PARALLEL COMPUTING DIRECTIVES AND OBJECT-ORIENTED PROGRAMMING FOR OPTIMIZATION METHODS

Guido Buzzi-Ferraris and Flavio Manenti

Politecnico di Milano, Dipartimento di Chimica, Materiali e Ingegneria Chimica "Giulio Natta"
Piazza Leonardo da Vinci 32, 20133 Milano, ITALY

This research activity is mainly aimed at showing potentialities in coupling object-oriented programming with parallel computing. Wide margins of benefits could be obtained at the same time in algorithm efficiency and robustness with a relatively small programming effort. The case of unconstrained multi-dimensional optimization in presence of multimodality, discontinuities, and narrow valley issues is proposed as quantitative example.

1. INTRODUCTION

We are undergoing two silent revolutions that directly involve Process Systems Engineering (PSE) and Computer-Aided Process Engineering (CAPE) communities, besides many other scientific and industrial areas: the object-oriented programming and the parallel computing on personal computer (Buzzi-Ferraris, 2010). Both of them have been widely discussed in the literature as they significantly modify the numerical analysis as it was conceived since the second part of the previous century as well as the way to apply numerical methods and algorithms for solving more and more complex and multifaceted issues.
Nevertheless, since it is not clearly stated in the current literature, it is worth remarking that the parallel computing is easy to integrate in object-oriented programming and their combination seems to be particularly appealing as many objects generated by the same class might run simultaneously on different processors or cluster nodes. By thinking parallel and object-oriented both together, it is possible to write by new many algorithms which were not considered for solving numerical problems because of their inferior performances in procedural programming and sequential computing.
Thus, this research activity specifically deals with the development of very robust optimizers that exploit all features of object-oriented programming (Buzzi-Ferraris, 1994; Buzzi-Ferraris and Manenti, 2010a), which allow going beyond the procedural programming and its limitations, and the shared memory that is nowadays available on common multi-processor machines (distributed memory machines such as clusters are not considered for the time being, even though the same reasoning here described can also be extended to this branch of parallel computing).
Basic concepts of coupling parallel computing with object-oriented programming for improving the optimizer robustness and efficiency are stated in Paragraph 2. Some well-known literature tests involving strong and weak multimodality, very narrow valleys, discontinuities in both the function and in its derivatives, and functions that are undefined somewhere in the domain are adopted as validation cases in Paragraph 3.

2. EXPLOITING SHARED MEMORY TO IMPROVE EFFICIENCY AND ROBUSTNESS

Conventional programs easily fail when some specific families of optimization problems have to be solved. Actually, very robust optimizers are required in the following cases:

- The function and/or its derivatives are discontinuous
- The function cannot be approximated by a quadric in correspondence with the optimum

- The function is undefined in some regions and the domain cannot be analytically described
- Very narrow valleys (or steep walls) are present
- The function is multimodal and the global optimum is required

Let us investigate the problem of very narrow valleys. From this perspective, the OPTNOV's method (Buzzi-Ferraris, 1967) seems to be one of the most appealing approach. It is important to realize the reason that makes traditional methods such as Simplex (Nelder and Mead, 1965), Hooke-Jeeves (Hooke and Jeeves, 1961), Rosembrock (Rosenbrock, 1960), Conjugate Gradients, Quasi-Newton ineffective when the function valleys are particularly narrow. All these methods perform one-dimensional searches along some specific axes selected in accordance with the method used. For example, Rosembrock's method is based on the rotation of axes of search to move towards the bottom of the valley. When the valley is very narrow, as qualitatively shown in Figure 1, it might happen that the Rosembrock's axis does not lead to any improvement of the function even though it is reasonably oriented like the bottom of the valley. The same condition occurs when all the traditional methods adopt whatever axis to perform a one-dimensional search.

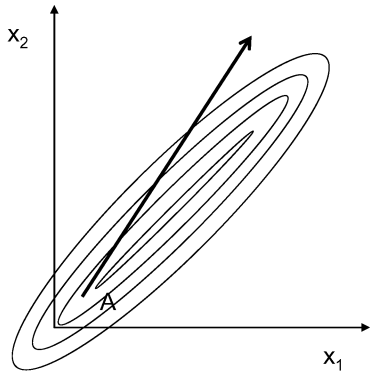

Figure 1. Traditional methods fail with very narrow valleys.

To exploit the search direction that inaccurately detects the bottom of the valley, it is necessary to change the point of view. OPTNOV's method is based on some simple ideas that make it particularly robust, but also efficient in the case of very narrow valleys:

- Whatever optimization algorithm is able to quickly find the bottom of the valley by starting from a point outside the same valley
- The line joining two points on the bottom of the valley is a reasonable valley direction; therefore a point projected along such a direction has good probabilities to be close to the valley
- Nevertheless, this valley direction must not be used as the direction of the one-dimensional search, rather it should be the direction where a new point is projected along
- The new projected point in the valley direction should not be discarded even though it is worse than the previous one; it should be adopted as the new starting point for the minimum search.
- This search must be performed in the sub-space orthogonal to the valley direction to prevent the problem of having small steps. In fact, if the axis is used to minimize the function (above all when it is the first axis of search), it is possible to go back to the previous point A (see Figure 1), by obtaining a very small movement.

This philosophy is particularly effective in an object-oriented programming coupled with parallel computing as many reduced optimizations must be carried out starting from distinct points and they can be independently solved. Consequently, this philosophy of simultaneously solving different optimization problems by starting from distinct guesses allows rationally facing even the search for the global minimum.

The concept to build up a program for effectively tackling all aforementioned issues is rather trivial as it is possible to develop an optimizer consisting of N objects, where N is the number of available processors and each of them uses in turn an optimizer reasonably robust.
Hence, two distinct problems must be solved: the first is the selection of points used in the N objects as initial guess and the second is which kind of optimizer must be used within each of these N objects.
For the sake of clarity, let us call outer the optimizer used to manage the overall optimization problem and to select the N starting points for the N objects and inner the optimizer used within each one of the N objects.
The inner optimizer must be particularly effective to find out the local minimum or the bottom of the valley even in presence of function/derivative discontinuities, slightly narrow valleys, and functions that cannot be approximated by quadratic functions or furthermore that can not be evaluated anywhere.
The outer optimizer must be particularly robust to investigate the function at the macroscale and to have good probabilities to find the global optimum and the proper movement along very narrow valleys. It is worth remarking that every optimization program, although particularly robust, can ensure neither the global optimum nor the solution of some hard problems.

2.1 Outer Optimizer

A lower bound of computational power has been selected: at least a QUAD CORE machine is needed since the outer optimizer requires at least four processors: three of them for applying the OPTNOV's philosophy and the fourth to poll the function elsewhere. The procedure is schematized in Figure 2. A and B are two points on the bottom of the valley. The line passing through them is a reasonable valley direction and the starting points simultaneously selected for three objects are in correspondence with the new points I, II, and III by exploiting the available processors. Distances δ and ε among points can be reduced or expanded according to the results: for example, if the point III brings to a better inner optimum, distances are expanded.
If $N > 3$ processors are available, the remaining $N-3$ processors are used to perform other simultaneously inner optimizations starting from different points (IV…), but, contrarily to what is usually carried out in genetic algorithms, these additional points are not randomly selected. This selection is efficiently carried out by using those techniques adopted and proven for the optimal design of experiments (Buzzi-Ferraris, 1999; Buzzi-Ferraris and Manenti, 2009; Manenti and Buzzi-Ferraris, 2009; Buzzi-Ferraris and Manenti, 2010b). While the inner optimizations of each object go on, the best solutions are collected. The new starting points are selected to maximize the minimum distance with respect to the collected points (see Figure 3).

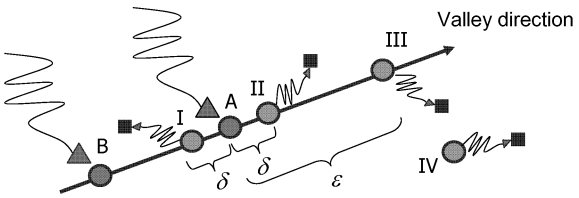

Figure 2. Points A and B are on the bottom of the valley (A is the best one); points I, II, and III are the possible point projections along the valley direction; point IV is used to poll the function in an unexplored region.

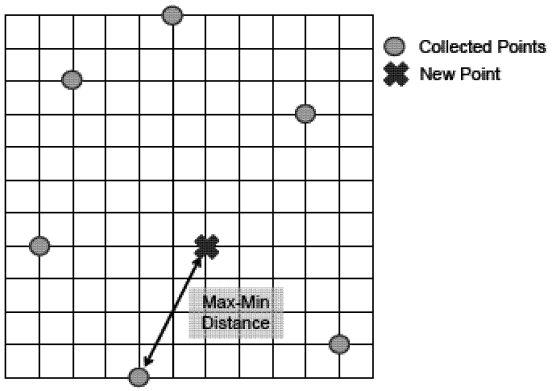

Figure 3. The minimum distance between each point of the grid and the collected points is evaluated. The new starting point is the point with the maximum minimum distance.

2.2 Inner Optimizer

It is worth remarking that even the inner optimizer must be opportunely robust:

- to manage possible first- and second-order discontinuities of the function;
- to overcome possible regions where the function is undefined;
- to effectively find a local minimum or a point at the bottom of the valley;
- to efficiently tackle the problem of slightly narrow valleys.

A modified version of the Nelder-Mead's Simplex method is used since the original Simplex method is reasonably robust to face function and derivative discontinuities, but not enough robust to solve other issues (Conn et al., 2009). Moreover, the original version of Simplex is not predisposed to face infeasible points of the function, but in this case the modification is trivial.

3. NUMERICAL TESTS

Many numerical tests were carried out to check the algorithm robustness for problems of different dimensions. Tests of Table 1 are well-known literature functions for:

- Strong multimodality: Rastrigin (1) and Haupt (2) functions were adopted (Figure 4 and Figure 5).
- Weak multimodality: the function is constant along at least one direction in correspondence of some local minima. Michalewicz's function (3) was adopted (Figure 6).
- Discontinuities and regions where the function is not defined (4). Figure 7 shows the function against the variable x_1 for the optimal value of x_2.
- Extremely narrow valleys: valleys consisting of steep walls make the search of the minimum a problematic issue for many optimizers. Buzzi-Ferraris's function (5) is adopted and reported in Figure 8.

$$F_{RASTRIGIN} = -110 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) \right) \tag{1}$$

$$\begin{cases} F_{HAUPT} = -a, & a > 0 \\ F_{HAUPT} = 0, & a \le 0 \end{cases} \quad \text{where:} \quad a = \prod_{i=1}^{4} \left(\sqrt{x_i} \sin\left(2\pi x_i\right) \right) \tag{2}$$

$$F_{MICHALEWICZ} = -\sum_{i=1}^{n} \left(\sin\left(x_i\right) \cdot \left(\sin\left(\frac{i \cdot x_i^2}{\pi} \right) \right)^{2 \cdot n} \right) \tag{3}$$

$$\begin{aligned} F_{BUZZI-FERRARIS,A} = & \sqrt{-945 + x_1 \left[1689 + x_1 \left(-950 + x_1 \left(230 + x_1 \left(-25 + x_1 \right) \right) \right) \right] + e^{-x_1}} + \\ & + 10\left| x_2 - 10x_1 \right| + 10\left| x_1 - 6 \right| \end{aligned} \tag{4}$$

$$F_{BUZZI-FERRARIS,B} = \left[x_2 - 10000\left(x_1 - 1\right)\left(x_1 - 3\right)\left(x_1 - 5\right)\left(x_1 - 7\right)\left(x_1 - 9\right) \right]^2 + \left(x_1 - 8\right)^2 \tag{5}$$

The algorithm here proposed is able to find the global minima of all functions (1)-(5) as reported in Table 1. In addition, the starting vector and the total amount of iterations is reported. It is worth remarking that the overall computational effort corresponds to the CPU time required to solve all the iterations divided by the number of available processors.

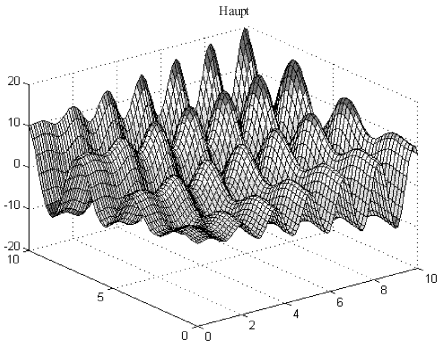

Figure 4. Haupt function

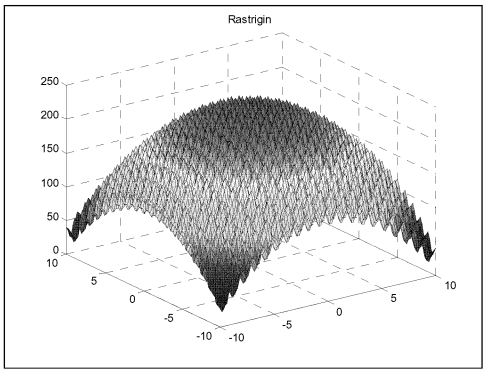

Figure 5. Rastrigin function

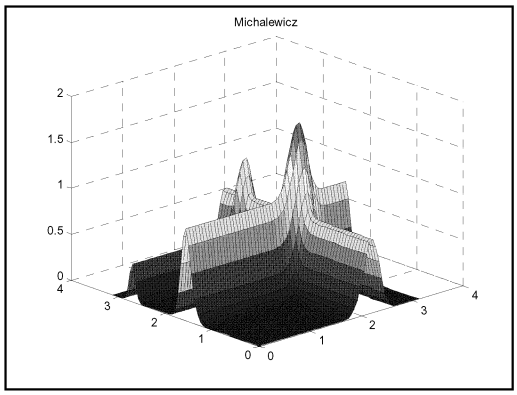

Figure 6. Michalewicz function

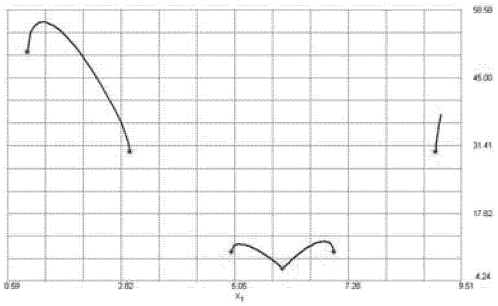

Figure 7. Buzzi-Ferraris's function A

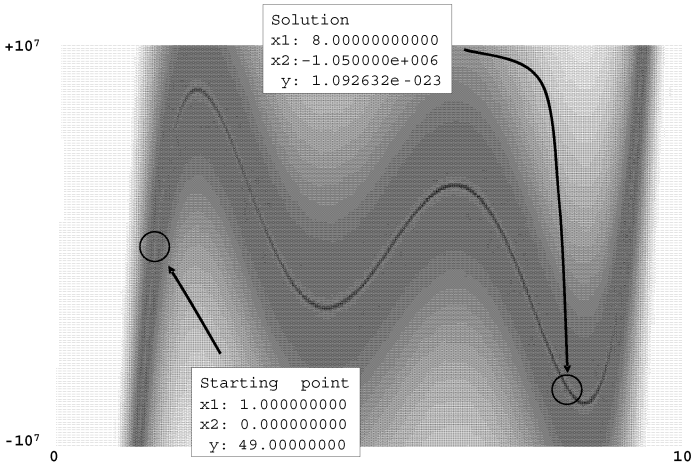

Figure 8. Buzzi-Ferraris's functions B

Table 1. Optimization tests

	Starting point	Number of iterations	Optimum value
Rastrigin $n = 2$	$\mathbf{x}_0 = 8.$	2414	-240
Rastrigin $n = 10$	$\mathbf{x}_0 = 8.$	8357	-1200
Haupt	$\mathbf{x}_0 = 0.$	5559	-19.8630560097267
Michalewicz $n = 2$	$\mathbf{x}_0 = 3.$	1683	-1.80130340983985
Michalewicz $n = 10$	$\mathbf{x}_0 = 3.$	20144	-9.660152
Buzzi-Ferraris A	$\mathbf{x}_0 = \{1.;1.\}$	16714	6.708389
Buzzi-Ferraris B	$\mathbf{x}_0 = \{1.;0.\}$	1085	1.009e-018

4. CONCLUSIONS AND FUTURE DEVELOPMENTS

This preliminary research activity shows the way and reports some benefits coming from the interaction of parallel computing and object-oriented programming. Specifically, the example of C++ class for robust optimization that could generate a series of objects so that each of them could run on a specific processor by increasing the same optimizer robustness with a small programming effort is proposed.

5. REFERENCES

Buzzi-Ferraris, G., 2010, New trends in building numerical programs. Computers and Chemical Engineering doi:10.1016/j.compchemeng.2010.07.004.

Buzzi-Ferraris, G., 1994, Scientific C++. Building Numerical Libraries, the Object-Oriented Way. 2nd Ed., 479pp, Addison-Wesley, Cambridge University Press, ISBN 0-201-63192-X.

Buzzi-Ferraris, G., & Manenti, F., 2010a, Fundamentals and Linear Algebra for the Chemical Engineer: Solving Numerical Problems. Wiley-VCH, Weinheim, Germany.

Buzzi-Ferraris, G., 1967, Ottimizzazione di funzioni a più variabili. Nota I. Variabili non vincolate. Ing. Chim. It. 3, 101.

Buzzi-Ferraris, G., 1999, Planning of experiments and kinetic analysis. Catalysis Today 52, 125-132.

Buzzi-Ferraris, G., & Manenti, F., 2009, Kinetic models analysis. Chemical Engineering Science 64(5), 1061-1074.

Buzzi-Ferraris, G., & Manenti, F., 2010b, Interpolation and Regression Models for the Chemical Engineer: Solving Numerical Problems. Wiley-VCH, Weinheim, Germany.

Conn, A.R., Scheinberg, K., & Vicente, L.N., 2009, Introduction to Derivative-free Optimization. MPS-SIAM Series on optimization.

Hooke, R., & Jeeves, T.A., 1961, "Direct Search" Solution of Numerical and Statistical Problems. J. of the Assn. For Computing Machinery 8, 212-229.

Manenti, F., & Buzzi-Ferraris, G. (2009). Criteria for Outliers Detection in Nonlinear Regression Problems. In J. Jezowski & J. Thullie (Eds.), Computer Aided Chemical Engineering (Vol. 26, pp. 913-917).

Nelder, J.A., & Mead, R., 1965, A simplex method for function minimization. Computer Journal 7, 308-313.

Rosenbrock, H.H., 1960, An Automatic Method for Finding the Greater or Least Value of a Function. the Computer Journal 3, 175-184.