SPOTGUI Manual

Version 1.3

This *pdf*–Document is a manual for SPOT GUI , which is a graphical user interface for the Software SPOT.

Authors: Martin Zaefferer, Tobias Zimmer

Last updated: March 4, 2012

Contents

1 Basics

1.1 SPOT

SPOT is short for Sequential Parameter Optimization Toolbox. It is a tool typically used for problems like optimizing strategy parameters of algorithms in the field of computational intelligence.

SPOT is available both as a Matlab toolbox, and as a package for R (language and environment for statistical computing). This manual is meant as a help document for the SPOT-GUI, which is based on the R-Distribution of SPOT. To test it, you have to install R and the SPOT package first.[1]

If you want further information on what SPOT is and how to work with it, refer to Bartz-Beielstein et. al. [2] [1]. There are also several help files included in the R-Package of SPOT. They can be accessed by typing:
> vignette("SPOT")

Viewing a more detailed manual for the R-Package is done by typing:
> vignette("SPOTmanual")

There is also a new vignette on multi critieria optimization (MCO) with SPOT.
> vignette("MultiCriteriaSPOT")

MCO however is not yet included in the GUI described here.

1.2 SPOT and R

To use the GUI it is mandatory to have both R and SPOT installed. To install R on Linux you could do:

sudo apt-get install r-base

For windows (and also for further information on R) you can check the R-Projects homepage http://cran.r-project.org/. Once R is installed on your system, you need to start it and type:

install.packages("SPOT")

This will download and install the latest version of SPOT available on CRAN (A platform

[1]See http://www.r-project.org/ for R, and http://cran.r-project.org/web/packages/SPOT/index.html for the SPOT package.

for distribution of R-packages), and will also install some other packages that are needed for SPOT.

After SPOT is installed the GUI can be started.

require("SPOT") spotGui()

Please note that Java should also be installed, since the GUI is based on it. If you opened this help file from within the GUI, you will not have to worry about this of course.

1.3 Working with the GUI

As a first step when the spot GUI is started on windows (this is not an issue on linux systems) the user needs to set the path to the R-installation (or more precisely the R.exe). For this purpose the “Path setup” Button in the File menu is used, which starts a dialog where the user can chose the right path. (That path is usually something like C:/program files/R/R-2.10.1/bin. If you can not find it, you can check the R link on your desktop, and right click it to see where it links to. If you cant find that either, make sure if R is actually installed.)

The GUI tries to save the path information in a .cfg file in the same folder as where the SPOTGUI.jar file is stored. If this is not possible the path will have to be reset each time the GUI is started.

Once this is done the GUI is used to create the files that are used as a basis for each SPOT experiment. These settings will be explained in the following chapters. The rest of the GUI should be self-explanatory. If any questions remain check the following example.

1.3.1 Vital configuration parameters of SPOT and their representation in the GUI

The GUI tries to provide an easy way of using SPOT for users who are not used to work with R or similar environments. Still, users should be aware of the fact that not all features supported by the SPOT R-Package are available for easy use with the GUI, or at least not directly. This will be explained in the following chapter 1.3.3.

In this chapter however, only the most vital parameters will be explained.

The red numbers indicate the representation in the GUI as shown in figures 1.1 and 1.2.

alg.path 1 Only needed if target algorithm or function is loaded from a R-script. Defines the path to the location of said script. This value is actually not needed by SPOT anymore, instead the GUI needs this information to load the target function before running SPOT.

alg.func 2 Function name of the target algorithm or function.

io.apdFileName 3 Name and path of the .apd -file (Algorithm Problem Definition file, holding all specification the user written algorithm needs to perform a complete optimization). Not needed unless required by a custom user function.

report.func 4 The report function creates textual and graphical output by evaluating the results of all other steps done before. It can be called individually with the "rep" step, and is included in each "auto" task.

init.design.func 5 Name of the function that creates the initial design of the experiment. In the GUI only functions already implemented in SPOT can be used.

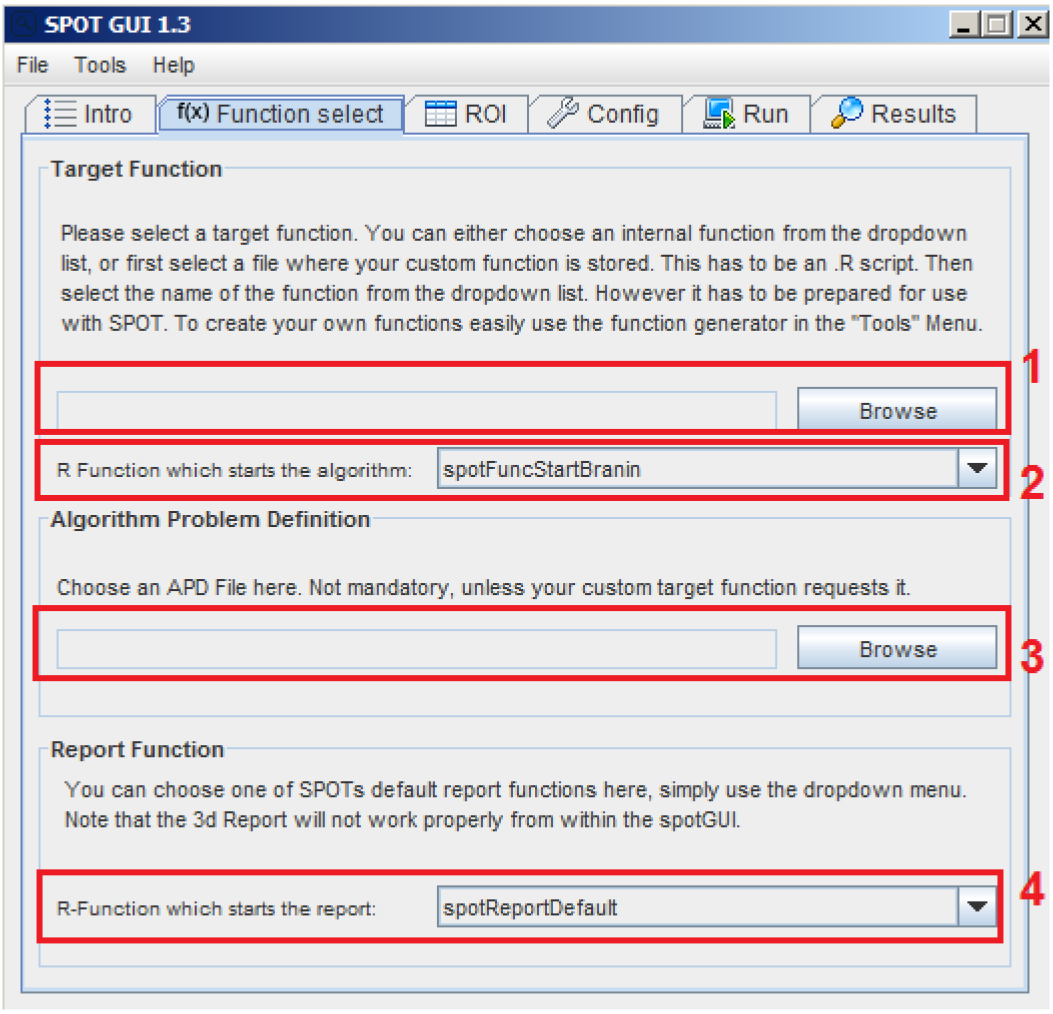
init.design.size 6 This is the number of initial design points to be created. This is required by some space filling design generators, and will be used in the *init.design.func* A value will be chosen by the *init.design.func* if *init.design.size* is set to "NA".

seq.predictionModel.func 7 Name of the function calling a predictor. In the GUI only functions already implemented in SPOT can be used.

auto.loop.steps 8 This is the number of looped steps that SPOT is allowed to perform in one "auto" task.

auto.loop.nevals 9 This is the number of target function evaluations that SPOT is allowed to perform in one "auto" task.

seq.design.size 10 Number of sequential design points to be created. The predictor will evaluate all these points, and the best points will be reevaluated by the target function/algorithm.



Target Function

Please select a target function. You can either choose an internal function from the dropdown list, or first select a file where your custom function is stored. This has to be an .R script. Then select the name of the function from the dropdown list. However it has to be prepared for use with SPOT. To create your own functions easily use the function generator in the "Tools" Menu.

R Function which starts the algorithm: spotFuncStartBranin

Algorithm Problem Definition

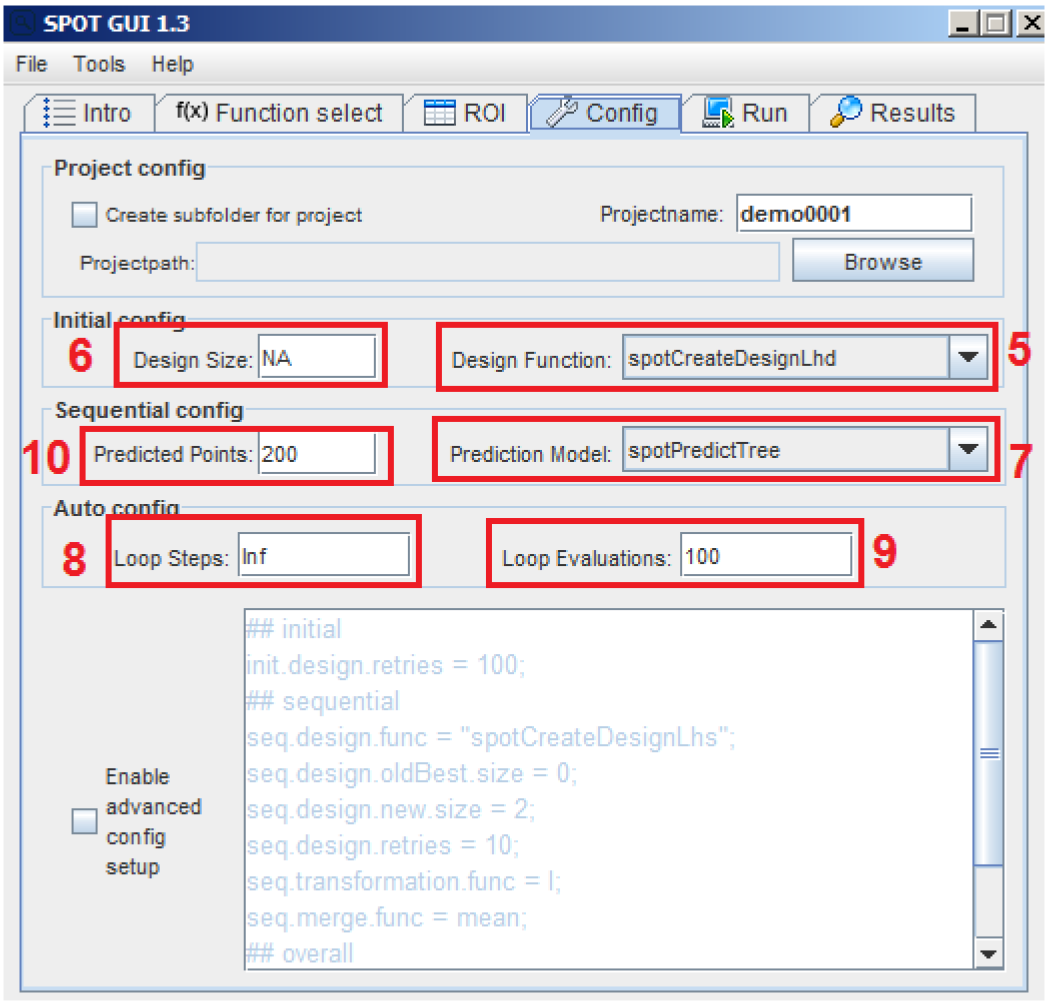
Choose an APD File here. Not mandatory, unless your custom target function requests it.

Report Function

You can choose one of SPOTs default report functions here, simply use the dropdown menu. Note that the 3d Report will not work properly from within the spotGUI.

R-Function which starts the report: spotReportDefault

Figure 1.1: Selection of functions to be used by SPOT



Project config

Create subfolder for project   Projectname: demo0001

Projectpath:

Initial config

6 Design Size: NA   Design Function: spotCreateDesignLhd 5

Sequential config

10 Predicted Points: 200   Prediction Model: spotPredictTree 7

Auto config

8 Loop Steps: Inf   Loop Evaluations: 100 9

Enable advanced config setup

```
## initial
init.design.retries = 100;
## sequential
seq.design.func = "spotCreateDesignLhs";
seq.design.oldBest.size = 0;
seq.design.new.size = 2;
seq.design.retries = 10;
seq.transformation.func = I;
seq.merge.func = mean;
## overall
```

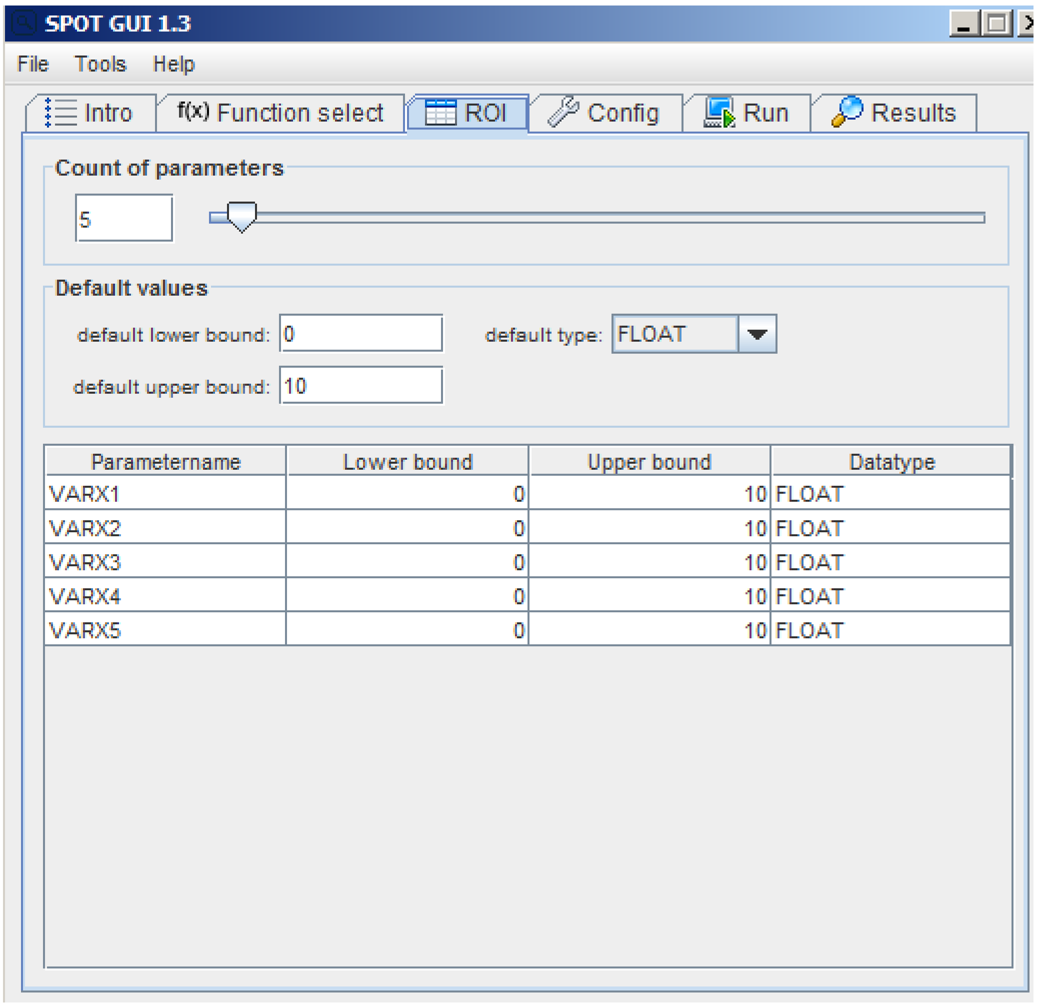
Figure 1.2: Configuration of parameters for SPOT

Please note that using the "Import Configuration" button in the file menu will not only influence the config tab, it will also try to set the paths for the function select tab.

1.3.2 The region of interest

Besides the above mentioned there are settings that describe how to handle the parameters to be optimized. This means giving them a name, giving them a data type, and defining a range in which they will be examined by SPOT.

This is done in a .roi file. The SPOTGUI prepares this with a table in the ROI tab.



Count of parameters: 5

Default values

default lower bound: 0

default upper bound: 10

default type: FLOAT

Parametername	Lower bound	Upper bound	Datatype
VARX1	0	10	FLOAT
VARX2	0	10	FLOAT
VARX3	0	10	FLOAT
VARX4	0	10	FLOAT
VARX5	0	10	FLOAT

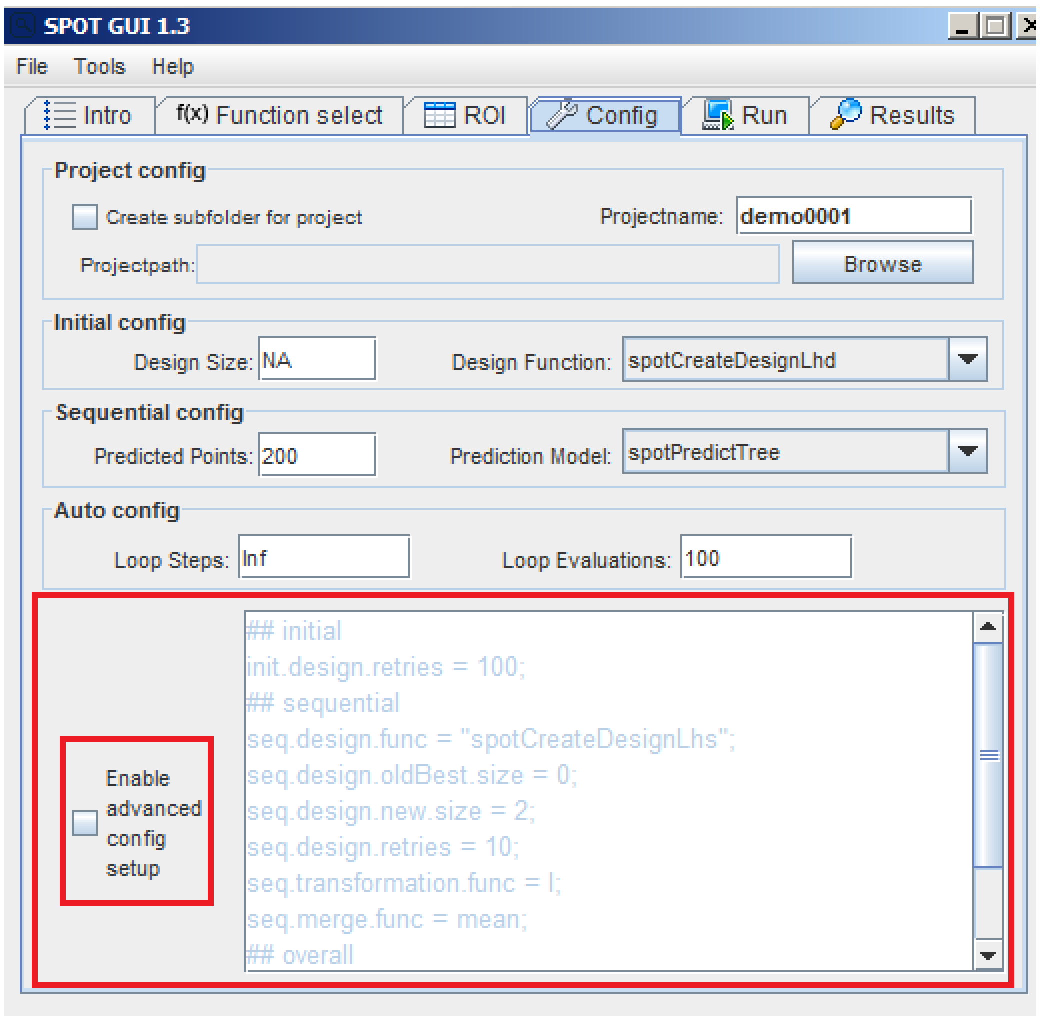
Figure 1.3: Configuration of the Region of Interest

If you open that tab you notice a field with a table in the lower part. This table is what matters. Above that there are some “default” values which are only to be used if you want to change all of the corresponding values in the table. And of course above that you can chose the number of parameters to be considered by SPOT, which defines the size of the table.

As with the configuration settings, an existing ROI can be loaded by hitting the corresponding import button in the File menu.

1.3.3 Using features or changing settings not implemented in the gui itself

In addition to the settings described in chapter 1.3.1, there are a large number of other settings used by SPOT. If an advanced user intends to work with those, instead of using the default values, the GUI offers the option to change, add or delete them by changing the “advanced config setup”. This needs first to be enabled or else, regardless of which values are in there, SPOT will only use defaults.



Project config

Create subfolder for project [ ]

Projectname: demo0001

Projectpath:

Initial config

Design Size: NA

Design Function: spotCreateDesignLhd

Sequential config

Predicted Points: 200

Prediction Model: spotPredictTree

Auto config

Loop Steps: Inf

Loop Evaluations: 100

Enable advanced config setup [ ]

```
## initial
init.design.retries = 100;
## sequential
seq.design.func = "spotCreateDesignLhs";
seq.design.oldBest.size = 0;
seq.design.new.size = 2;
seq.design.retries = 10;
seq.transformation.func = I;
seq.merge.func = mean;
## overall
```

Figure 1.4: How to set additional settings for SPOT

Once this is enabled, the user will already have a number of entries in a list of settings. The user could either change those, or delete them from the list. If something is not on the list, SPOT will instead use a default value. (However most of the entries on that list are already set to a default value to give an impression about what is reasonable)

Also note that if you load existing configuration files, the additional parameters will automatically be added to the list, and it will also automatically be enabled.

2 SPOTGUI: Simple Example

This chapter demonstrates how to use the SPOTGUI for solving a simple optimization Problems

This basic example will deal with the minimization of the Branin function. Note that the example presented here is very similar to one of the demos implemented in the SPOT-Package (demo01TreeBranin).

2.1 Target Function

Our aim is to find one of the three minima of the Branin function:

$$f_T(\vec{x}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$$

With the following constraints:

$$-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$$

The above mentioned minima are at:

$$f_T(\vec{x}^*) = 0.397887346$$
$${x_1}^* = -3.141592, {x_2}^* = 12.274999$$
$${x_1}^* = 3.141592, {x_2}^* = 2.275000$$
$${x_1}^* = 9.424777, {x_2}^* = 2.474999$$

Since the Branin function is both multi modal and non-separable it makes for an interesting choice as test function for SPOT. It is still easy enough to solve it in short time.

2.2 Setting up the experiment in the SPOTGUI

2.2.1 Selecting files

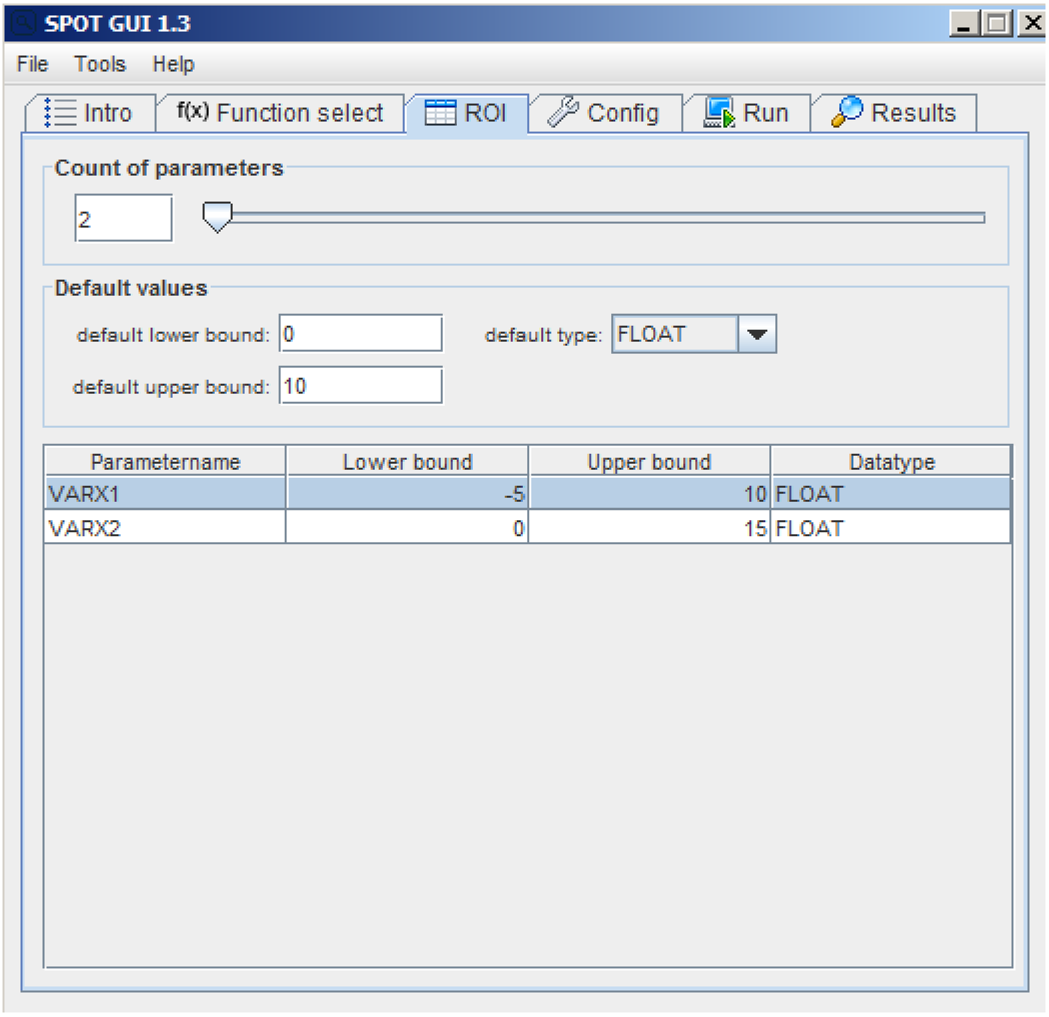
At this point the SPOTGUI should be started, and you should open the "function select"tab.

Actually there are not many changes to be made to the configuration in this tab. The target function we want and a report function are already chosen, as they are default values. If the Branin function was an external function, the Browse button would have to be used to search it. An .apd file has not to be chosen, either, as we do not want to specify any problem specific settings.

2.2.2 Region of Interest

Now the next tab can be opened, which deals with the region of interest. The region of interest is the range in which the parameters of the Branin function will be optimized. For this example the constraints as shown in chapter 2.1 will be used as a ROI. Also we need to chose the right names for the parameters, because they will be passed with that name to the target function. The correct settings are shown in figure 2.1.

Note that the default values are only of interest if you want to set values for a whole number of parameters instead of editing them one by one. Also instead of setting up the roi you could use an already existing roi file and load it with the "Import Region of Interest"button in the File menu.



Count of parameters: 2

Default values

default lower bound: 0

default type: FLOAT

default upper bound: 10

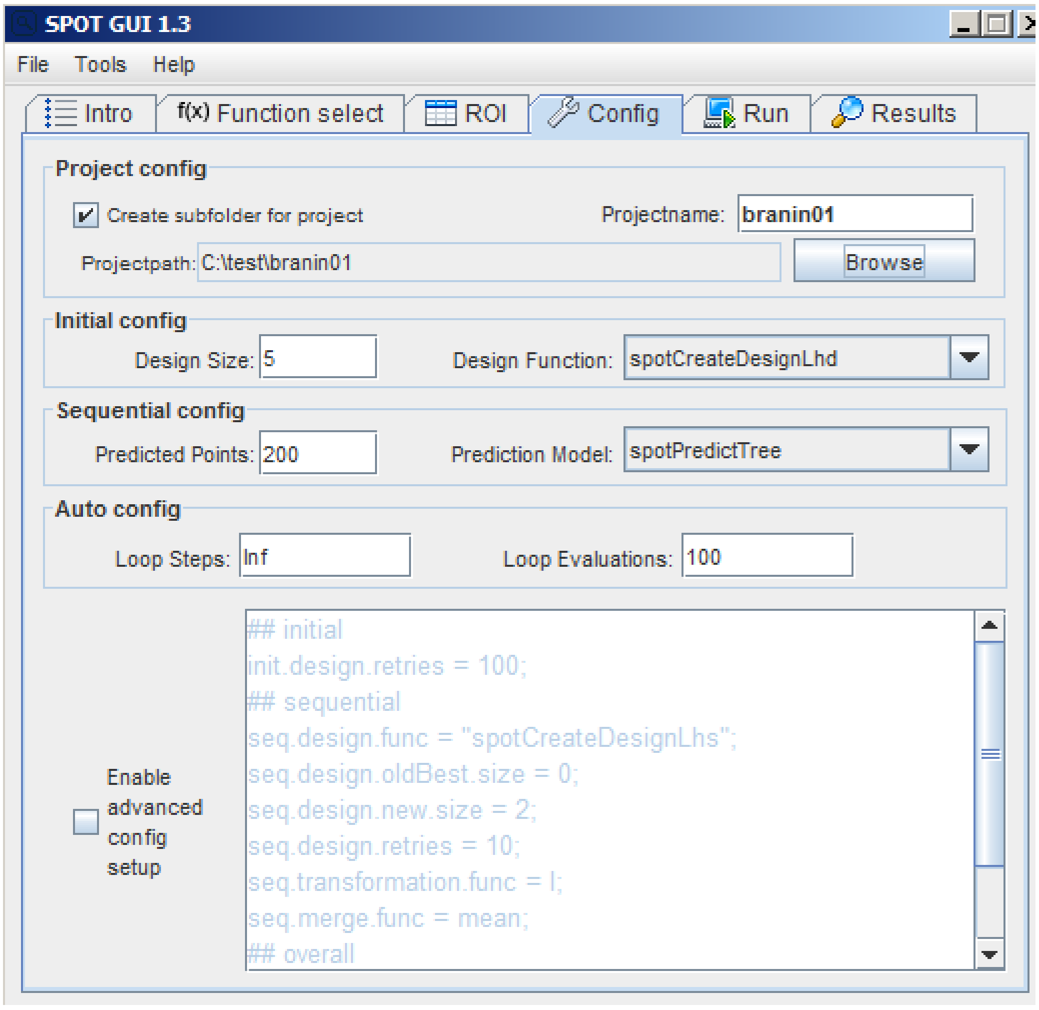
Parametername	Lower bound	Upper bound	Datatype
VARX1	-5	10	FLOAT
VARX2	0	15	FLOAT

Figure 2.1: Configuration of the Region of Interest for the Branin example

2.2.3 Configuring Experiment Settings

The settings to be chosen for this example are shown in figure 2.2. The model chosen for optimization is a tree based approach, the design function is based on the latin hypercube design. Loop Steps is the number of steps before SPOT stops, and is set to infinite in this case.

Loop Evaluations is the number of evaluations of the target function, and is in this case the only stopping criterion since Loop Steps are set to Inf. The Predicted Points parameter defines how many parameter settings will be tested with the tree based model, before (based on the prediction) the best of these will be tested with the target function i.e. the Branin function.



Project config

[✓] Create subfolder for project

Projectname: branin01

Projectpath: C:\test\branin01

Initial config

Design Size: 5

Design Function: spotCreateDesignLhd

Sequential config

Predicted Points: 200

Prediction Model: spotPredictTree

Auto config

Loop Steps: Inf

Loop Evaluations: 100

[ ] Enable advanced config setup

```
## initial
init.design.retries = 100;
## sequential
seq.design.func = "spotCreateDesignLhs";
seq.design.oldBest.size = 0;
seq.design.new.size = 2;
seq.design.retries = 10;
seq.transformation.func = I;
seq.merge.func = mean;
## overall
```

Figure 2.2: Settings to be chosen for the Branin example

Like in the conf tab there is again the option to load an existing .conf file from the File menu, which a user could also try to get a feeling for reasonable configurations.

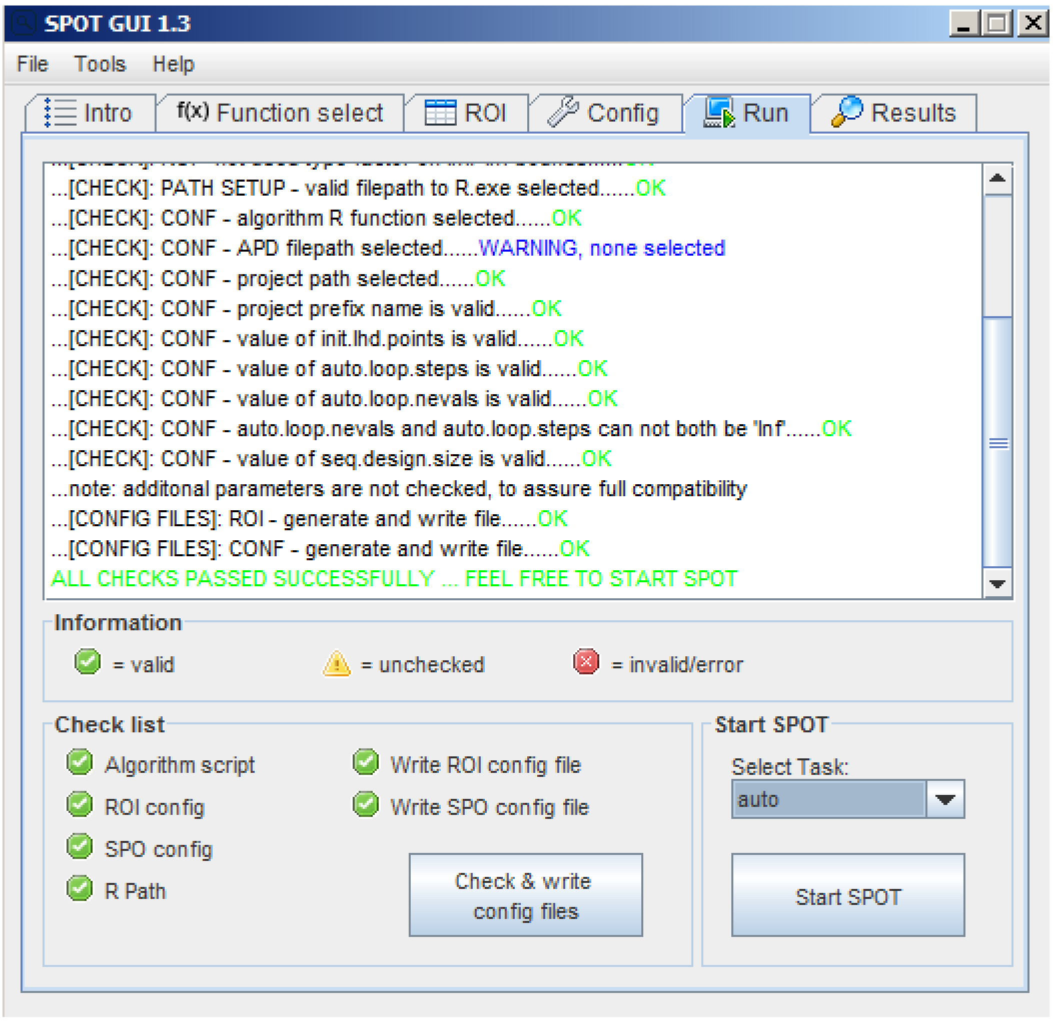
Do not forget to set the project path however, as seen in the upper part of this tab.

2.2.4 Starting the example

Now we can finally start the run tab. On this tab you must first press the “Check and write config files” Button, as this will create the files needed by spot from the information you gave to the gui. This will first check if all your input is valid and existent, and notify you if not. This will be shown as text in the upper part of the window, and also the icons near the button will tell you which parts of the check procedure were successful.

If all your settings are good, you should get the results as shown in figure 2.3. If this is the case, you should now be able to chose a task. This tells spot what you want to do. You have basically two options: Either start each task manually (i.e.: First init, then run, seq, run, seq, run, ..., and finally report). Or you can use the auto task to let SPOT handle your experiment. As shown in the figure "auto" will be used for this example. Once an appropriate task is chosen the experiment can be started.

Once the experiment is running, the GUI will wait for the SPOT task to finish, before you can interact with it again. Once it is finished a logfile will be shown in the run tab, showing you the console output that SPOT produced. But you can also see that logfile together with the other results in the result tab.



```
...[CHECK]: PATH SETUP - valid filepath to R.exe selected......OK
...[CHECK]: CONF - algorithm R function selected......OK
...[CHECK]: CONF - APD filepath selected......WARNING, none selected
...[CHECK]: CONF - project path selected......OK
...[CHECK]: CONF - project prefix name is valid......OK
...[CHECK]: CONF - value of init.lhd.points is valid......OK
...[CHECK]: CONF - value of auto.loop.steps is valid......OK
...[CHECK]: CONF - value of auto.loop.nevals is valid......OK
...[CHECK]: CONF - auto.loop.nevals and auto.loop.steps can not both be 'Inf'......OK
...[CHECK]: CONF - value of seq.design.size is valid......OK
...note: additonal parameters are not checked, to assure full compatibility
...[CONFIG FILES]: ROI - generate and write file......OK
...[CONFIG FILES]: CONF - generate and write file......OK
ALL CHECKS PASSED SUCCESSFULLY ... FEEL FREE TO START SPOT
```

✓ = valid ⚠ = unchecked ✗ = invalid/error

Figure 2.3: Checking and running the Branin example with the SPOTGUI

2.2.5 Viewing results

To view any produced results, open the Results tab. This tab is pretty much straightforward. All you need to do is chose a file, and hit the button to view it. Textfiles will be opened inside the GUI, PDF files will be opened with the standard PDF viewer installed on your operating system.

This is just meant to easily access the different files, and view them after running an experiment, they can as well be opened with any other text editor or PDF reader.

Bibliography

[1] Bartz-Beielstein, T., Lasarczyk, C., Preuß, M.: Sequential parameter optimization. In: McKay, B., et al. (eds.) Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland. vol. 1, pp. 773–780. IEEE Press, Piscataway NJ (2005)

[2] Bartz-Beielstein, T., Parsopoulos, K.E., Vrahatis, M.N.: Design and analysis of optimization algorithms using computational statistics. Applied Numerical Analysis & Computational Mathematics (ANACM) 1(2), 413–433 (2004)