Metis block

**Task 1: Application research**

The first step in the development of the partitioning application was to understand the basics of the tools we were about to use for the implementation. We decided to use metis as it was an easy to install tool, which could be set in any HPC facility in a reasonable time. Additionally, since the purpose of our application is not performance, but usability.

The first work package consisted in reading thoroughly the Metis manual available at the Metis webpage[REFERENCE TO METIS WEBPAGE] and create a summary for every member of the team to properly understand the functions and standalone programs available for partitioning. The summary document is available in the appendix[REFERENCE TO APPENDIX AND REMEMBER TO ADD THE METIS BASICS DOCUMENT].

**Task 2: Application planning**

The group decided to implement the application based on a ssh connection from the webpage, and for being able to implement the settings for the partitioning, we decided to use a configuration file that checked a series of integer values. The application would be split in the configuration file reading function and the partitioning function, based on those parameters.

**Task 3: Configuration file**

After defining the application components, the team focused on defining the configuration file properly and coding an algorithm to read it and store the values in an array, for being able to partition taking those values into account.

**Task 4: Partitioning code**

The first approach was to develop a C program using the Metis Library functions available, implying a graph or mesh file reading and decomposition. The team developed an initial prototype for this purpose, but it had some issues with memory allocation for big input files, hence the approach was changed to exploit the standalone partitioning programs.

This implied losing some of the configuration options, but the memory allocation was no longer needed, and the application was quickly developed and tested against basic graphs and meshes. For calling the standalone programs, the configuration file reading function was modified to generate a string with the required command,

**Task 5: Output**

The team studied the best way to rename the output files, and the general output name filename.out.parts.<# of partitions>.<index> was implemented. This way, the user would be able to identify the number of partitions easily, and save results from different settings for the same input. The index value is increased each time the program is executed for the same file with the same number of partitions.

**Task 6: Code evolution**

Once the website was finished, the team spent some time perfecting the interface between the website and the application, and some minor tweaks had to be implemented in the code, for guaranteeing seamless interaction. The final code is available in the Appendix[ADD REFERENCE AND PUT CODE INTO THE APPENDIX].