How to use the TSC2 code

Catalin Stoean

October 12, 2011

1 What does the program do?

TSC2 is an evolutionary technique for multimodal function optimization. It is designed for maximization, but could be easily transformed for minimization. It finds a collection of the most important optima for a given function. For more information about TSC2, please see [1].

2 How to run the program

The main file can be found in the package "multiNationSpecies" and is called "Main.java". It uses another class called "Individual.java" that refers to the individuals in the EA. Also the Main file has a "parametersfile.txt" input text document. For a very simple run, one should set the parameter values in *parametersfile.txt*, in there put $useLHDs = false$ (will get back to this later), and put the arguments to the $Main$ class as the path to the *parametersfile.txt* document and $runs = 10$ for 10 repeated runs of the current configuration. Surely, one can put whichever value for the $runs$ argument. In such a case, the results are printed only at the console and the amount of output data is rather poor, but it is good to see the code running.

Concerning the variable mentioned above, the algorithm can make use of the Latin Hypercube Designs (LHDs) when this $useLHDs$ variable is set $true$. LHDs are obtained through Latin Hypercube Sampling (LHS), which is a statistical method for generating a good distribution of collections of parameter values. For more reading about LHS, please refer to [2]. In most of the cases we had the $useLHDs$ variable set $true$. We used LHS as a means for automatic parameter tuning, having thus a better controlled distribution for the parameter values involved. In these situations you will have 30 different average results in the output folder "TSC" and to the corresponding function folder. Surely, some of the configurations are not suitable for the problem and it is expected that not all of them give good results; actually, for some hard functions, there are only a few configurations that give good output. The user is advised to import the obtained result "*.res" file in Excel (or something similar) so that they can be visualized easily.

When the LHDs are used, the number of runs (repeats) is read from the input file and it is set in all cases to 30. These input files are located in the *Parameters intervals* folder and they are distributed depending on the values for the mutation strength parameter: if the intervals of the input function domain are larger, the mutation strength parameter can take higher values. The following separations were considered:

- $[0, 5]$ for functions: Ursem F1, Ursem F3, Ursem F4, Six-Hump Camel Back, Shifted 6-hump, Waves, Shubert, Michalewicz
- $[0, 15]$ for functions: De Jong, Branin RCOS, Rastrigin, Composition for 2 dimensions
- $[0, 30]$ for functions: De Jong, Rastrigin, Composition, all 10 dimensions
- $[0, 80]$ for Ackley function.

3 The input parameters file

In the main folder the "parametersfile.txt" can be found. All the data in this input file is read from the "Main" class; even if the same variables appear in the "Main" class with values, they are overwritten by the values in the parameters file. The order of the parameters in the input file is not important, the reading of the value is done corresponding to the variable at the left.

The following parameters can be set:

popSize - the size of the population. In our experiments we had values up to 200.

functionNo - the number of the function that is currently optimized. There are currently 19 manually added functions and they are followed by the CEC 2005 benchmark functions [3].

vectorSize - refers to the number of gradations that are used or, more precise, the number of intermediate points that are used for detecting whether two points follow the same optimum or not. It is used in the *detect-multimodal* method. Depending on the multimodality of the function, this can have values starting from 1 up to 15 (positive integer). The higher its value, the more precise the method is in detecting if two individuals follow the same peak or not, but the more expensive it gets in consuming fitness evaluations.

pc - the recombination (crossover) probability. It has its value in the interval [0, 1].

pm - the mutation probability. It has its value in the interval [0, 1].

mutationStrength - the mutation strength that is used for controlling the mutation with normal distribution. Its value is real and its choice depends on the function to optimize and its definition interval. In experiments we used values up to 80 when the number of dimensions was also very high.

fitnessEvaluationsTotal - the stop condition given by a fixed number of evaluations. It is an integer value and we set it often to 30000.

eps - the error that is allowed when the optimum is said to be reached. We had it 0.1.

noOfVariables - how many variables the function for optimization has. It is a positive integer.

selType - the type of selection for reproduction: it can be 0 or 1, meaning it is performed globally, on the entire population, or locally, within each subpopulation. This parameter was used especially at the beginning of the experimentation, we remained with global selection, that is the value 0.

replaceParentAutomaticallyMutation - a boolean variable (*true* or $false$). If *true*, the offspring obtained after mutation replaces the parent even if it is not better in terms of its fitness. We let it $false$ in our experiments, meaning that some sort of elitism was embedded.

useInteriorPoints - there has been an attempt from us to use the evaluated interior points within the *detect-multimodal* method in such a way that when the two initial points follow the same peak, the best interior point computed is compared to the two points and, if better, it would replace the worst of the two initial individuals, thus accelerating the convergence of he population. We found out that the idea was not successful, probably due to the fact that the population became too early homogenous and exploration was left out, so we abandoned it, but we did not remove the parameter. It is only switched of by setting it $false$.

tournamentSize - the selection for reproduction type is tournament and the size of the tournament can be set from the parameters file. In our experiments we used a binary tournament, so its value was 2.

Except from these parameters, there are other important ones that can be set directly from the $Main.java$ file. The most important ones are:

percentSeeds refers to the percent of the population that can be included in the set of seeds. The seeds are copied from one generation to another and if too many individuals were seeds and the function was highly multimodal, the entire population could be blocked into seeds and no evolution would happen. Therefore, only a limited number of individuals can be seeds (20 being a good default value).

noOfRuns - how many repeats to be performed for putting some objective results as average value. 30 is a good value but, just for testing whether the code runs OK, 1 should be used for getting quickly the results. Note that when *useLHDs* is *true* the *noOfRuns* is already set 30, as its value is read from the input LHD file.

useLHDs - whether the LHS are used or not.

findBestFitness is an interesting variable: for multimodal functions one could search for the best solution found (in which case the $findBestFitness$ should be set $true$), or one could be interested in the number of peaks found when the function has several peaks that are important (*findBestFitness* $= false$ in that case). Note that the algorithm is insensible to such modifications, but in the output file the first column gives this result - the best fitness evaluation when *findBestFitness* $= true$ and the number of peaks found when *findBestFitness* $= false$.

4 Adding another function

As mentioned above, there are currently 19 manually added functions and they are followed by the CEC 2005 benchmark functions [3]. Then, there is also the function with the number 50 which represents the practical application in [1]. The names for the first functions can be found in the "Main.java" file, just below the declaration of the $functionNo$ variable, at the beginning of the file. The rest can be found in [3] - from these we used only *Shifted Rastrigin F9* and *Rotated Hybrid Composition Function F21* in [3], that is $functionNo = 22$ and 34, respectively. The two functions are calculated in the *readParameters(Properties prop)* method inside the $Main$ class.

All the names for the functions are also introduced in the method *getFunctionName()*; this is used for understanding for which functions are the results printed. So, it would be a good idea to add the name of the function that you intend to add in this method.

Next, the function should have a *case* included in the $eval(Individual\ \_c)$ method. To refer to one position (gene) of the individual $\_c$, then the item *genes* will be used as $\_c.genes[i]$ for the i-th gene. The result of the evaluation of the function should be put into the *ret* variable.

Next step is to include the positions of the *peaks* and that means adding a *case* in the $setPeaks()$ method for our function. Naturally, the case number should be identical to the one from $functionNo$, $getFunctionName()$, and $eval(Individual\ \_c)$.

Last thing to do when adding another function is to insert in the *Individual(int* $\_functionNo$*, int* $\_noOfGenes$*)* constructor (of the $Individual$ class) the left and right domains in the corresponding *case* value.

However, if you plan to use the LHDs (and I advise you to do so for finding fast good parameters and see many output results), there are a couple of things to be done:

- Insert into the *getInputFileName()* method the path for the current function to the LHD input file. This selection is based on the values that the mutation strength should take: below 5, below 15, below 30 or below 80. Please, read section 2 for details.
- Create in the TSC folder an empty folder for the function with the exact name you previously set for this function in the *getFunctionName()* method.

5 Other options

There are many methods implemented in the code that are not used in the current form of the code but they could be used if necessary. For instance, there are many output methods for printing results in the proper format for using them with the R statistical software. Some print the results at the console, others write them to a file. Here and there in the code, there are (a few) comments that can help one understand what lies in there. Many other methods are included in the code and they can called for other observations, e.g. one can see the best, worst and average fitness evaluation, prepare data for plotting in R individuals from different species with distinct colors, how many evaluations are spent per generation, etc.

6 LHD result files

The results files when *useLHDs* is *true* are saved in the TSC folder and then in the corresponding function folder. The extension of the file is *res* and it should be printed in a spreadsheet in order to visualize/sort the results properly. If several runs are done for the same function, the results are saved in the same file, below the previous values. The user should not pay much attention to the console output, the interesting part is in the output file.

The first row contains a header with the meanings for the values below. The first result may represent the best fitness evaluation when the parameter *findBestFitness* = $true$ and the number of peaks found when *findBestFitness* = $false$. This is the main result, and it is followed by the important input parameter names (the ones that have major influence over the results and are read from the input LHD file) and then some other results follow: number of peaks or best fitness evaluation, depending on what was the first output, the peak ratio, number of attraction basins detected, percentage of the runs when all desired peaks are found, the peak accuracy for all peaks - meaning how close they are as fitness evaluations to the optima, distance accuracy - how close they are to the optima as locations, number of generations, runtime, and other input parameters that are not read from the LHD input file.

7 Feedback is welcome!

Hope that this file will get you going with the code. Otherwise please, do not hesitate to ask for help at catalin.stoean@inf.ucv.ro for setting the program running.

References

[1] **Stoean, C., Preuss, M., Stoean, R., Dumitrescu, D.**, Multimodal Optimization by Means of a Topological Species Conservation Algorithm, IEEE Transactions on Evolutionary Computation, Vol. 14, No. 6, pp. 842-864, 2010.

[2] **Iman, R. L.**, Latin Hypercube Sampling, vol. 3, Encyclopedia of Statistical Sciences, Wiley, NY, pp. 408-411, 1999, http://www.swtechcon.com/PDF/appdixa.pdf.

[3] **Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., Tiwari, S.**, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real- Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2005.