IFREMER BREST

Optimization for MARS code

Automatic MPI 2-D domain decomposition

Luc Guéret

07/08/2009

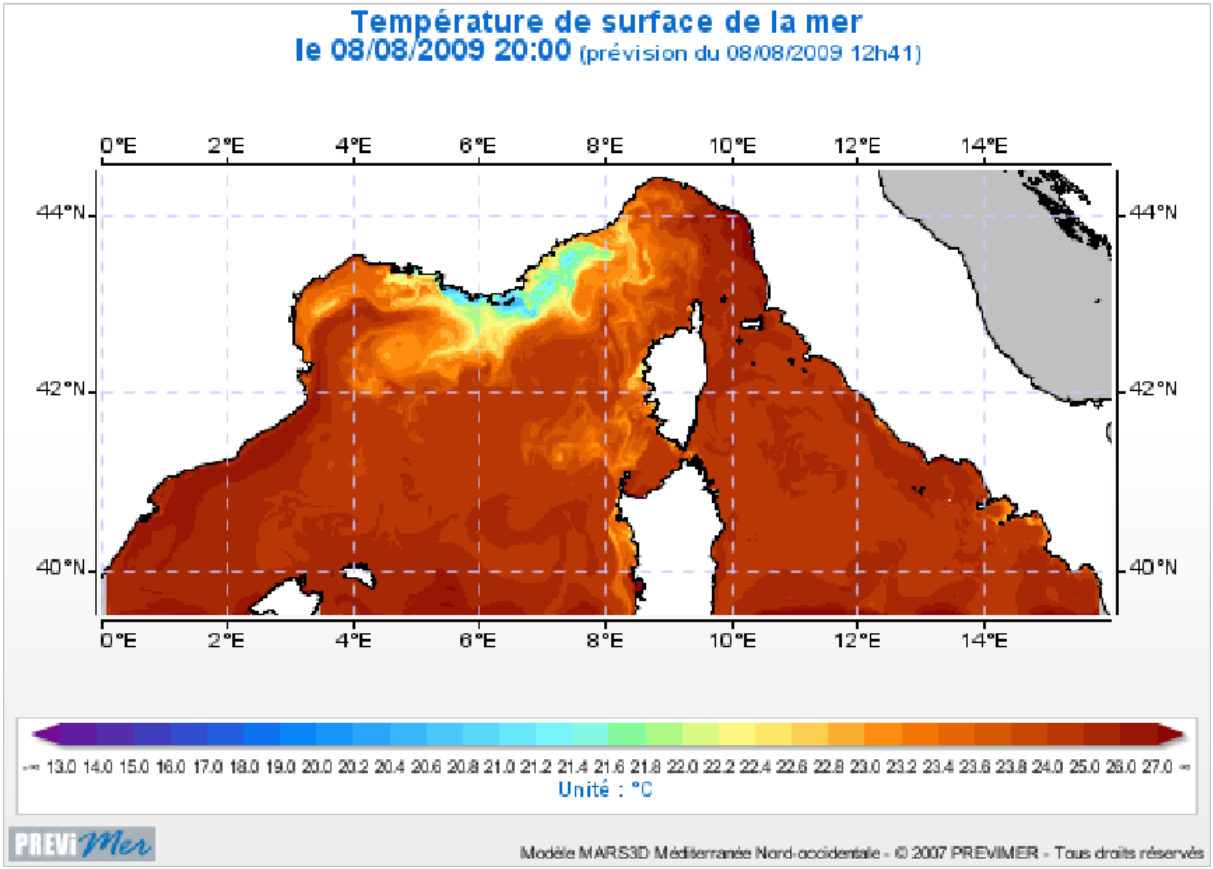
Maître de stage: Tina Odaka

Summary

Introduction

IFREMER uses a super calculator called CAPARMOR (CAlcul PARallèle Mutualisé pour l'Océanographie et la Recherche), which can run parallelized codes, with MPI (Message Passing Interface), to treat bathymetric data.

The data treated is a group of points on a map, which contain more or less information, depending on what result is calculated : weather prevision, current, animal population in sea… The code used is MARS (Model for Applications at Regional Scale), which numerically solves fluids mechanics equations.



We would like to shorten up the computational time of MARS. Faster a job is finished, more jobs can be launched, and especially, the more often same simulations, with recent data, can be launched, to have fresh information. If we can make all the cores working for the same size of data, we will not loose computational time by waiting a over loaded cores to finish up his job.

The purpose of my stage was to find a way to distribute 2D bathymetric data into P groups of equally wet points. Indeed, up to now, the repartition was handmade, the purpose is to turn it automatic, with a FORTRAN program. These calculation data are only the 'wet points' of bathymetries. We have to adapt to any type of map, to give the best repartition possible.

The restriction, due to the structure of the global code, is that the division of the map must have the form of lined up rectangles, in both dimensions.

I) Suggested solution explanation

1) Efficiency criterion

The total working time for P parallel nodes is the working time of the one which is the most loaded. We treat the average load P times, whereas during that time, we could treat the max load P times. That is why the objective efficiency criterion for any map is the division of the average load by the maximum load.

We tried to increase this criterion by decreasing the max load of P nodes. It is impossible to test all the existing solutions, due to complexity, therefore we will have to find tricks, to get the more close possible from the perfect solution.

A compromise must be made between the time spent to treat wet points inside a core, and the exchange between cores. It is possible to handle this compromise, by obtaining the concrete ratio between one atomic time of each of those two different costs. Then we can have the total time cost. If this data is not available, it is possible to give a minimum value, that the thickness must not cross. This condition allows to avoid having too thin rectangles, situation corresponding to many exchanges, compared to intra core calculus.

2) Different methods for different cases

a) One dimension

When P is small, a one dimension division is the best solution, as far as the areas are thicker than the minimum given. The subdivisions are found by counting, one by one, the wet points, so that making rectangle shape heaps, the more equal possible.

b) All active / looking for bigger grids

Two dimensions subdivisions are not so obvious to determine. Indeed, if the previous method, executed separately vertically and horizontally, succeeds in getting a very low standard deviation, it has a very hazardous effect on the maximum.

If the all active option is imposed, we have nothing more to do than going to the next step. Otherwise, we may test larger grids, in order to be better centered on wet points place. All we have to do is to keep the coordinates of the useful domains, and leave the empty domains that remain.

c) Genetic type algorithm

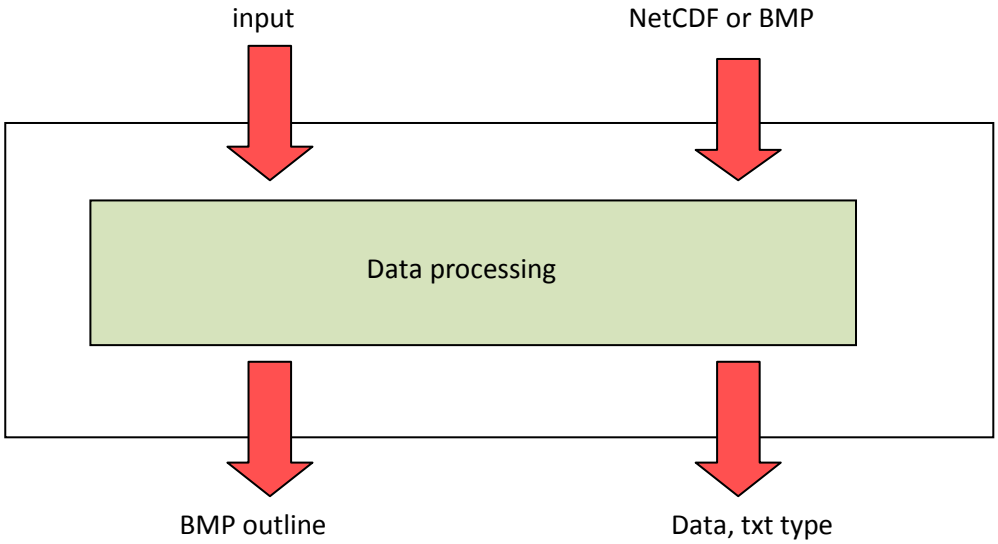
As the number of solutions is too important to browse in a decent time, a random research seems a relevant way. A population of many maps is initialized with the result obtained from now. Then, inside a loop, those maps undergo a slight random modification of their domains borders, until we stop, after having noticed a sufficient stabilization.

The most important aspect is the choice of the succession, from the previous generation to the next. Indeed, to improve the efficiency, the best elements must spread, whereas degenerated solutions have to disappear. Thus, the smaller is the maximum sub domain load, the bigger must be the proportion of descendants of this map in the next generation. A mathematical transformation achieves that goal, and gives this proportion, in function of the maximum.

This algorithm is used in two overall times, first by trying to decrease the standard deviation, then by trying to decrease the maximum. Decreasing the standard deviation doesn't enhance the efficiency, however it makes possible to escape from local potential wells, where the program remains trapped sometimes, otherwise.

II) Program structure

1) File processing

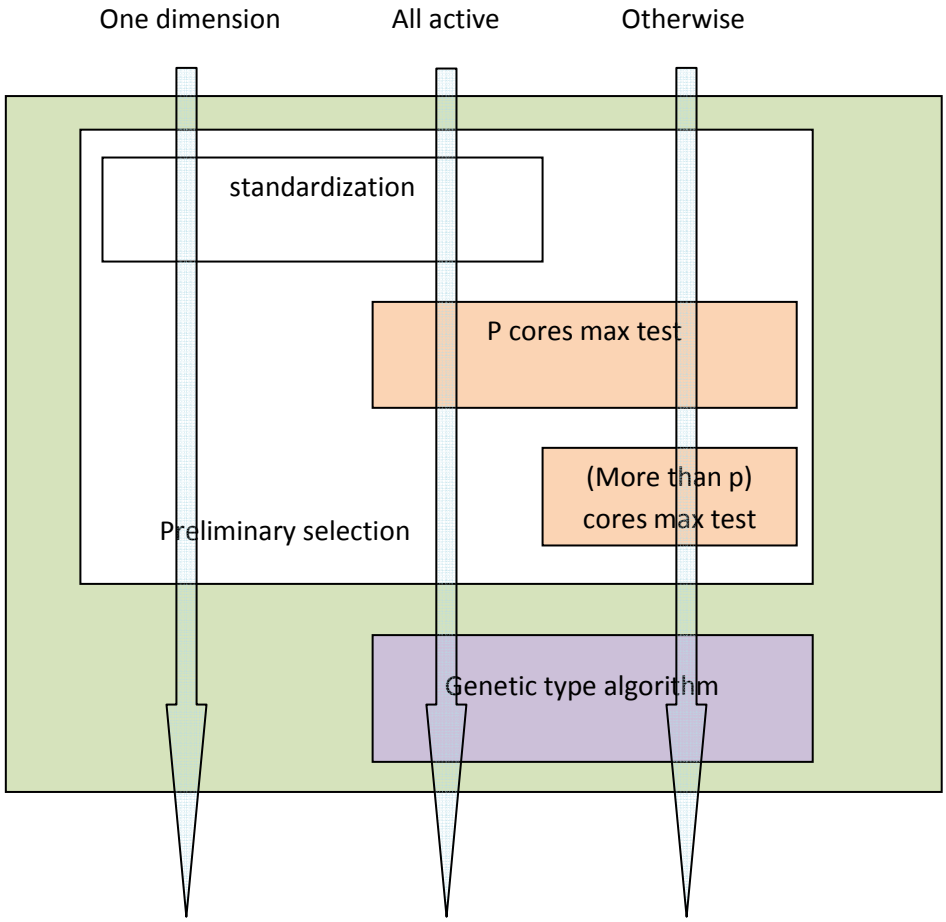


The file manipulation is done by a module called manipulation_fichiers. If the input file is present, the settings enclosed in it are taken to run the program. If not, there are default values.

The input data file format which defined the bathymetry can be a NetCDF, or a bitmap.

An outline of the cutting is automatically recorded, under a bitmap form. Concerning the coordinates of the chosen domains, it is saved in a text file. Those two files have both the same name as the input file containing the map.

2) Data processing



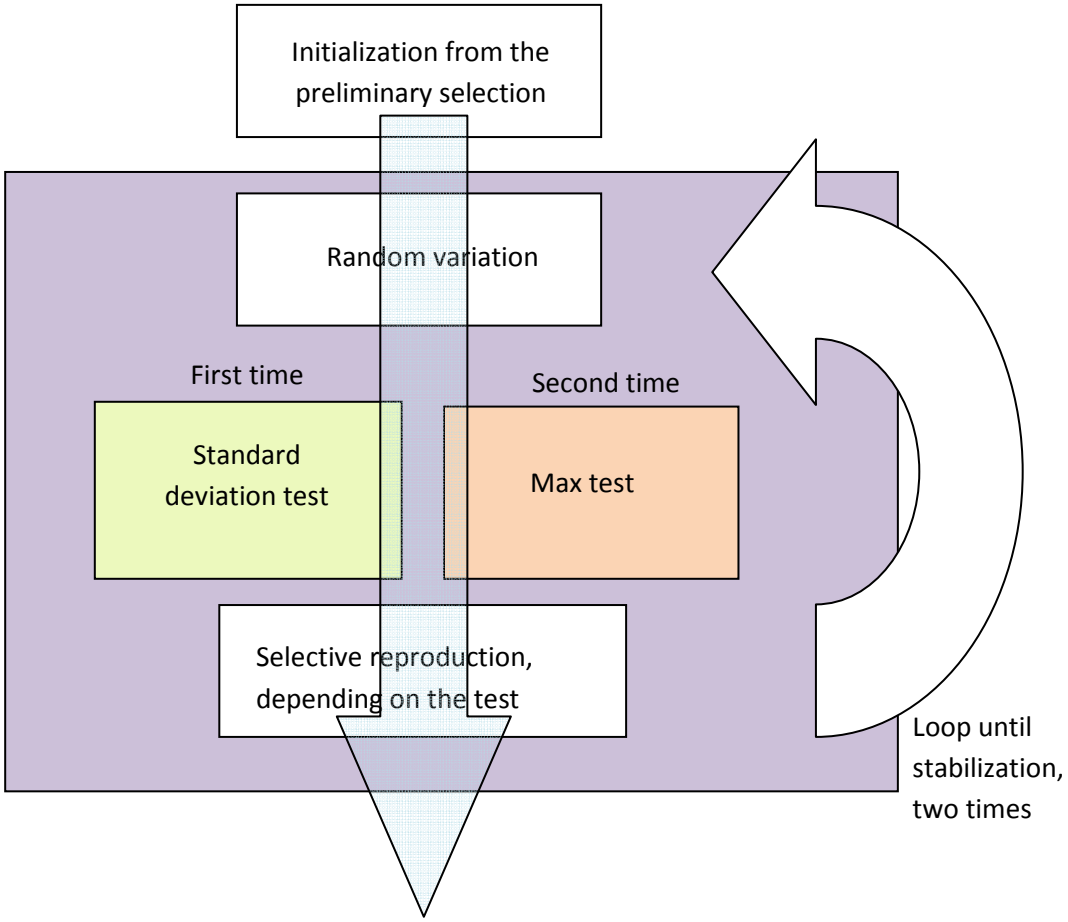
The user has to choose between three cases: one dimension, all active, and otherwise. One dimension implies all active.

The one dimension treatment is only composed of a standardization, equally distributing the amount of wet points between the different domains.

The all active treatment only compares the different possible cuttings, given the possible cases of P divisors. Then it goes to the genetic selection step. It undergoes a standardization to fasten the standard deviation part in the genetic algorithm. The term “all active” only implies that the whole map is shared to the whole set of cores. However, a few sub-domains can turn out inactive. The interest of this option is to adapt to the current computation code, which doesn’t support the exclusion of inactive areas.

As for the treatment done without those options, the preliminary selection also tests grid combinations, with a number of divisions greater than P, to check if a grid can fit with P active domains, or almost; the other domains, which entirely cover dry ground, are listed inactive.

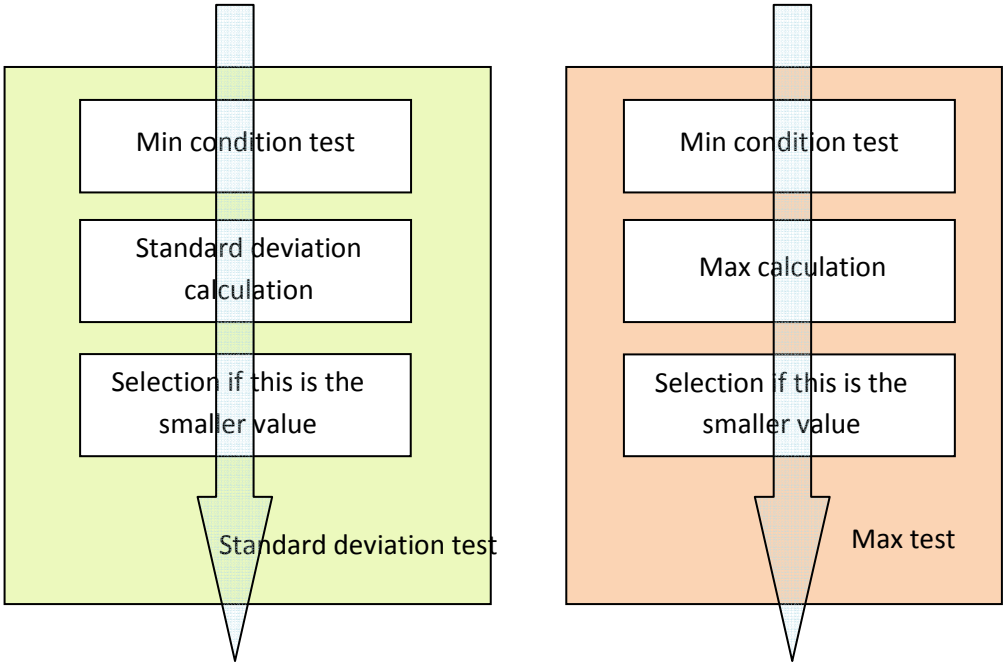
a) Genetic type algorithm



From the preliminary selection result, for each individual, the genetic type algorithm imposes the population to go throw a transformation loop. It stop when it is stabilized.

In each loop, each map grid is first slightly moved, randomly. Then, it is tested, to finally give them more or less reproduction for the next loop, regarding their ranking, according to the test. In a first time, the test is on the standard deviation. In a second time, it is on the maximum. This algorithm provides to progress to a good solution, if not the best.

b) Tests



Those test are executed in different parts of the program, as it is easy to observe, thanks to the colors.

An important detail to note is that the importance of the exchange is taken in consideration in the max test.

III) Results

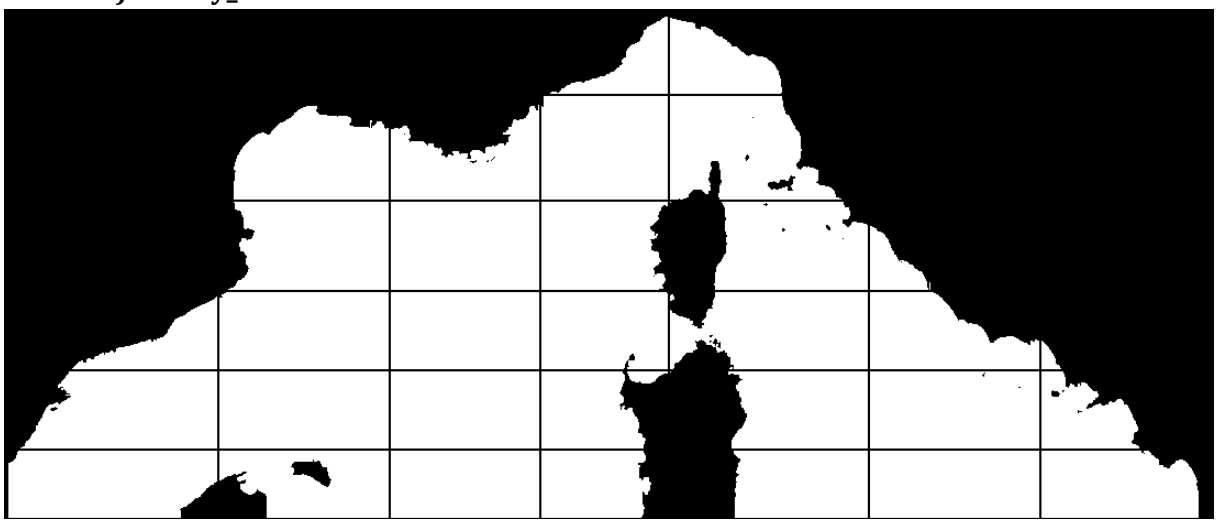
1) One dimension

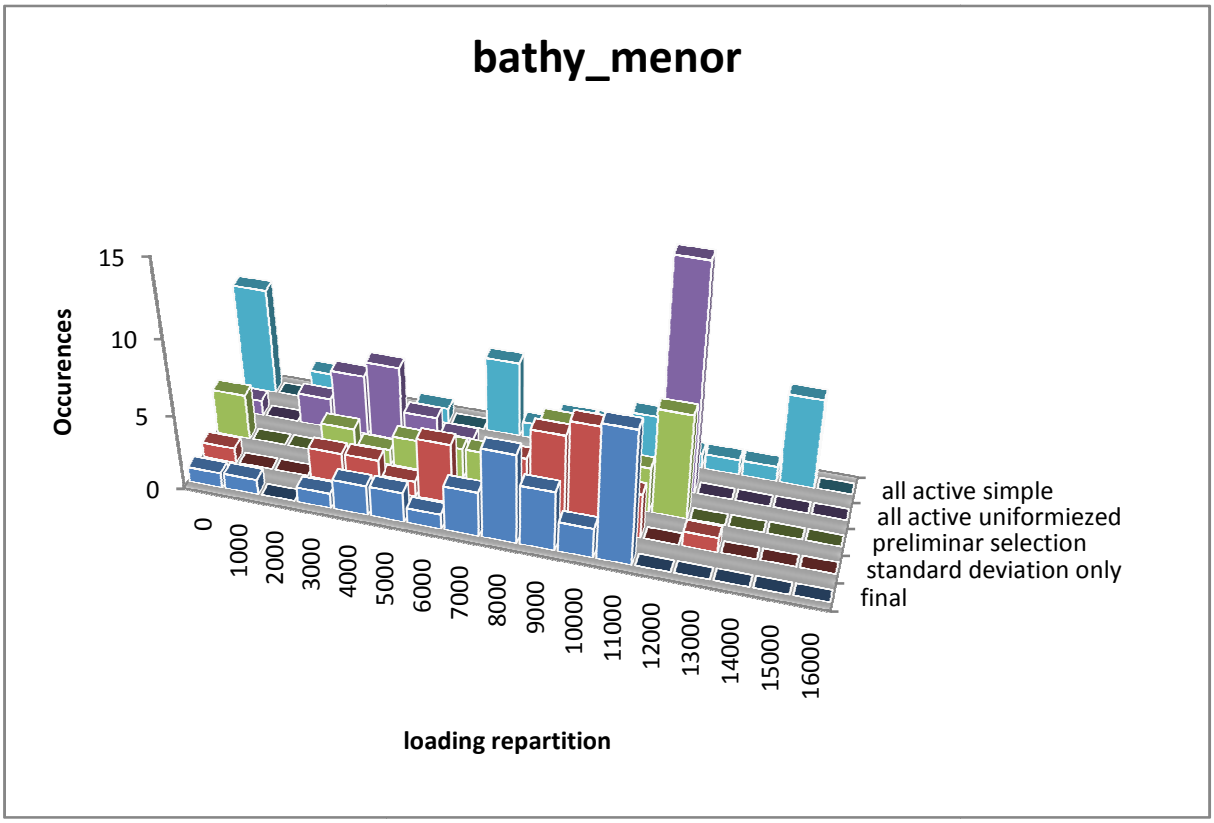
When P is small enough, the best solution is the one dimension cutting. The efficiency is near 100% (efficiency here: 97. 15%). However, for a big P, the importance of the exchanges must not be forgotten.

2) All the maps with P = 32 nodes

The following graphics represents the repartition of the different sub domains loadings. For example, a bar above 1000, and with 1100 to the right, represents the number of cores which have between 1000 and 1100 wet points to treat. They make us possible to observe the relative efficiency to decrease the max, between the different options or steps.

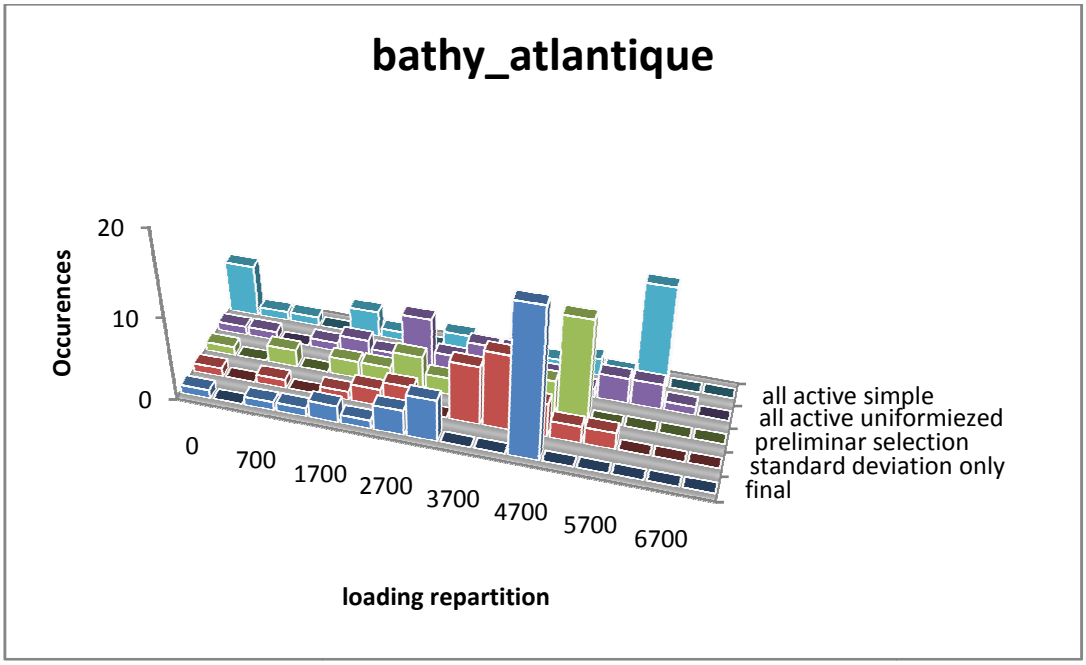
a) Bathy_menor





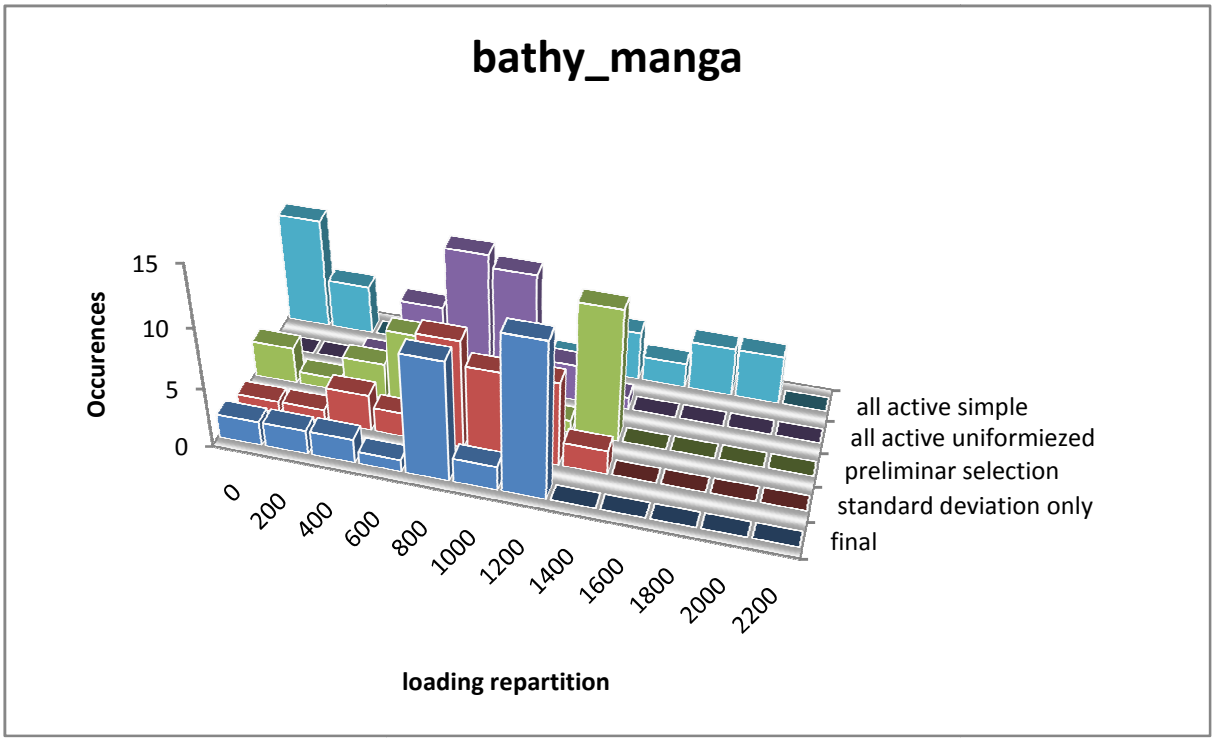
efficiency : 74. 37%

b) Bathy_atlantique



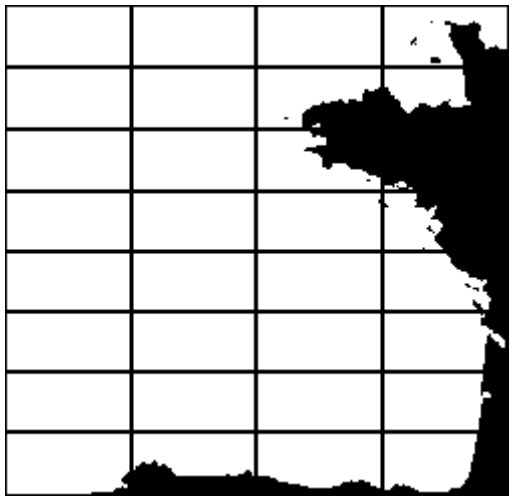
efficiency : 76. 08%

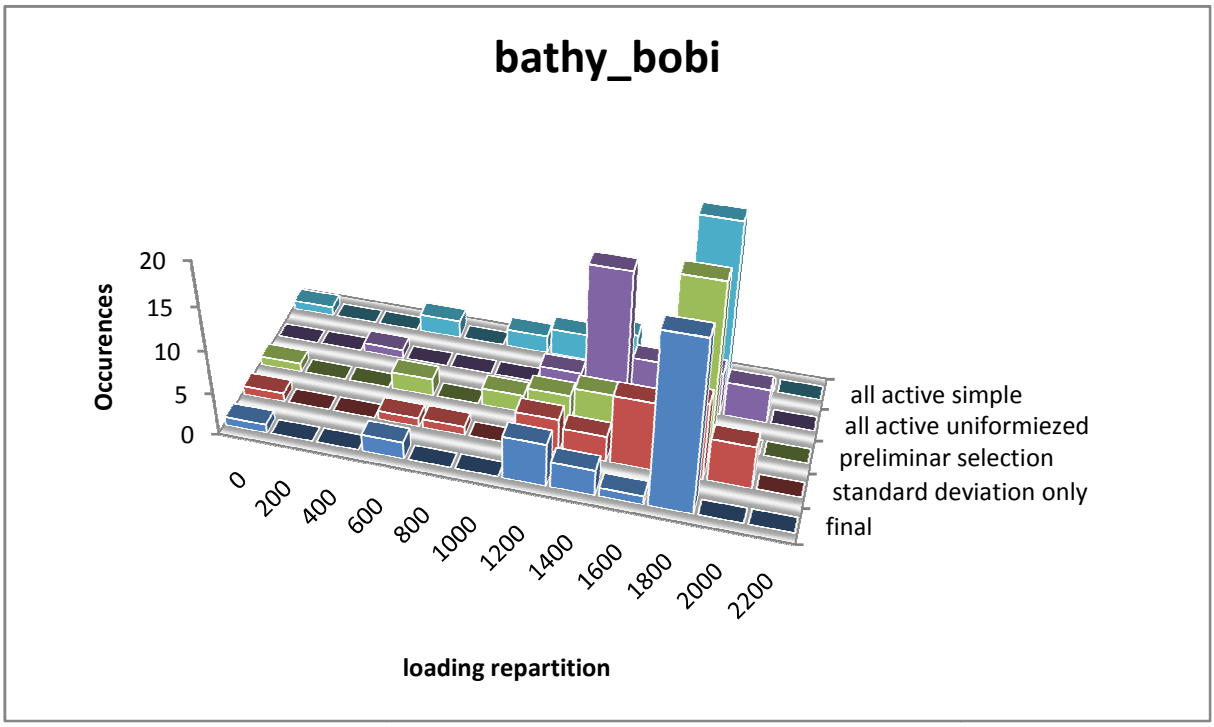
c) **Bathy_manga**



efficiency : 73. 13%

d) Bathy_bobi

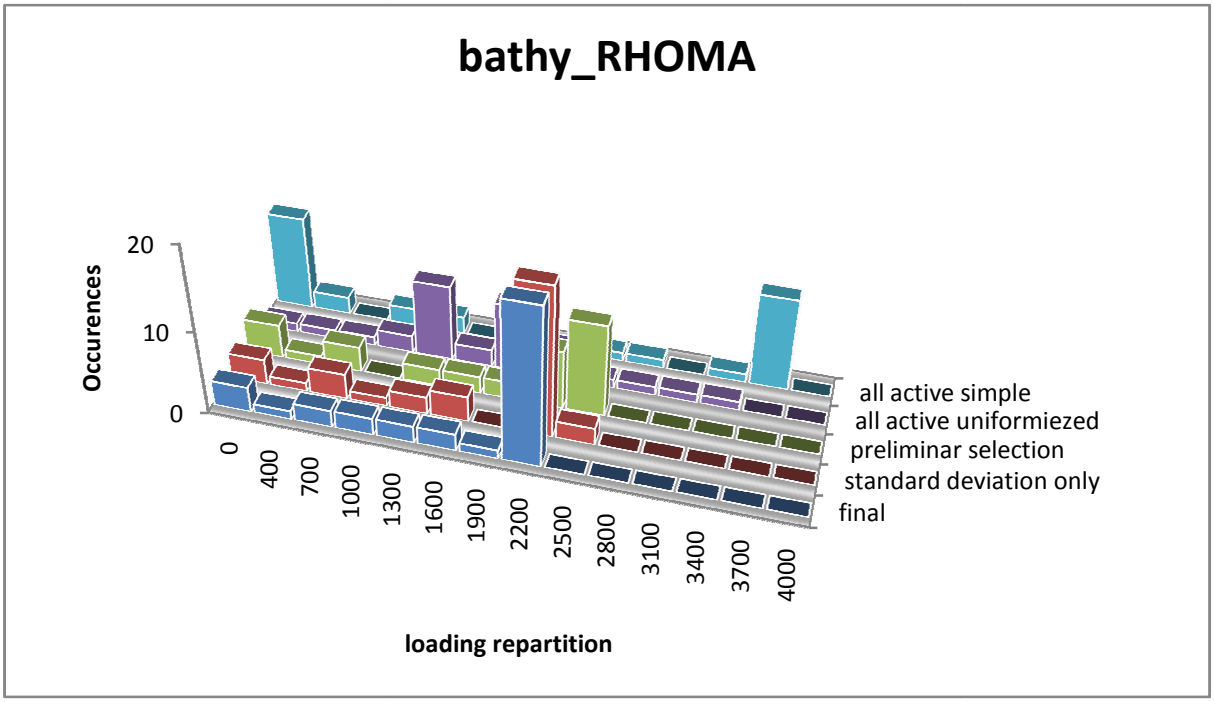




Saturation, due to the big, regular area of wet points

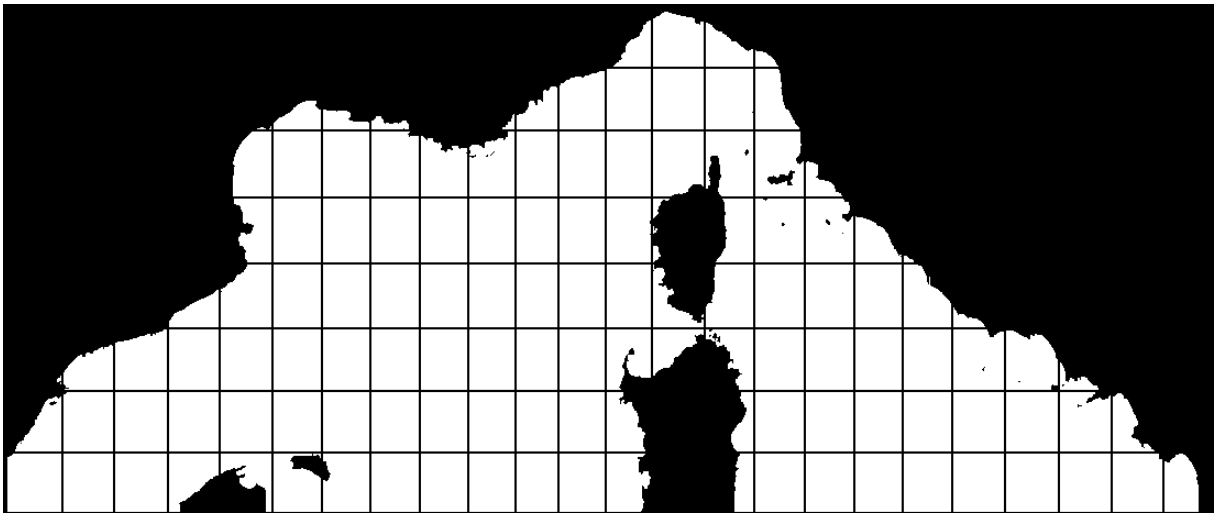
efficiency : 83. 66%

e) Bathy_RHOMA



efficiency : 73. 58%

3) Big values of P



Here is the solution given for P = 128. It is quasi impossible to reduce little by little the maximum, because most of the domains reach it. The genetic type algorithm has no effect. It is not very surprising, we are already in an optimal case, at the end of the preliminary selection.

(efficiency : 76. 03%)

Conclusion

The program gives us average efficiency between 70% and 80%. The one dimension will be preferred for P small. For P too high, the genetic algorithm is no longer efficient, but anyway, the result of the preliminary selection, by only taking some elements of a larger grid, is most of the time better than 80%.

Some better results could be obtained, by reshaping the overall treatment code. A recursive method dividing the areas in two parts of equal loading, at each iteration (therefore only concerning two power form P) would be perfect for repartition. Nevertheless, this gain should be compared with the influence of more complex exchange between cores.

A sad collateral damage of cutting in the water