Guide

1 About the program

This program provides a subpopulation-based evolutionary algorithm with multi-level parallelism to take advantage of parallel architectures involving multicore CPUs and multiple GPUs for accelerating an electroencephalogram (EEG) feature selection problem. The procedure has been mainly developed with MPI to distribute subpopulations among the nodes of the cluster. In each node, two scheduling alternatives for evaluation of individuals according to the number of received subpopulations (one or more), have been implemented. Inside of each node, OpenMP is used to distribute dynamically either subpopulations or individuals among devices. The fitness evaluation of the individuals is performed by using OpenMP in CPU and OpenCL in GPU. This way, by taking into account the devices characteristics, the procedure provides three parallelism levels in CPU and up to four levels in GPU.

2 Program compilation and use of parameters

There is a *Makefile* file to build the project. Running the following order in a Unix shell the program will compile:

```
make -j N_FEATURES=NF COMP=COMPILER
```

Where *NF* is the number of features to use (columns) of the database, which must be between 4 and the total number of features of the database. Variable *COMP* set the MPI compiler (*mpic++* by default). The executable file, named *hpmoon* will be generated in the "*bin*" folder. For running it, in the shell the next order must be executed:

```
mpirun --bind-to none --map-by node --host node1,…,nodeX ./bin/hpmoon –conf config.xml
```

Where *config.xml* is the necessary configuration file for the correct performance of the program, specified by the *–conf* option and located in the root folder of the project.
In addition, the user can indicate separately through line arguments the most of setting of the XML file. Table 1 summarizes the list of parameters and their possible values, and how to use them in the line of arguments. In any case, the special option *-h* displays the available options and examples of use.
The option *"--map by node"* is mandatory because is necessary to guarantee that the MPI processes and the nodes are mapped correctly. In the XML configuration file, the information of the devices for each node is ordered according to the MPI process id.
The option *"--bind-to none"* is also mandatory as the program uses OpenMP threads to evaluate the fitness of the individuals. This option avoids the mapping of all OpenMP thread to the same CPU core.
On the other hand, the *Makefile* file contains a rule to generate *Doxygen* documentation in the *"docs/html"* folder. This can be done by running the following command:

```
make documentation
```

Finally, the files and documents generated when compiling the project can be deleted. There are two types of cleaning depending on the content to be deleted. The command:

```
make clean
```

Deletes the following contents:

- **Binary files.**
- **.o files.**
- **~ files.**

For a complete cleaning, run the following command:

```
make eraseAll
```

Which will remove the same content as the previous command and also the following content:

- ***gnuplot* files.**
- **Documentation files** generated by *Doxygen*.

gnuplot files contain the fitness of the individuals in the first Pareto front and the necessary source code for the *gnuplot* program. If the user would generate a graph using *gnuplot* and the source code generated by the program, it will also be deleted when using this command.

3 The XML configuration file

The XML configuration file is required to run the program. The parameters of the XML file are read and used at runtime while the parameter used in the *make* command is read and used at compile time to avoid dynamic memory. The parameters are:

- ***NSubpopulations*** is the total number of subpopulations (only for islands-based model).
- ***SubpopulationSize*** is the number of individuals of the subpopulation.
- ***NGlobalMigrations*** is the number of migrations of individuals between subpopulations of different nodes.
- ***NGenerations*** is the number of evolves of a subpopulation (generations of individuals).
- ***MaxFeatures*** is the maximum number of features initially set to "1".
- ***DataFileName*** is the name of the file which will contain the fitness of the individuals in the first Pareto's front.
- ***PlotFileName*** is the name of the file which will contain the *gnuplot* code for data display.
- ***ImageFileName*** is the name of the file which will contain the image data (graphic) after using the *gnuplot* command to generate it.
- ***TournamentSize*** is the number of individuals competing in the tournament.
- ***NInstances*** is the number of instances to use (rows) of the database.
- ***FileName*** is the name of the file containing the database.
- ***Normalize*** specifies if the database must be normalized or not.
- ***NDevices*** is the number of OpenCL devices that will run the program in a specific node.
- ***Names*** specify the names of the OpenCL devices that will run the program in a specific node. The values must be separated by commas.
- ***ComputeUnits*** specifies the compute units for each previous OpenCL device that will run the program. The values must be separated by commas too and in the same order than their corresponding devices.
- ***WiLocal*** specifies the number of work-items (threads) per compute unit for each previous OpenCL device that will run the program. The values must be separated by commas too and in the same order than their corresponding devices.
- ***CpuThreads*** specifies the number of CPU threads to use in the fitness evaluation. If this parameter is set to "1" and *NDevices* is set to "0", the program is run in sequential mode.
- ***KernelsFileName*** is the name of the file containing the OpenCL kernels.

The following table summarizes the restrictions of input parameters. The parameters passed to the *make* command are shown in uppercase. In lowercase, the parameters found in the XML configuration file. If a parameter has the "*CMD OPTION*" column as "-", it means that this parameter is not available as argument of the command line.

PARAMETER	RANGE	CMD OPTION
N_FEATURES	4 <= ***NF*** <= Number of features of the training database	-
NSubpopulations	1 <= ***NP***	-ns
SubpopulationSize	4 <= ***PS***	-ss

NGlobalMigrations	1 <= ***NM***	-ngm
NGenerations	0 <= ***NG***	-g
MaxFeatures	1 <= ***MaxF***	-maxf
DataFileName	Valid filename	-plotdata
PlotFileName	Valid filename	-plotsrc
ImageFileName	Valid filename	-plotimg
TournamentSize	2 <= ***TS***	-ts
NInstances	4 <= ***NI*** <= Number of instances of the training database	-trni
FileName	Existing training database	-trdb
Normalize	1 or 0	-trnorm
NDevices	0 <= ***ND***	-
Names	Existing device name	-
ComputeUnits	1 <= ***CU***	-
WiLocal	1 <= ***WL*** <= Maximum number of local work-item of the device	-
CpuThreads	0 <= **CT**	-
KernelsFileName	Existing kernels file	-ke
Display usage	-	-h
List OpenCL devices	-	-l

Table 1. Range of values of the input parameters and how to use them from the arguments line (if available).

4 OpenCL optimization and limitations. MPI and OpenMP use

The following points should be considered to obtain good performance when running the program:

1) The evaluation function for each individual has been parallelized in GPU with OpenCL. A compute unit is formed by *WiLocal* work-items and evaluates only one individual. Therefore, *WiLocal* should be a multiple of 32 or 64 according to the device for improve the performance. The user can approximate the optimal value of *WiLocal* and *ComputeUnits*. The value is calculated as the number of stream processor or CUDA cores divided by the number of compute units. For example, the *Nvidia GeForce GTX 770* has 1536 CUDA cores and 8 compute units, so 1536/8 = 192 local work-items, but sometimes it is better to increase this value, for example, 256, 512 or 1024 according to special cases. In the case of 256 work-items, *WiLocal* = 256 and *ComputeUnits* = 8, comprising in total of 256 * 8 = 2048 work-items. The best combination is determined by the characteristics of the problem. In CPU, *CpuThreads* should have a value equal to the number of logical cores.
2) The sort function according to the Pareto's front, `nonDominationSort`, contains one loop of quadratic order and is related to the number of individuals. For good quality results it is not necessary to increase the number of individuals too. It's better to increase the number of iterations of the program (number of generations), or the number of subpopulations.
3) On GPU, the program gets better performance with values of *N_FEATURES* and *NInstances* higher than the number of local work-items. However, the database is stored in local memory and their capacity is very limited (approximately 49 KB depending on the device). So the program will abort if the database is too big.
4) If multiple devices are specified, and only one subpopulation is present in the node, the evaluation of the individuals is distributed dynamically among the devices by using OpenMP pragmas. Each OpenMP thread handles one device. This way, each device is independent and compute chunks of individuals equals to its number of compute units or CPU threads until all individuals are evaluated.
5) To run the executable, only one MPI process is necessary. This is the best scenario when only one computer or node is available. However, the program can be run by using more than one MPI process. In this case, the MPI process 0 distributes the subpopulations among the available workers (nodes) and the rest of processes are the workers (MPI processes 1, 2, …). This situation should be considered when the program is run on a cluster which contains multiple nodes.