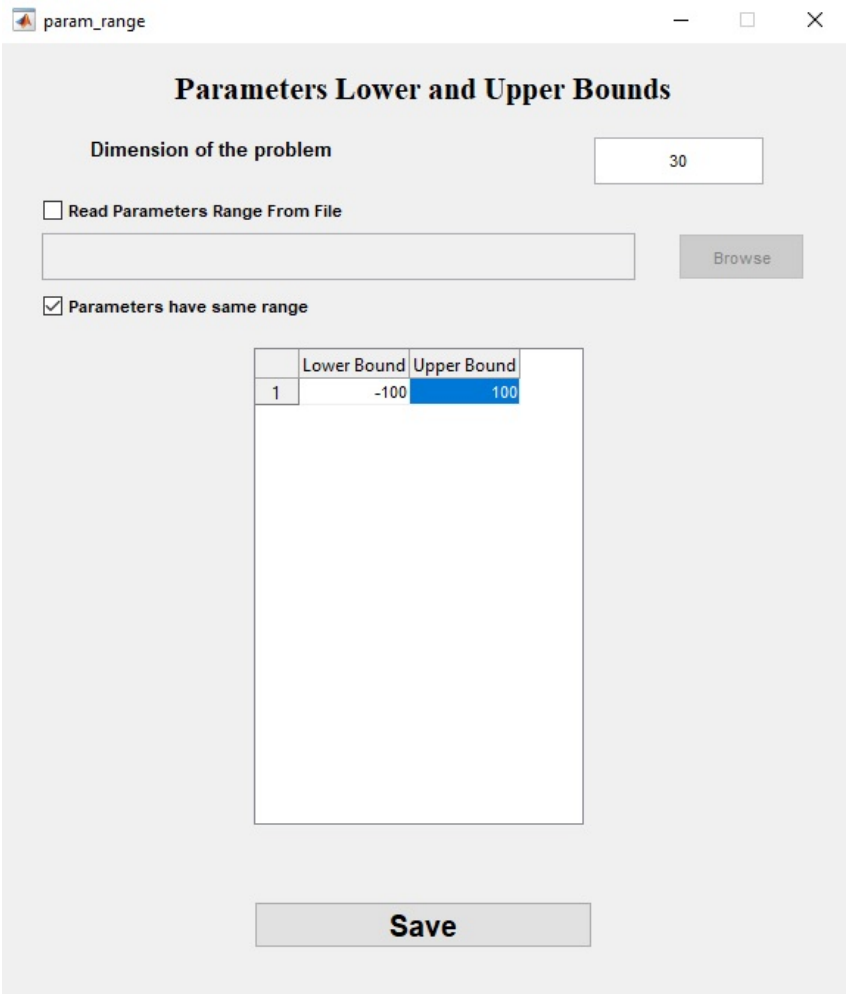


Figure 6.4: Defining range of parameters for example F1.

7 Test cases settings

A number of test cases are provided for exploring the SC-SAHEL algorithm. These test cases are mathematical functions which are used for evaluating optimization algorithms. Table 7.1 shows name, dimension, range of parameters, and optimum function value for these test cases for interested users. For further details on these test functions please refer to [4, 5, 7]

Table 7.1: List of test cases settings [4, 5, 7].

Function	Name	Dimension	Range	Optimum value
F1	Sphere Model	Any	[-100,100]	0
F2	Schwefel's Problem 2.22	Any	[-10,10]	0
F3	Schwefel's Problem 1.2	Any	[-100,100]	0
F4	Schwefel's Problem 2.21	Any	[-100,100]	0
F5	Generalized Rosenbrock's Function	Any	[-30,30]	0
F6	Step Function	Any	[-100,100]	0
F7	Quartic Function	Any	[-1.28,1.28]	0
F8	Generalized Schwefel's Problem 2.26	Any	[-500,500]	-12569.5
F9	Generalized Rastrigin's Function	Any	[-5.12,5.12]	0
F10	Ackley's Function	Any	[-32,32]	0
F11	Generalized Griewank Function	Any	[-600,600]	0
F12	Generalized Penalized Functions	Any	[-50,50]	0
F13	Generalized Penalized Functions	Any	[-50,50]	0
F14	Shekel's Foxholes Function	2	[-65.536,65.536]	1
F15	Kowalik's Function	4	[-5,5]	0.0003075
F16	Six-Hump Camel-Back Function	2	[-5,5]	-1.031628
F17	Branin Function	2	x_1 = [-5,10] x_2 = [0,15]	0.398
F18	Goldstein-Price Function	2	[-2,2]	3
F19	Hartman's Family	4	[0,1]	-3.86
F20	Hartman's Family	6	[0,1]	-3.32
F21	Shekel's Family	4	[0,10]	-10.1532
F22	Shekel's Family	4	[0,10]	-10.4028
F23	Shekel's Family	4	[0,10]	-10.5363

References

[1] Wei Chu, Xiaogang Gao, and Soroosh Sorooshian. A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences*, 181(22):4909–4927, 2011.

[2] Qingyun Duan, Soroosh Sorooshian, and Vijai Gupta. Effective and efficient global optimization for conceptual rainfall-runoff models. *Water resources research*, 28(4):1015–1031, 1992.

[3] Muzaffar Eusuff, Kevin Lansey, and Fayzul Pasha. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering optimization*, 38(2):129–154, 2006.

[4] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[5] Matin Rahnamay Naeini, Tiantian Yang, Mojtaba Sadegh, Amir AghaKouchak, Kuo lin Hsu, Soroosh Sorooshian, Qingyun Duan, and Xiaohui Lei. Shuffled complex-self adaptive hybrid evolution (sc-sahel) optimization framework. *Environmental Modelling & Software*, 104:215 – 235, 2018.

[6] A Kai Qin and Ponnuthurai N Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE, 2005.

[7] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102, 1999.