Laboratory Work—Project

*The project should be done by pairs of students. Singletons are allowed, triples are not. All the material related to the project (*viz.*, C or C++ source code, Makefile, documentation) shall be uploaded to* madoc *before April 10th. The documentation must describe at a high level and justify the choices made in parallelizing the sequential program.*

Program `optimization-seq.cpp` in the archive optimization.tar.gz on *madoc* implements an *interval arithmetic branch-and-bound procedure* to find the minimum of a binary real function over a box of domains (See Fig. 1).

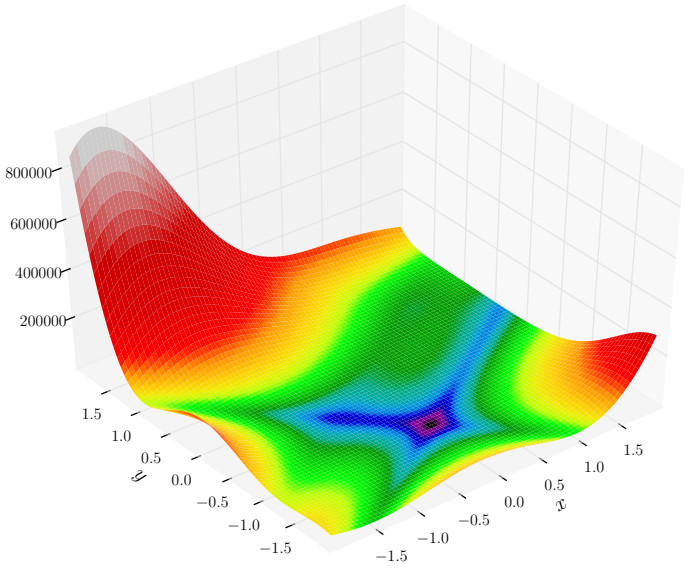

Figure 1: The Goldstein-Price function (Source: Wikipedia)

Given $f(x, y)$, the function we seek to minimize, and $I_x \times I_y$ the initial box of domains in which the minimum is sought, the branch-and-bound algorithm works as follows (See Fig. 2):

1. We start with an upper bound for the minimum μ set to $+\infty$.
2. f is evaluated over $I_x \times I_y$. Thanks to interval arithmetic, the result of the evaluation is an interval $[a, b]$ that encloses all possible values of f for pairs of reals in $I_x \times I_y$:

$$\forall x \in I_x, \forall y \in I_y : f(x, y) \in [a, b]$$

3. If a is greater than μ, we know that the current box of domain cannot contain a minimum of f over the initial box because the minimum value of f in the current box is greater than the worst minimum already found; The current box is then not investigated further;
4. If b is smaller than μ, we know that the current box contains a proven minimum better than the one we knew, and we then update μ to the new value b;

5. If the current box of domains is smaller than some threshold ε, we consider that it contains a minimum (otherwise, it would have been discarded earlier), and we add it to a global list of potential minimizers;

6. If the current box is larger than ε, we split it into four subboxes by splitting the current domains for x and y at their middle. We then call recursively the algorithm on each sub-box.

In addition, when the global minimum is updated (step 4), we remove from the list of potential minimizers all boxes such that the lower bound of f on them is greater than the new minimum.

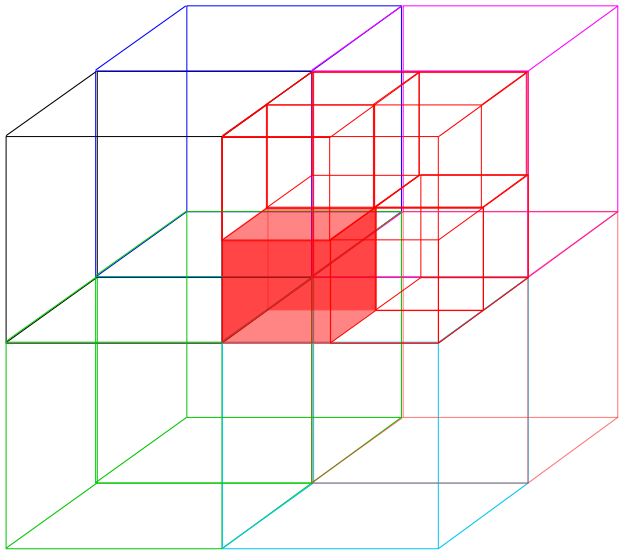

Figure 2: Interval arithmetic Branch-and-Bound procedure. The filled red box is not split further because it is shown not to contain a minimum of f.

To Do

You will find on *madoc* an archive optimization.tar.gz, which contains the following files:

- `functions.h`: header file declaring examples of functions to optimize;
- `functions.cpp`: the implementation of example functions to optimize;
- `interval.h`: header file for the `interval` class and interval arithmetic;
- `interval.cpp`: implementation of the `interval` class;
- `minimizer.h`: header file for the `minimizer` class representing a potential minimizer;
- `minimizer.cpp`: implementation of the `minimizer` class;
- `optimization-seq.cpp`: sequential implementation of the interval arithmetic branch-and-bound algorithm;
- `Makefile`: a Makefile to compile the whole application.

1. Using MPI, implement a parallel version of the branch-and-bound algorithm. Experiment with different ways of choosing the granularity level and report on their effectiveness;

2. Using OpenMP, Intel TBB and/or C++11 threads (your choice), add shared-memory parallelism to your MPI program. Try to get the best performances overall compared with the sequential program.

You are free to develop heuristics and techniques to better take advantage of parallelism for this particular problem.