An extension of the Particle Swarm Optimization algorithm

Sonia Ben Ahmed

Abir Khan

Oussama Laaumari

Directed by **Mr. Philippe de Peretti**

Programming in **SAS/IML**

Master 1 Econometrics and Statistics

Sorbonne School of Economics

Year 2021-2022

Abstract

This paper focuses on presenting the Particle Swarm Optimization (PSO) algorithm, a stochastic method that has been inspired by certain animal behaviors in biology such as birds or fishes. The principle of this algorithm is to find a global optimum as fast as possible by analyzing its neighborhood and avoiding diverging paths. Among other optimization methods and depending on the context, it is one of the most efficient algorithms, especially for highly non-linear functions, as others have difficulties to find the global optimum. This paper extends the standard algorithm particularly by building algorithms capable of minimizing single or dual functions with constraints.

Keywords : Particle Swarm Optimization, SAS, Pareto optimality, Multi-Objective optimization problem, Optimization under constraint.

Contents

List of Figures

Chapter 1

Introduction to the Particle Swarm Optimization algorithm

As previously said, the Particle Swarm Optimization algorithm took its origin from biology. More precisely, R. Eberhart and Kennedy, 1995, developed this method based on the study of social behavior of animals. Let's take the example of bird flocks. When they are seeking for food, each bird will be attracted by the best position it has found so far but also by the best position found by other birds. By gaining information at each iteration, at some point, they will converge to the place where food is available. This collective behavior from birds can usually be seen by their V-shape movement. Same intuition goes for shoals of fishes and swarms of bees.

Classified as a Swarm Intelligence approach, the PSO algorithm consists of finding the global solution thanks to a group of particles, called a swarm, scattered in the research space, that share information with each other in order to approach the best position as a group. The PSO algorithm is a metaheuristic model : a function may have several local optimums, but only one global optimum. This algorithm attempts at finding this global solution, or at least, getting near to it. Each particle is a potential solution to the problem. They are assigned an initial speed, position and neighborhood. A particle has the ability to memorize positions and communicate with other particles. Each particle will attempt to reach its best position and memorize it, but also memorize the best

position found by the swarm. Then, at each iteration, depending on the information it has, the particle adjusts its position to a new optimal one. The process is repeated until the swarm reaches the global optimum.

Throughout the years, several authors have improved the standard algorithm to enhance its effectiveness. Moreover, numerous variants have appeared, some of which are the chaotic PSO (CPSO) (Hefny and Azab, 2010) integrating concepts from chaos theory, the fuzzy PSO (FPSO) (Abdelbar et al., 2005) combining PSO and fuzzy theory sets, or the bare-bones PSO (BBPSO) (Kennedy, 2003). We can also find hybrid PSO algorithms, that is to say PSO algorithms combined with other optimization methods, and PSO extensions.

The aim of this paper is to study the effectiveness of the PSO algorithm applied to complex functions. Note that this paper will only focus on minimizing functions.

First, we will present the standard PSO algorithm, and apply it to several unconstrained functions. We will post-validate the results found by using the Newton-Raphson method. Secondly, we will present the extended algorithm, applied to a single-objective constrained optimization. Finally, we will do the same with a multi-objective constrained optimization.

1.1 The parameters selection : overview and explanation

This part aims at presenting the standard version of the PSO algorithm in a theoretical way. To do so, we will rely on the available literature. We will introduce the parameters and fundamental equations.

1.1.1 The acceleration coefficients

Denoted c_1 and c_2, two constants that represent the acceleration coefficients, they can be non-constant in some cases depending on the optimization problem. c_1 is the 'cognitive' weight and c_2 is the 'social' weight; they control how much the best global and individual positions should influence the velocity and trajectory of the particle. c_1 and c_2 can take different values, but Shi and Eberhart, 1998 advise that c_1+c_2 must be less or equal to 4. There are thus 3 cases.

- $c_1 > c_2$: the particles would trust themselves more than the group and would converge more towards their *pbest* than their *gbest* and would therefore waste too much time exploring a small non-optimal region.

- $c_1 < c_2$: the particles would follow the swarm more without searching enough in their region and would thus be attracted to their *gbest* rather than their *pbest*.

- $c_1 = c_2$: In the basic PSO algorithm, c_1 and c_2 are equal to 2, so 1 each. This corresponds to the case where the particles are attracted in the same way to the *pbest* and their *gbest*. This is the case that the authors advise.

Let's estimate the best choice of c_1 and c_2 for the Eggholder function :

$\forall$ (x,y) $\in$ [-512 ; 512],

$$f(x,y) = -(y+47)\sin\sqrt{|\frac{x}{2}+(y+47)|} - x\sin\sqrt{|x-(y+47)|} \tag{1.1}$$

With the help of a heatmap, we will see which combination of c_1 and c_2 is optimal.

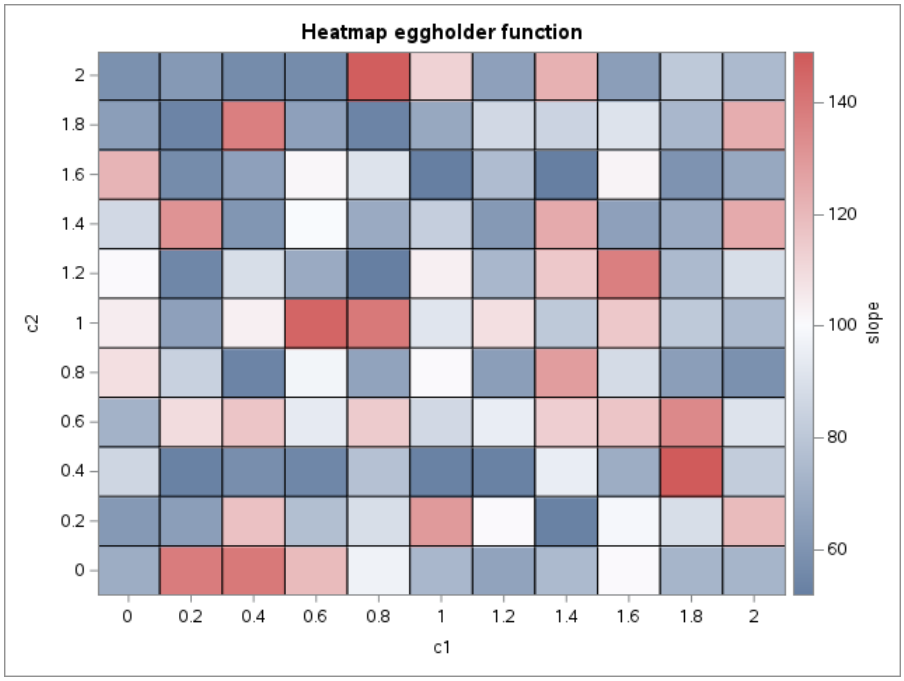

Figure 1.1: Heatmap of Eggholder function

'Slope' is the number of iterations needed for the algorithm to reach the global minimum.

Here are the results of our PSO algorithm :

The Global Minimum of Eggholder function on [-512 ; 512] is :

Gbestx	
512	404.23181

Gbesty
-959.6407

With 50 iterations, $c_1 = 1.4$ and $c_2 = 1.6$, the algorithm finds the right global minimum.

1.1.2 The maximum velocity

According to Russell Eberhart, *Vmax* is the only parameter that should be determined by the user. It is necessary to set a *Vmax* to avoid any overflow. There are two cases :

• If *Vmax* takes a large value, it allows a global exploration.

• If *Vmax* takes a small value, it allows a local and intensive exploration (Saptarshi Sengupta, 2018).

1.1.3 The inertia weight

The inertia factor of a particle refers to a parameter that controls the search capability at the local or global level. It determines how much the previous velocity has an influence on the motion of the current particle. A large value of w (higher than 1) allows one to search in a larger space with a larger range of motion and thus to do a so-called "global" exploration. In this case, more importance is given to the self-knowledge of the particle than to the swarm's knowledge. A small value of w (lower than 1) increases the search speed of the particles and is synonymous with a small amplitude of movement and therefore local exploration, but it prevents the algorithm from converging to a local optimum. The choice of this factor will depend on the problem at hand and the trade-off between local and global exploration. Van Den Bergh et al., 2007 has established a strong relationship between c_1, c_2 and w which can be modeled by the following inequality:

$$w > \frac{1}{2}(c_1 + c_2) - 1 \tag{1.2}$$

w should be in the range [0.9, 1.2] but can also be a function of time or even a random number. There is no mathematical model that dynamically adjusts the inertia factor, but there is a negative linear relationship (R. C. Eberhart and Shi, 2000 Xin et al., 2009) as a function of time between w_{max}=0.9 and w_{min}=0.4 :

$$w_t = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} * t \tag{1.3}$$

Good results were found for a linearly decreasing value from 0.9 to 0.4.

Chatterjee and Siarry, 2006 suggested that the inertia factor can be modified with a nonlinear function of time as :

$$w_t = \left(\frac{t_{max} - t}{t_{max}}\right)^n * (w_{min} - w_{max}) + w_{max} \tag{1.4}$$

where t_{max} is the maximum number of iterations and n is the nonlinear modulation index chosen by the user. According to Chatterjee and Siarry, n should be in the range [0.9, 1.3].

We have chosen to stick to the inertia factor stated in the original article (equation 1.4).

1.1.4 Maximum of particles and iterations

It is necessary to define the number of particles at the beginning to solve the problem. According to Saptarshi Sengupta, 2018 and experience, a large number of particles can only lead to better results but this is not done without conditions. Indeed, it depends on two parameters, the size of the search space or the definition domain of the problem, and the ratio between the computing capacity of the machine and the maximum search time. Same goes for the iterations; a high number of iterations allows the PSO algorithm to converge with more 'chances', because it has more trials, but it also requires more power and time from the machine.

1.1.5 Abbreviation and mathematical notation

Here are the important mathematical abbreviations and notations in table form:

Parameters	Explanation
w	Inertia weight parameter used to compute the velocity of each particle
c_2	Cognitive real acceleration coefficient used to compute the particle's velocity
c_2	Social real acceleration coefficient used to compute the particle's velocity
t_{max}	the maximum number of iterations
n	the non-linear modulation index chosen by the user
$Vmax$	The maximum velocity
w_{max}	The maximum inertia weight
w_{min}	The minimum inertia weight
V	Particle's velocity
gbest	Global best position of a particle in the swarm
pbesti	Personal best position of particle i
Xi	The position of particle i
r_1	Cognitive uniformly distributed random vector used to compute the particle's velocity
r_2	Social uniformly distributed random vector used to compute the particle's velocity
gbesty	It's the global best, which is the current best position found by the whole group
d	Number of dimensions of the search space
xmax	Upper limit of the dimension d in the search space
xmin	Lower limit of the dimension d in the search space
g	Global best position of a particle in the swarm
pclmax	Maximum of particles

1.2 Fundamental equations

At the end of an iteration, each particle has memorized its optimal position so far and also the optimal position found by the swarm. Thanks to this information, it adjusts its new position for the next iteration. In our algorithm, we denote X_i the current position of the particle. We denote *pbest* the best position found so far by the particle (*pbestx* and *pbesty*), and *gbest* the global best position found so far by the swarm (found by one particle within the swarm, *gbestx* and *gbesty*).

The velocity of the particle is given as follows :

$$V_i = wV_i + c_1r_1(pbestx_i - X_i) + c_2r_2(gbestx_i - X_i) \tag{1.5}$$

where r_1 and r_2 are random uniform parameters between 0 and 1 that reset at every iteration. Some authors have shown that the presence of an uniform variable improves the performance of the PSO algorithm by avoiding early convergence and thus maximizing chances to get to the global optima.

Three parts contribute to the velocity. First, it is influenced by the inertia, $w*V_i$ where w is the weight. The intuition behind this part is simply that the particle moves in the same direction with the same speed. This term prevents the particle from changing its position to a completely diverging path and thus sticking to the current direction. This is why it is seen as a momentum. The second term, $c_1r_1(pbestx_i - X_i)$, is the personal or cognitive component. It enhances the particle's performance, particularly thanks to the parameter c_1, the cognitive constant presented before. It allows it to return to its best position found so far ($pbest$). Finally, it is also influenced by the social component, $c_2r_2(gbestx_i - X_i)$: the particle is in fact also attracted by the global solution found so far by the swarm ($gbest$).

The new position of the particle is updated as follows :

$$X_i = X_i + V_i \tag{1.6}$$

where V_i is the new velocity computed before.

1.3 PSO algorithm's flowchart

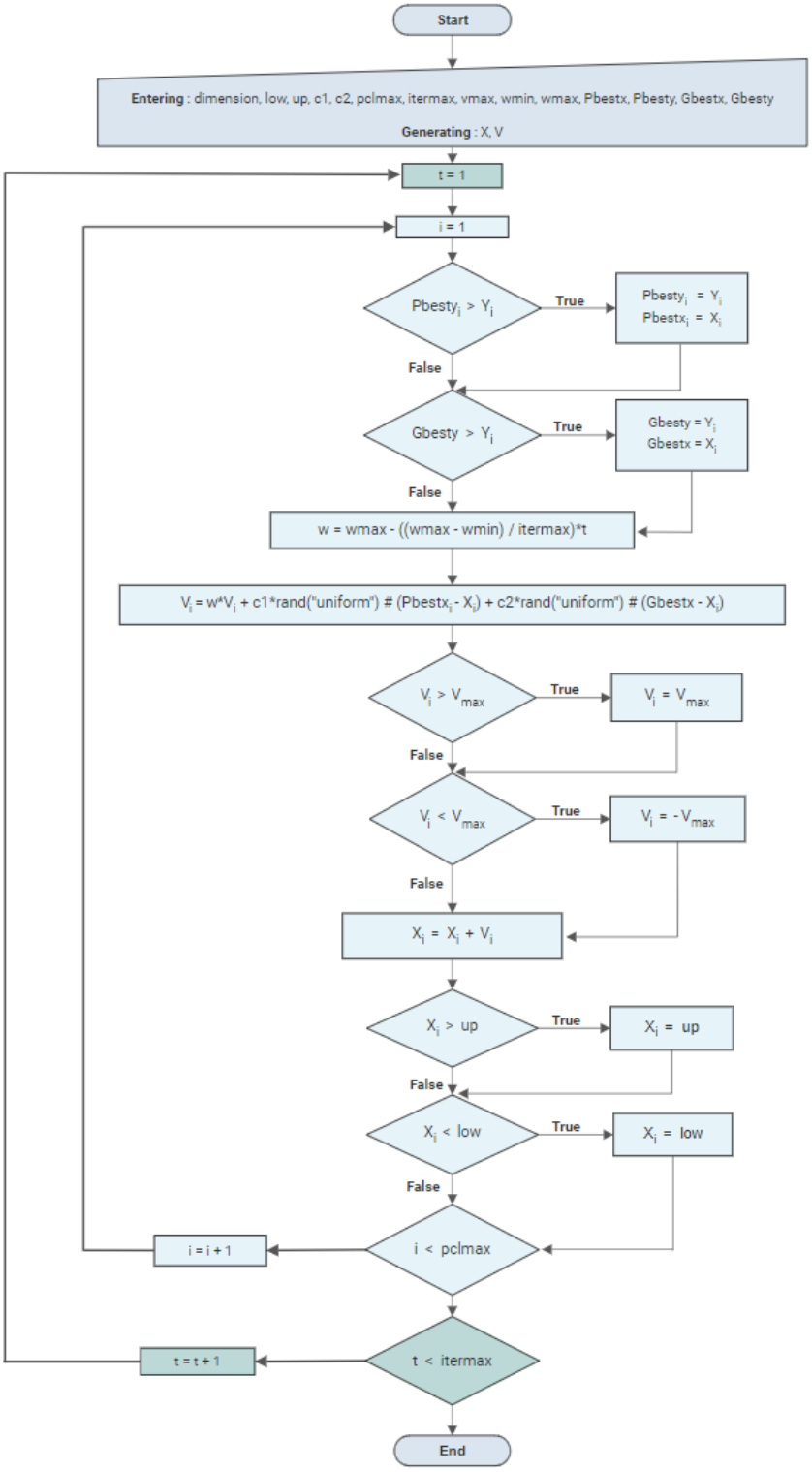

Figure 1.2: PSO algorithm's flowchart

Chapter 2

PSO algorithm : an unconstrained single-objective optimization

2.1 Eggholder

The Eggholder function is a multimodal non convex and non linear two-dimensional function with a large number of local minima and a global minimum.
This function has one global minima **f(x, y) = -959.6407** found at **x = (512, 404.2319)**.

2.1.1 Characteristics of the function

$\forall$ (x,y) $\in$ [-512 ; 512],

$$f(x,y) = -(y+47)\sin\sqrt{|\frac{x}{2}+(y+47)|} - x\sin\sqrt{|x-(y+47)|} \tag{2.1}$$

On the following graphic representation of the function, we can see multiple peaks. On the first graph, we can't see the global minimum - its hidden by other ones. But on the second graph, we can see the global minimum at the point (512,404).

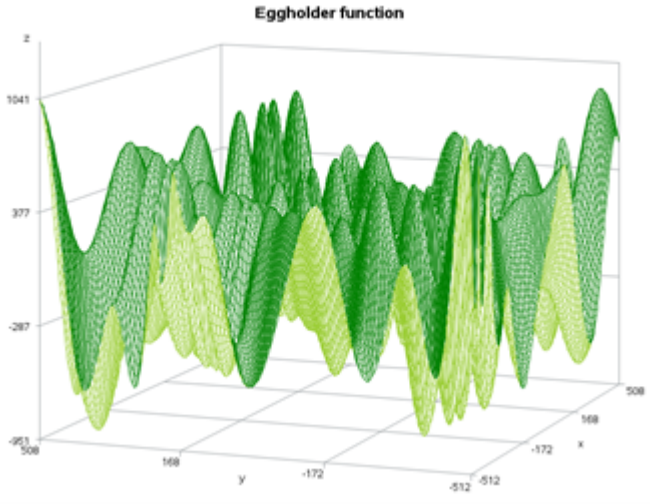

Figure 2.1: 3D chart of Eggholder function

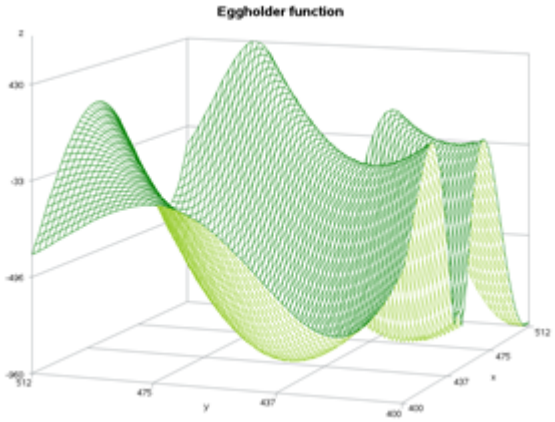

Figure 2.2: 3D chart of Eggholder function, zoomed in the global minima area

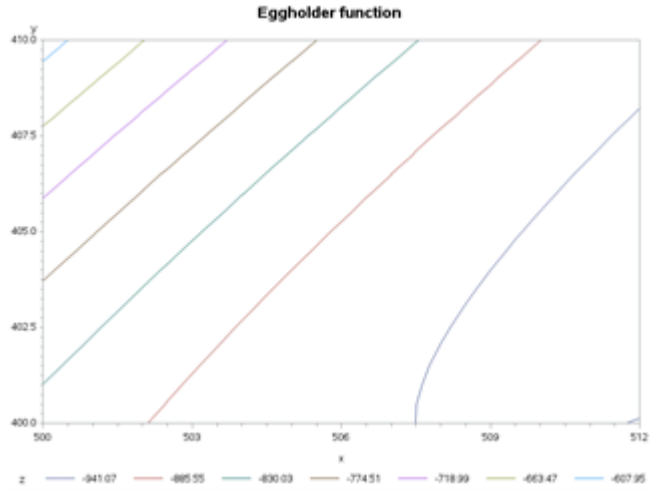

Figure 2.3: Contour lines of Eggholder function and global minimum

2.1.2 PSO results

Here are the results of our PSO algorithm :

The Global Minimum of Eggholder function on [-512 ; 512] is :

Gbestx	
512	404.23181

Gbesty
-959.6407

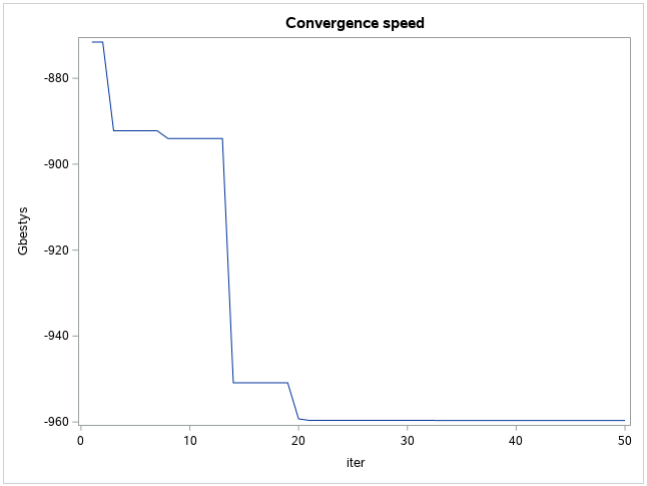

Figure 2.4: Convergence speed of the algorithm for Eggholder function

2.1.3 Post-validation

We use the Newton-Raphson method to post-validate our results.
We notice that the gradient did not cancel and that the Newton-Raphson method made 5 iterations. It did not cancel the function objective gradient because it cannot leave the upper bound. In this paper, all post-validations have been made thanks to the NLPQN method provided by SAS. We have selected the "Original Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the inverse Hessian matrix" in the option vector. The global minimum is reached at the point (512,404.2317) and is value is -959.64.

2.2 Ackley

The Ackley function is also a non convex and non linear two-dimensional function with a large number of local minima and a global minimum.
This function has one global minima $\mathbf{f(x, y) = 0}$ found at $\mathbf{x = (0, 0)}$.

2.2.1 Characteristics of the function

$\forall$ (x,y) $\in$ [-5 ; 5],

$$f(x,y) = -20\exp[-0.2\sqrt{0.5(x^2+y^2)}] - \exp[0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20 \qquad (2.2)$$

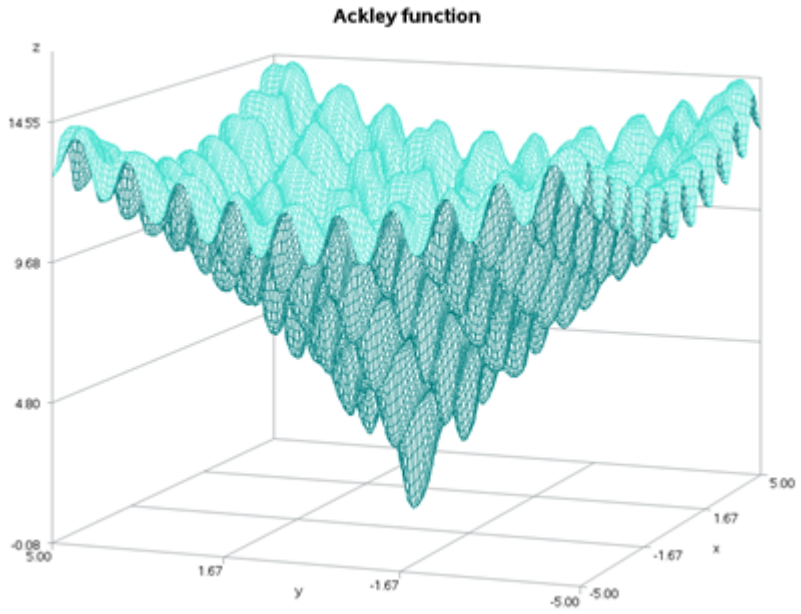

Figure 2.5: 3D chart of Ackley function

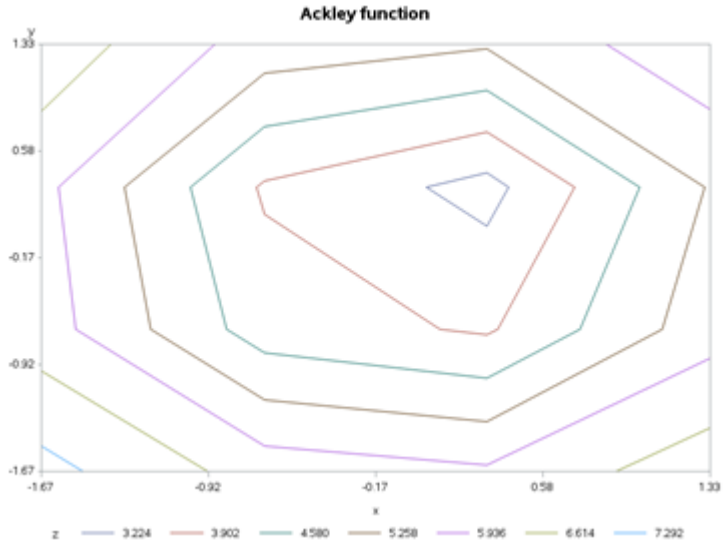

Figure 2.6: Contour lines of Ackley function and global minimum

2.2.2 PSO results

The Global Minimum of Ackley function on [-5 ; 5] is :

Gbestx	
-5.275E-6	0.0000111

Gbesty
0.0000347

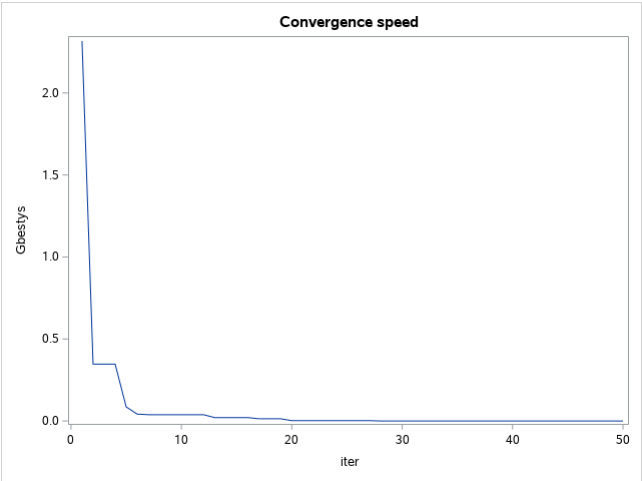

Figure 2.7: Convergence speed of the algorithm for Ackley function

2.2.3 Post-validation

The Newton-Raphson method efficiently finds the global minimum of the Ackley function with only 1 iterations and 34 function calls. The global minimum is reached at the point (0,0) and is value is 0.

2.3 Bukin

The sixth Bukin function is also a non convex and non linear two-dimensional function with a large number of local minima and a global minimum.
This function has one global minima $\mathbf{f(x, y) = 0}$ found at $\mathbf{x = (-10, 1)}$.

2.3.1 Characteristics of the function

$\forall$ x $\in$ [-15 ; 5] and $\forall$ y $\in$ [-3 ; 3]

$$f(x, y) = 100\sqrt{|y - 0.01x^2|} + 0.01|x + 10| \tag{2.3}$$

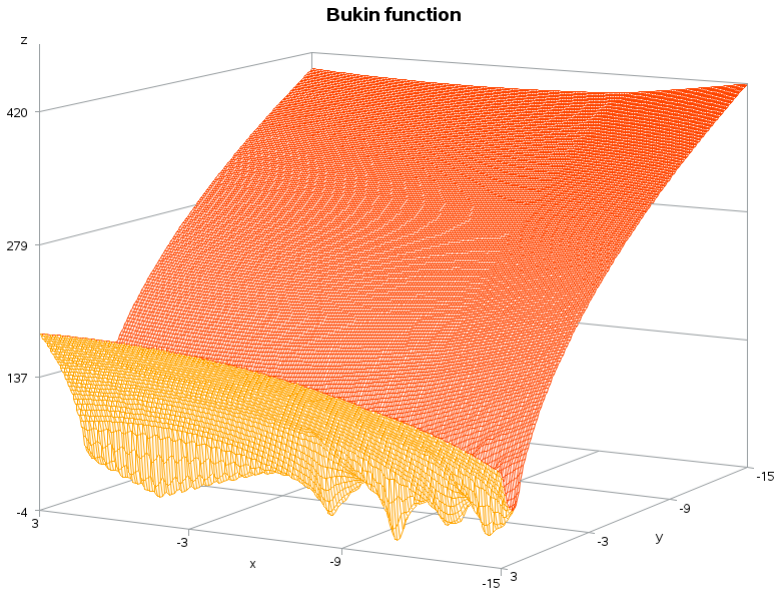

Figure 2.8: 3D chart of Bukin function

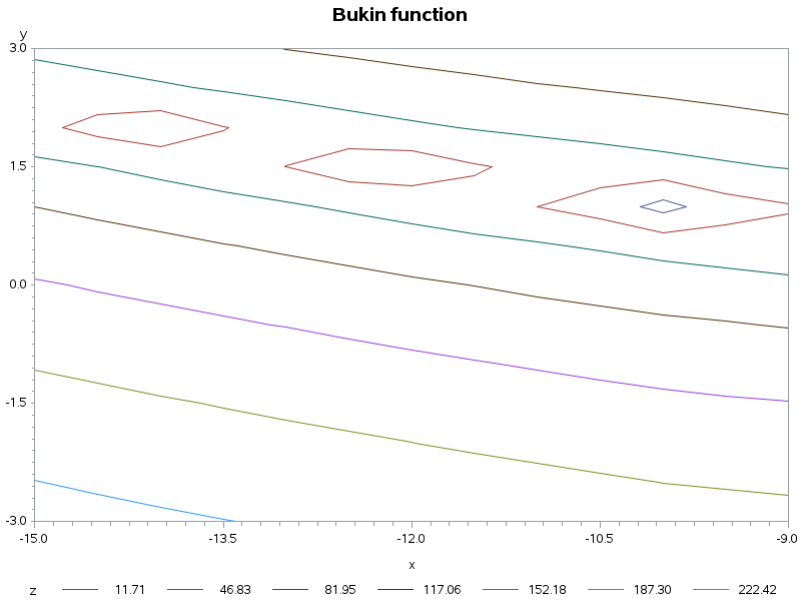

Figure 2.9: Contour lines of Bukin function and global minimum

2.3.2 PSO results

The Global Minimum of Bukin function on [-15 ; 3] is :

Gbestx	
-9.998459	0.9996918

Gbesty
0.0171347

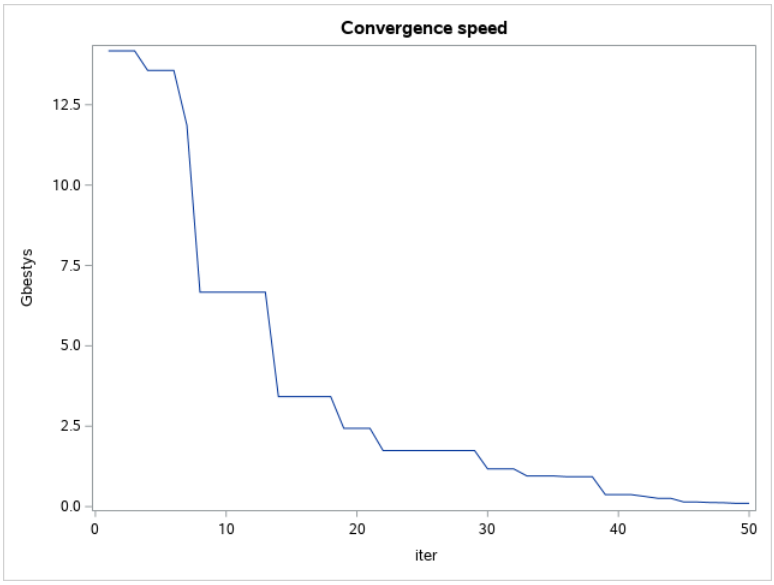

Figure 2.10: Convergence speed of the algorithm for Bukin function

2.3.3 Post-validation

The call has been launched with a random x vector and it had difficulty in finding the global minimum because it did 11 iterations and 95 function calls. The global minimum is reached at the point (-10,1) and is value is 0.

Chapter 3

Single Constrained Objective Particle Swarm Optimisation (SCOPSO)

3.1 Introducing a constraint

In any optimization problem under constraint, the constraint reduces the domain definition of the objective function. Therefore, the simple version of the Particle Swarm Algorithm is also able to resolve this type of problem - we only need to give it the bounds of the definition domain under constraint.

3.1.1 Consequences on the choice of the parameters

Since the introduction of one or more constraints reduces the definition domain, the search domain of the particles is also reduced. Depending on the shape of the objective function, it is then necessary to reduce the speed of the particles' movement or to increase the inertia weight so that they do not fail to visit a local minimum which could be a global one.

3.1.2 PSO's flowchart under constraint

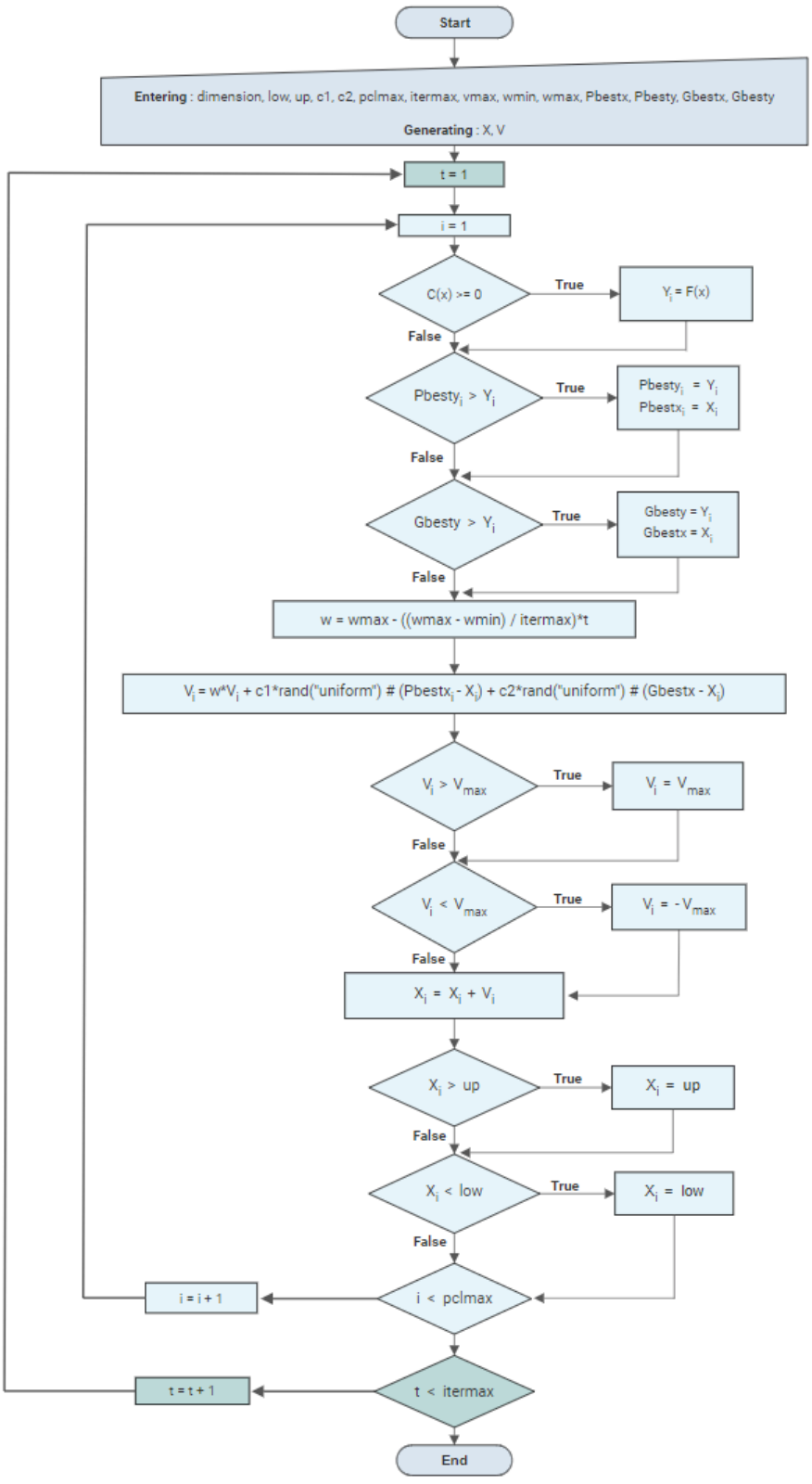

Figure 3.1: Flowchart of PSO algorithm under constraint

3.2 Applications

3.2.1 Rosenbrock function constrained to a disk

The Rosenbrock function constrained to a disk is a two-dimensional non linear function with a non linear constraint.

Characteristics

$\forall$ (x,y) $\in$ [-1.5 ; -1.5],

$$Minimize: \quad f(x,y) = (1-x)^2 + 100(y-x^2)^2 \tag{3.1}$$

$$Subject\ to: \quad x^2 + y^2 \leqslant 2 \tag{3.2}$$

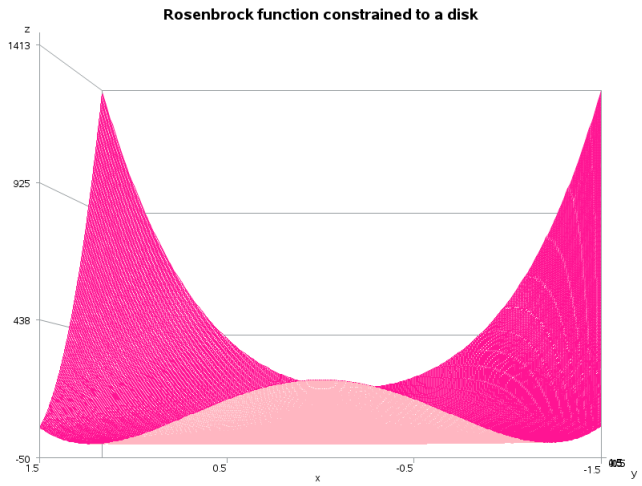

Figure 3.2: 3D chart of Rosenbrock function constrained to a disk (1)

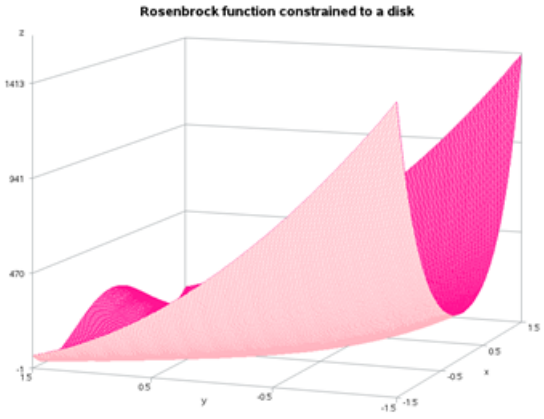

Figure 3.3: 3D chart of Rosenbrock function constrained to a disk (2)

PSO's results

The Global Minimum of Rosenbrock function constrained to a disk on [-1.5 ; 1.5] is :

Gbestx	
0.999947	0.9998861

Gbesty
8.9782E-9

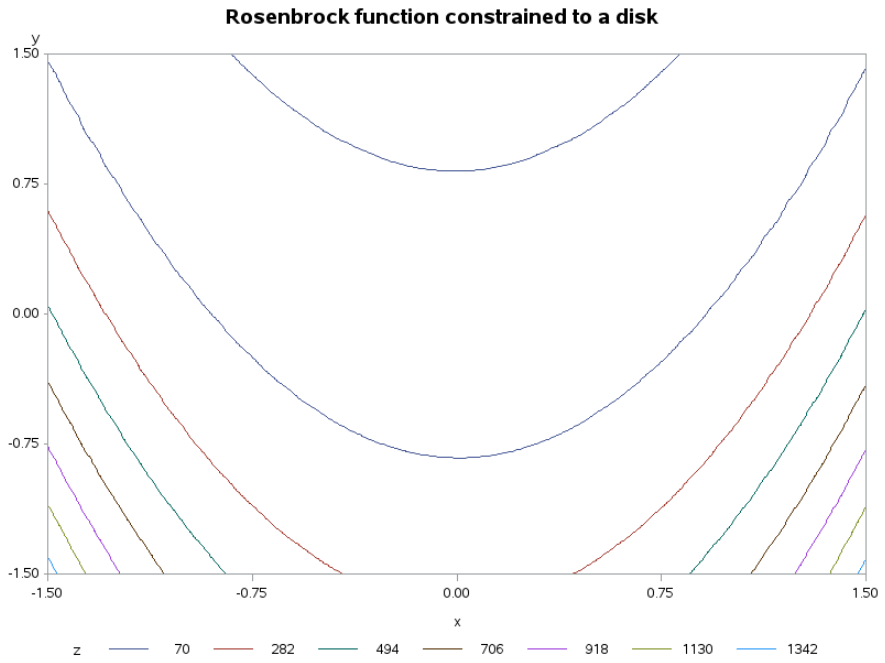

Figure 3.4: Contour lines of Rosenbrock constrained to a disk function and global minimum

Post-validation

The call has efficiently found the global minimum - it is reached at the point (0.99,0.99) which is equivalent to the real one (1,1) and is value is 0.

3.2.2 Gomez and Levy's function

Characteristics

The Gomez and Levy's function is a two-dimensional function.

$\forall$ (x) $\in$ [-1 ; 0.75], $\forall$ y $\in$ [-1 ; 1],

$$Minimize: \quad f(x,y) = 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4 \tag{3.3}$$

$$Subject\ to: \quad -\sin(4\pi x) + 2\sin^2(2\pi y) \leqslant 1.5 \tag{3.4}$$

Here's the graphic representation made by Simionescu, 2020

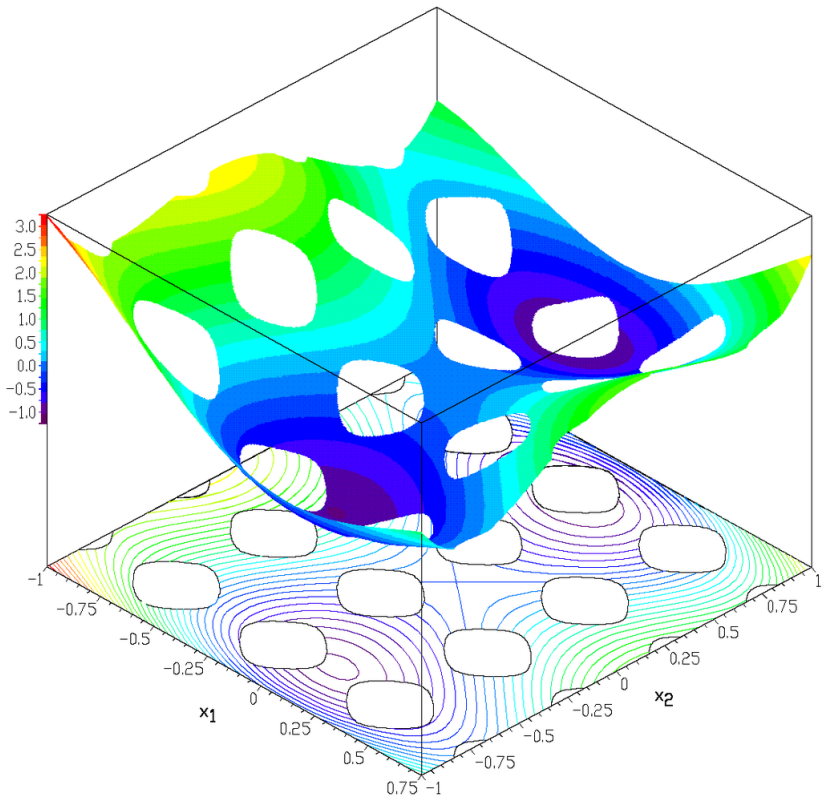

Figure 3.5: 3D chart of Gomez and Levy's function

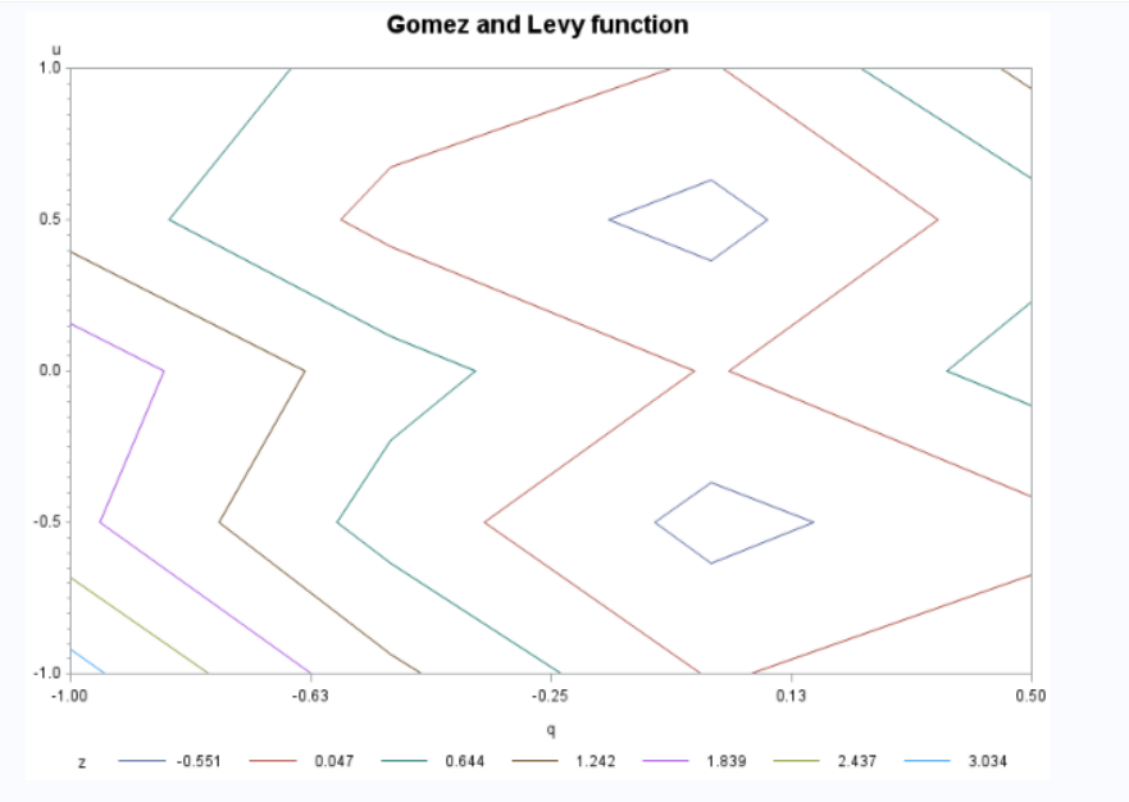

Figure 3.6: Contour lines of Gomez and Levy function

PSO's results

The Global Minimum of Gomez and Levy's function is :

Gbestx	
-0.089841	0.7126551

Gbesty
-1.031628

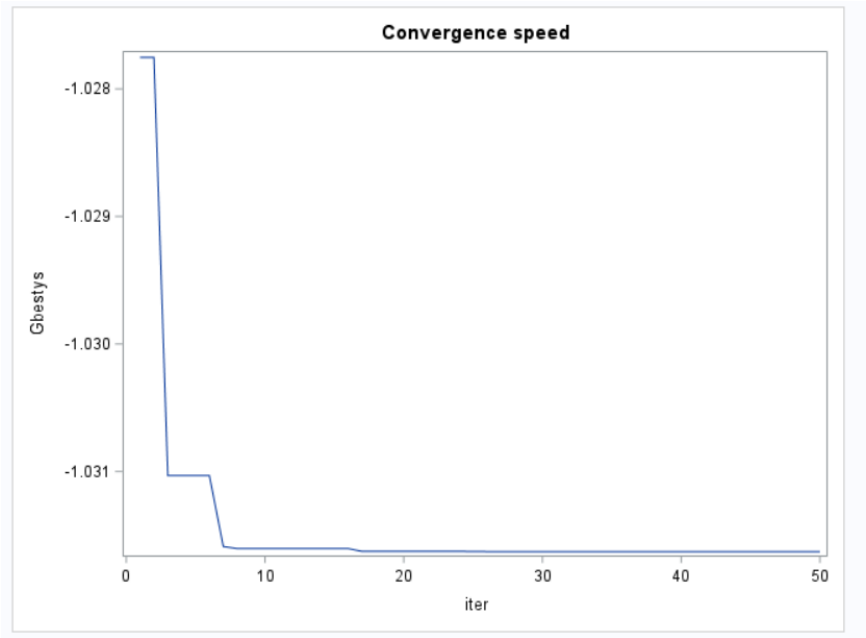

Figure 3.7: Convergence speed of the algorithm for Gomez and Levy's function

Post-validation

The call has efficiently found the global minimum - it is reached at the point (0.089,-0.71) and is value is -1.031.

Chapter 4

Multiple Constrained Objective Particle Swarm Optimisation (MCOPSO)

4.1 Introduction

Optimization problems with multiple objectives and constraints are increasingly numerous, particularly in engineering and logistics. The PSO algorithm has long been put aside in favor of the Differential Evolution (DE) algorithm - but he is strongly returning.

According to Kachitvichyanukul, 2012 the PSO algorithm is a better choice than the DE algorithm when we face a low-dimension problem because of its ability to find good solutions faster. Even under constraints it stays very efficient because it works on the influence of best solution on population while having the ability to reach good solution without local search.

4.1.1 Algorithm's structure

Only few modifications are necessary on the single constrained objective PSO's flowchart to include more objectives functions :

- When the function objective is defined, we have to define as many Y_i as there are objective functions. Contrary to the constraints, which can be defined in a module, we must have to define Y_i separately.
- We also need as many *pbesty* as there are objective functions.
- In order to draw the Pareto front, we need to gather all the *gbesty* for each function by creating as many vectors as there are objective functions.

4.1.2 Optimality criteria

A solution is Pareto optimal when it minimizes an objective function whom the optimization cannot be improved without hurting others objective functions. In a multi-objective optimization, the plurality of objective functions often implies a plurality of Pareto optimal solutions. These Pareto optimal solutions form a Pareto frontier.

4.1.3 Application

We have chosen the Chankong and Haimes, 2008 optimization problem.

$$Minimize : \begin{cases} f_1(x,y) = 2 + (x-2)^2 + (y-1)^2 \\ f_2(x,y) = 9x - (y-1)^2 \end{cases} \tag{4.1}$$

$$Subject\ \ to : \begin{cases} g_1(x,y) = x^2 + y^2 \leqslant 225 \\ g_2(x,y) = x - 3y + 10 \leqslant 0 \end{cases} \tag{4.2}$$

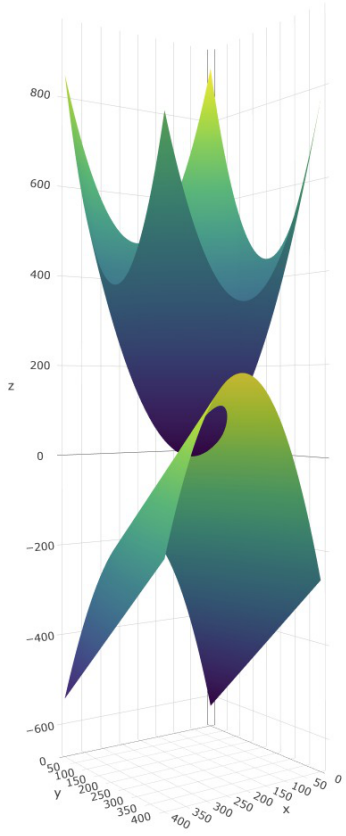

Figure 4.1: 3D first chart of Chankong and Haimes function

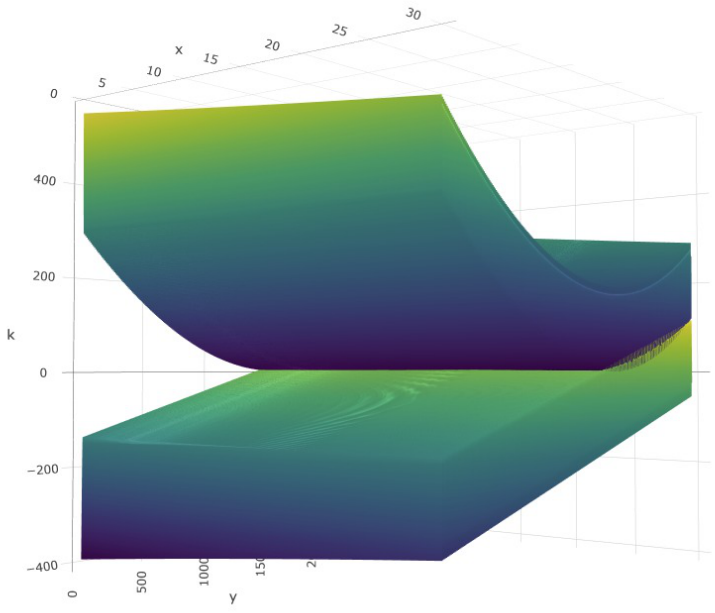

Figure 4.2: 3D second chart of Chankong and Haimes function

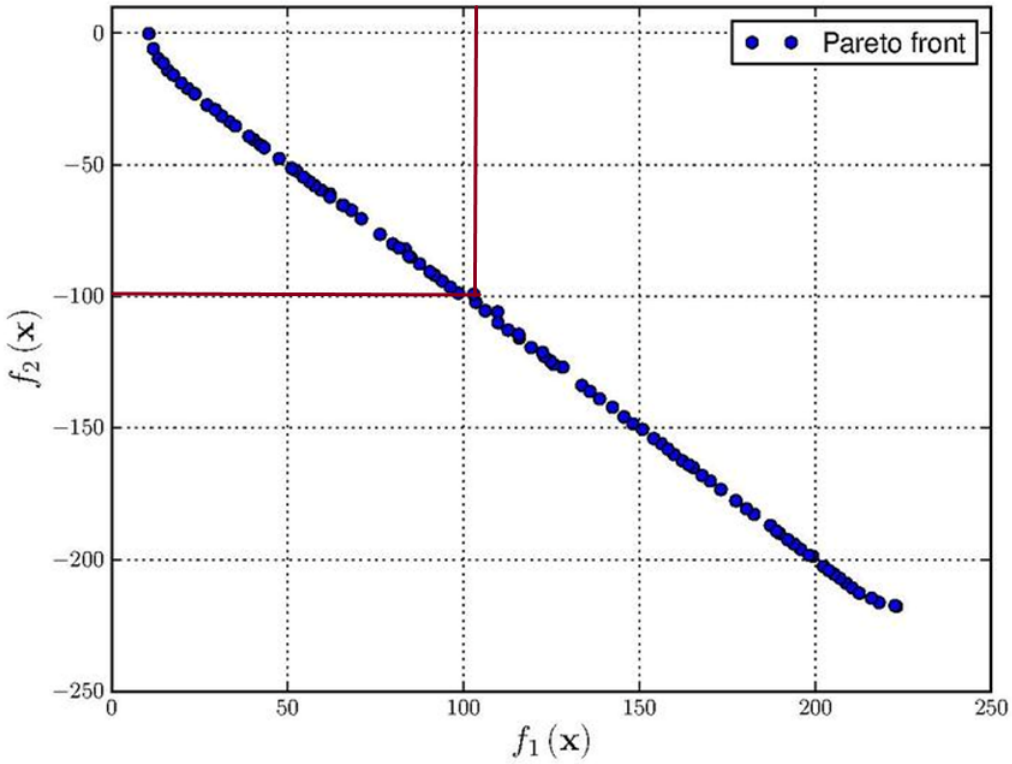

Figure 4.3: Chankong and Haimes' Pareto frontier

Here's the result of our algorithm :

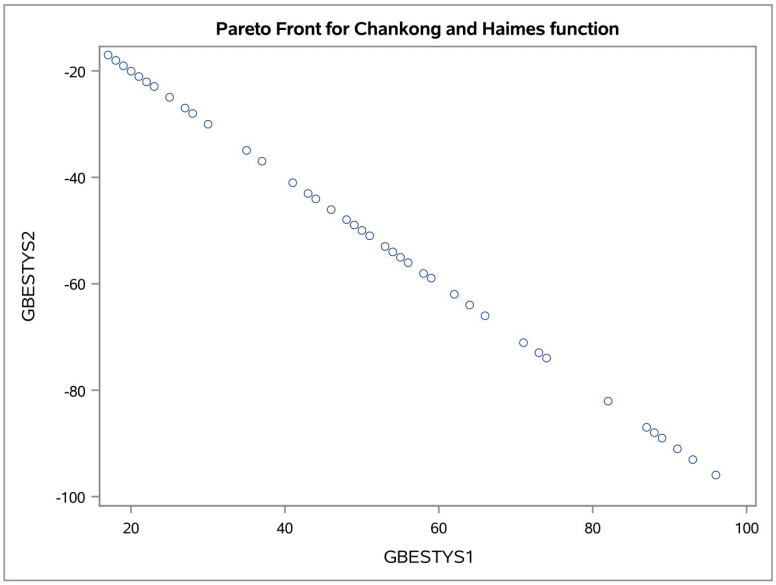

Figure 4.4: Our Pareto frontier

Our algorithm reduces the Pareto frontier proposed by Chankong and Haimes - this is good because it reduces the set of Pareto optimal solutions and it facilitates decision-making.

Chapter 5

Conclusion

This paper aimed at testing the effectiveness of our PSO algorithm. To do so, we applied the algorithm through a range of different types of functions : unconstrained functions, single constrained functions and bi-objective constrained functions. Through these tests, we can conclude that our Particle Swarm Optimization algorithm, coded under SAS, performs very well.

In Chapter 2, we tested the effectiveness of the basic PSO algorithm on three functions : Eggholder, Ackley and Bukin, three functions that are highly non-linear and are non-convex. In each case, the algorithm finds the global minima within a short period of time. We post-validated our results thanks to another algorithm, Newton-Raphsen, already built in SAS (NLPQN procedure).

In Chapter 3, we extended our algorithm to support optimization under constraints. We tested it on two constrained functions, and it also performed very well.

Finally, in Chapter 4, we went even deeper by enhancing the algorithm in order to support constrained bi-objective functions. Since there are several Pareto optimal solutions, the objective of the algorithm here was to reduce the number of solutions, which it did.

References

Abdelbar, A., Abdelshahid, S., & Wunsch, D. (2005). Fuzzy pso: A generalization of particle swarm optimization. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *2*, 1086–1091 vol. 2. https://doi.org/10.1109/IJCNN.2005.1556004

Chankong, V., & Haimes, Y. Y. (2008). *Multiobjective decision making: Theory and methodology*. Courier Dover Publications.

Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & operations research*, *33*(3), 859–871.

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, *1*, 84–88.

Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. *Proceedings of the IEEE international conference on neural networks*, *4*, 1942–1948.

Hefny, H. A., & Azab, S. S. (2010). Chaotic particle swarm optimization. *2010 The 7th International Conference on Informatics and Systems (INFOS)*, 1–8.

Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: Ga, pso, and de. *Industrial Engineering and Management Systems*, *12*, 215–223. https://doi.org/10.7232/iems.2012.11.3.215

Kennedy, J. (2003). Bare bones particle swarms. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, 80–87. https://doi.org/10.1109/SIS.2003.1202251

Saptarshi Sengupta, S. B., Richard Alan Peters II. (2018). "Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives". *Machine Learning and Knowledge Extraction*, *1*.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, 69–73.

Simionescu, P. A. (2020). A collection of bivariate nonlinear optimisation test problems with graphical representations. *International Journal of Mathematical Modelling and Numerical Optimisation*, *10*(4), 365–398.

Van Den Bergh, F. et al. (2007). *An analysis of particle swarm optimizers* (Doctoral dissertation). University of Pretoria.

Xin, J., Chen, G., & Hai, Y. (2009). A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. *2009 International Joint Conference on Computational Sciences and Optimization*, *1*, 505–508.

Appendix 1

Newton-Raphson pour Eggholder

Optimization Start					
Paramètres estimés					
N	Paramètre	Estimation	Fonction objective gradient	Contrainte Borne inférieure	Contrainte Borne supérieure
1	X1	500.000000	-20.744759	-512.000000	512.000000
2	X2	400.000000	15.164084	-512.000000	512.000000

Parameter Estimates	2
Lower Bounds	2
Upper Bounds	2

Optimization Start			
Active Constraints	0	Objective Function	-846.5692074
Max Abs Gradient Element	20.744759459		

Optimization Results			
Iterations	5	Function Calls	11
Gradient Calls	10	Active Constraints	1
Objective Function	-959.6406627	Max Abs Gradient Element	3.7654471E-8
Slope of Search Direction	-4.97701E-8		

Optimization Results				
Paramètres estimés				
N	Paramètre	Estimation	Fonction objective gradient	Contrainte Borne active
1	X1	512.000000	-3.385559	Upper BC
2	X2	404.231795	-3.765447E-8	

Appendix 2

Newton-Raphson pour Ackley

Optimization Start			
Paramètres estimés			
N	Paramètre	Estimation	Fonction objective gradient
1	X1	0	2.828428
2	X2	0	2.828428

Parameter Estimates	2

Optimization Start			
Active Constraints	0	Objective Function	0
Max Abs Gradient Element	2.8284275532		

Optimization Results			
Iterations	1	Function Calls	34
Gradient Calls	3	Active Constraints	0
Objective Function	0	Max Abs Gradient Element	2.8284275532
Slope of Search Direction	-28.28427553		

Optimization Results			
Paramètres estimés			
N	Paramètre	Estimation	Fonction objective gradient
1	X1	-1.31612E-17	2.828428
2	X2	-1.31612E-17	2.828428

Appendix 3

Newton-Raphson pour Bukin

Optimization Start					
Paramètres estimés					
N	Paramètre	Estimation	Fonction objective gradient	Contrainte Borne inférieure	Contrainte Borne supérieure
1	X1	-10.400000	-20.810000	-15.000000	-5.000000
2	X2	-0.900000	-100.000000	-3.000000	3.000000

Parameter Estimates	2
Lower Bounds	2
Upper Bounds	2

Optimization Start			
Active Constraints	0	Objective Function	198.164
Max Abs Gradient Element	100.00000008		

Optimization Results			
Iterations	11	Function Calls	95
Gradient Calls	17	Active Constraints	0
Objective Function	0.0000138031	Max Abs Gradient Element	26.850276841
Slope of Search Direction	-142.5918459		

Optimization Results			
Paramètres estimés			
N	Paramètre	Estimation	Fonction objective gradient
1	X1	-10.001271	6.692439
2	X2	1.000254	26.850277

Appendix 4

Newton-Raphson pour Rosenbrock

Optimization Start				
Paramètres estimés				
N	Paramètre	Estimation	Fonction objective gradient	Fonction Lagrange gradient
1	X1	-1.500000	-454.999961	-302.499986
2	X2	1.500000	-149.999996	-302.499976

Parameter Estimates	2
Nonlinear Constraints	1

Optimization Start			
Objective Function	62.5	Maximum Constraint Violation	2.5
Maximum Gradient of the Lagran Func	302.49998606		

Optimization Results			
Iterations	39	Function Calls	46
Gradient Calls	41	Active Constraints	0
Objective Function	1.990091E-10	Maximum Constraint Violation	0
Maximum Projected Gradient	0.0000634804	Value Lagrange Function	1.990091E-10
Maximum Gradient of the Lagran Func	0.0000634804	Slope of Search Direction	-5.53298E-11

Optimization Results				
Paramètres estimés				
N	Paramètre	Estimation	Fonction objective gradient	Fonction Lagrange gradient
1	X1	0.999986	0.000063480	0.000063480
2	X2	0.999972	-0.000036752	-0.000036752

Appendix 5

Newton-Raphson pour Gomez and Levy

Optimization Start						
Paramètres estimés						
N	Paramètre	Estimation	Fonction objective gradient	Fonction Lagrange gradient	Contrainte Borne inférieure	Contrainte Borne supérieure
1	X1	0.230000	1.739085	1.739085	-1.000000	0.750000
2	X2	0	0.230000	0.230000	-1.000000	1.000000

Parameter Estimates	2
Lower Bounds	2
Upper Bounds	2
Nonlinear Constraints	1

Optimization Start			
Objective Function	0.2057726843	Maximum Constraint Violation	0
Maximum Gradient of the Lagran Func	1.7390845307		

Optimization Results				
Paramètres estimés				
N	Paramètre	Estimation	Fonction objective gradient	Fonction Lagrange gradient
1	X1	0.089868	0.000185	0.000185
2	X2	-0.712677	-0.000319	-0.000319