Hybridization of Differential evolution with Least-Square Support Vector Machines

Vitaliy Feoktistov, Stefan Janaqi

Hybridization of Differential Evolution with Least-Square Support Vector Machine

Vitaliy Feoktistov and Stefan Janaqi
Centre de Recherche LGI2P - Site EERIE
l'École des Mines d'Alès
Vitaliy.Feoktistov@ema.fr, Stefan.Janaqi@ema.fr

1 Introduction

In this article we give a hybrid method for choosing "good" individuals for the next generation of Differential Evolution (DE). Keeping DE to work as is, we add the Least-Square Support Vector Machine (LS-SVM) approximation in the end of each generation cycle. Such approximation uses a subset of selected individuals of the population. As a SVM core we choose a second order polynomial kernel function. The next individual is the optimum of the SVM approximation function which can be computed analytically as a solution of a system of linear equations. This method leads us to the improvement of algorithm convergence.

2 Differential Evolution

Differential Evolution is a recently invented global optimization technique (Storn and Price, 1995). It can be classified as an *iterative stochastic method*. Enlarging the Evolutionary Algorithms' group, DE turns out to be one of the best *population-based* optimizers (Storn and Price, 1996). In the following lines we give a brief description of DE algorithm.

An optimization problem is represented by a set of variables. Let these variables form a D-dimensional vector in continuous space $X = (x_1, \ldots, x_D) \in \mathbb{R}^D$. And let there is some criterion of optimization $f(X) : \mathbb{R}^D \to \mathbb{R}$, usually named either *fitness* or *cost* function. Then, the goal of optimization is to find the values of the variables that minimize the criterion, i.e. to find

$$X^* : \ f(X^*) = \min_X f(X) \tag{1}$$

Often, the variables satisfy boundary constraints

$$L \le X \le H \ : \quad L, H \in \mathbb{R}^D \tag{2}$$

As all Evolutionary Algorithms, DE deals with a *population* of solutions. The population $\mathbb{P}$ of a generation g has NP vectors, so-called *individuals* of population. Each such individual represents a potential optimal solution.

$$\mathbb{P}^g = X_i^g, \quad i = 1, \ldots, NP \tag{3}$$

In turn, the individual contains D variables, so called *genes*.

$$X_i^g = x_{i,j}^g, \quad j = 1, \ldots, D \tag{4}$$

The population is *initialized* by randomly generating individuals within the boundary constraints,

$$\mathbb{P}^0 = x_{i,j}^0 = rand_{i,j} \cdot (h_j - l_j) + l_j \tag{5}$$

where *rand* function generates values uniformly in the interval $[0, 1]$.

Then, for each *generation* the individuals of a population are updated by means of a *reproduction* scheme. Thereto for each individual *ind* a set of other individuals π is randomly extracted from a population. To produce a new one the operations of *Differentiation* and *Recombination* are applied one after another. Next, the *Selection* is used to choose the best. Now briefly consider these operations.

Here, we show a typical model of the *Differentiation*, others can be found in (Storn and Price, 1995). For that three different individuals $\pi = \{\xi_1, \xi_2, \xi_3\}$ are randomly extracted from a population. So, the result, a *trial* individual, is

$$\tau = \xi_3 + F \cdot (\xi_2 - \xi_1) \,, \tag{6}$$

where $F > 0$ is the *constant of differentiation*.

After, the trial individual τ is recombined with updated one *ind*. The *Recombination* represents a typical case of a genes' exchange. The

trial one inherits genes with some probability. Thus,

$$\omega_j = \begin{cases} \tau_j & \text{if} \quad rand_j < Cr \\ ind_j & \text{otherwise} \end{cases} \tag{7}$$

where $j = 1, \ldots, D$ and $Cr \in [0, 1)$ is the *constant of recombination*.

The *Selection* is realized by comparing the cost function values of updated and trial individuals. If the trial individual better minimizes the cost function, then it replaces the updated one.

$$ind = \begin{cases} \omega & \text{if} \quad f(\omega) \leq f(ind) \\ ind & \text{otherwise} \end{cases} \tag{8}$$

Notice that there are only three control parameters in this algorithm. These are NP – population size, F and Cr – constants of differentiation and recombination accordingly. As for the terminal conditions, one can either fix the number of generations g_{max} or a desirable precision of a solution VTR (*value to reach*).

The pattern of DE algorithm is presented in the following way :

Algorithm 1 Differential Evolution

Require: F, Cr, NP – control parameters
  initialize $\mathbb{P}^0 \leftarrow \{ind_1, \ldots, ind_{NP}\}$
  evaluate $f(\mathbb{P}^0)$
  while (stop condition) **do**
    for all $ind \in \mathbb{P}^g$ **do**
      $\mathbb{P}^g \rightarrow \pi = \{\xi_1, \xi_2, \ldots, \xi_n\}$
        $\tau \leftarrow Differentiate(\pi, F)$
        $\omega \leftarrow Recombine(\tau, Cr)$
      $ind \leftarrow Select(\omega, ind)$
      $g \leftarrow g + 1$
    end for
  end while

3 Least Squares Support Vector Machine method

Support vector machine (SVM) was proposed as a method of classification and nonlinear function estimation (Vapnik, 1995). The idea is to map data into higher dimensional space, where an optimal separating hyperplane can be easily constructed. The mapping fulfils by means of kernel functions, which is constructed applying the Mercer's condition. The principal examples of kernels are polynomials, radial basis functions, multilayer perceptrons and others. In comparison with neural network methods SVM's gives a global solution obtained from resolving a quadratic programming problem, whereas those techniques suffer from the existence of many local minima.

The *least squares* version of SVM's (LS-SVM) has been recently introduced (Suykens et al., 2002; Suykens and Vandewalle, 1999). There, the solution is found by solving a linear system of equations instead of quadratic programming. It results from using equality constraints in place of inequality ones. Such linear systems were named as *Karush-Kuhn-Tucker* (KKT) or *augmented* systems.

Nevertheless, it remains the problem of matrix storage for large-scale tasks. In order to avoid it an iterative solution based on the conjugate gradient method has been proposed. Its computational complexity is $O(r^2)$, where r is rank of matrix. We mention briefly LS-SVM's applied to function approximation problem.

The LS-SVM model is represented in the feature space as

$$y(X) = \langle \upsilon, \phi(X) \rangle + b \,, \tag{9}$$

with $X \in \mathbb{R}^D$, $y \in \mathbb{R}$ and $\phi(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^{n_h}$ is a nonlinear mapping to higher dimensional feature space.

For given training set $\{X_k, y_k\}_{k=1}^n$ the optimization problem is formulated as

$$\min_{\upsilon, \epsilon} \Upsilon(\upsilon, \epsilon) = \frac{1}{2} \langle \upsilon, \upsilon \rangle + \gamma \frac{1}{2} \sum_{k=1}^{n} \epsilon_k^2 \tag{10}$$

subject to equality constraints,

$$y_k = \langle \upsilon, \phi(X_k) \rangle + b + \epsilon_k \tag{11}$$

where $k = 1, \ldots, n$.

So Lagrangian is

$$L = \Upsilon - \sum_{k=1}^{n} \alpha_k \{ \langle \upsilon, \phi(X_k) \rangle + b + \epsilon_k - y_k \} \tag{12}$$

where α_k are Lagrange multipliers.

By applying the Karush-Kuhn-Tucker conditions of optimality(Fletcher, 1980; Fletcher, 1981)

$$\frac{\partial L}{\partial \upsilon} = \frac{\partial L}{\partial b} = \frac{\partial L}{\partial \epsilon_k} = \frac{\partial L}{\partial \alpha_k} = 0 \tag{13}$$

the result can be transformed in matrix form

$$\left[\begin{array}{c|c} 0 & \vec{1}^T \\ \hline \vec{1} & \Omega + \gamma^{-1} I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \qquad (14)$$

where $y = [y_1 \dots y_n]$, $\vec{1} = [1 \dots 1]$, $\alpha = [\alpha_1 \dots \alpha_n]$ and Mercer's condition

$$\Omega_{ij} = \langle \phi(X_i), \phi(X_j) \rangle = K(X_i, X_j) \qquad (15)$$

with $i, j = 1, \dots, n$.

Then by solving the linear system (14) the approximating function (9) is

$$y(X) = \sum_{k=1}^{n} \alpha_k K(X_k, X) + b \,. \qquad (16)$$

4 Principle of hybridization

The idea of DE's hybridization with LS-SVM lies in the approximation of the cost function at the end of each generation. For that, n best individuals are chosen from the population by means of a sort procedure. The LS-SVM approximation (16) is constructed on basis of these individuals. Then, by solving a linear system of D variables the optimum of such approximation can be easily found. Next, if its cost function is better (*lower* in case of minimization) than that of the worst individual, the obtained optimum replaces the worst one in the population. It is shown on the figure 1. Now we'll discuss it in more detail.

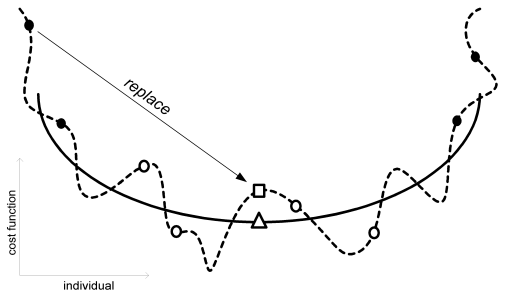

FIG. 1 – DE's hybridization with LS-SVM.

Let the cost function $f(X)$ be a nonlinear (perhaps epistatic, non differentiable and multimodal) function. Generally, there is no winning approach to calculate the global optimum by deterministic methods. Instead, heuristics give positive results, but they may converge quite slowly. Here we show one of the methods applied to iterative population-based technique, which allows to increase the convergence rate. By way of illustration we chose the Differential Evolution algorithm. The proposed hybridization does not touch the algorithm itself, but only incorporate one extra function at the end of each iteration. We consider this procedure in four steps.

Firstly, n best individuals (*circles* on the figure 1) are picked out from the population $\{ind_k, f(ind_k)\}_{k=1}^{n}$. The selection criterion is the n best values of individuals' cost function $f(ind_k)$. Thereto a sort procedure is used. For providing a good approximation it is necessary to balance n between the dimension of individuals D and the population size NP taking into consideration the properties of cost function. Our choice of inferior limit for n is $n > D + 1$. The LS-SVM support values α_k are proportional to the errors at the data points (Suykens et al., 2002). Moreover, a "big" n also increases the numerical errors (nearly singular matrix $\frac{\alpha_{min}}{\alpha_{max}} \to 0$) on solving the system (21). The superior limit $n \leq NP$ is obvious. It is always necessary to keep in mind : the less is n the less iterations are needed to find the support values α_k.

Secondly, these individuals represent support vectors (17) for the LS-SVM approximation model (9) and we find their support values α_k. Such type of SVM's is preferable due to its rapidity ($[(n+1) \times (n+1)]$ system solving) and the possibility of handling a great number of training data (a Hestenes-Stiefel conjugate gradient method).

$$\{ind_k, f(ind_k)\}_{k=1}^{n} \Leftrightarrow \{X_k, y_k\}_{k=1}^{n} \qquad (17)$$

Thirdly, in particular case it is easily to find the optimum of this approximation (*triangle* on the figure 1). Let (16) is a model for function approximation, where n is a number of support vectors (*depth of approximation*) and $K(X_k, X)$ is a second order polynomial kernel function

$$K(X_k, X) = (\langle X_k, X \rangle + 1)^2 \,, \qquad (18)$$

with $X, X_k \in \mathbb{R}^D$. Such a kernel was chosen to provide the convexity of the approximation function and the facility to find its optimum

analytically. This optimum X^* can be found from the following condition of optimality :

$$\frac{dy(X)}{dX} = \frac{\partial y(X)}{\partial X(\xi)} = 0\,, \tag{19}$$

where $X(\xi)$, $\xi = 1, \ldots, D$ are components of a vector X.

The extremum condition (19) gives the system of D linear equations :

$$\begin{aligned}
&\frac{\partial y(X)}{\partial X(\xi)} = \sum_{k=1}^{n} \alpha_k \cdot \frac{\partial K(X_k, X)}{\partial X(\xi)} = \\
&= 2\sum_{k=1}^{n} \alpha_k \left(\langle X_k, X\rangle + 1\right) X_k(\xi) \equiv 0 \quad \Leftrightarrow \\
&\sum_{k=1}^{n} \alpha_k X_k(\xi) \sum_{\varphi=1}^{D} X_k(\varphi) X(\varphi) = \\
&\sum_{\varphi=1}^{D} X(\varphi) \cdot \sum_{k=1}^{n} \alpha_k X_k(\xi) X_k(\varphi) = \\
&= -\sum_{k=1}^{n} \alpha_k X_k(\xi)
\end{aligned} \tag{20}$$

Or alternatively, in the matrix form

$$A \cdot X = B\,, \tag{21}$$

where

$$\begin{aligned}
A = a(\xi, \varphi) &= \sum_{k=1}^{n} \alpha_k X_k(\xi) X_k(\varphi) \\
B = b(\xi) &= -\sum_{k=1}^{n} \alpha_k X_k(\xi)
\end{aligned} \tag{22}$$

The optimum is found by solving this system.

Fourthly, we compare calculated from (21) optimum X^* (*square* on the figure 1) with the worst individual in the population ind^* (the highest *black point* on figure 1). The better one takes place in the population.

$$ind^* = \begin{cases} X^* & \text{if} \quad f(X^*) \le f(ind^*) \\ ind^* & \text{otherwise} \end{cases} \tag{23}$$

where

$$ind^* : \; f(ind^*) = \max_{1 \le i \le NP} f(ind_i). \tag{24}$$

Thus, at each new generation we refresh the population with an approximated optimum in the hope that it falls near to the real one, or at least it replaces the worst individual of the population.

5 Comparison of results

In order to test our approach we chose three test functions (25) from a standard test suite for Evolutionary Algorithms (Whitley et al., 1996). The first function, Rotated Ellipsoid f_1, is a true quadratic non separable optimization problem. The next two functions, Rastrigin's f_2 and Ackley's f_3, are an example of highly multimodal functions. They contain millions of local optima in the interval of consideration.

$$\begin{aligned}
f_1(X) &= \sum_{i=1}^{20} \left(\sum_{j=1}^{i} x_j \right)^2 \\
f_2(X) &= \sum_{i=1}^{20} [x_i^2 - 10cos(2\pi x_i) + 10] \\
f_3(X) &= -20 \exp\left(-0.2 \sqrt{\frac{1}{30} \sum_{i=1}^{30} x_i^2} \right) - \\
&\quad - \exp\left(\frac{1}{30} \sum_{i=1}^{30} \cos(2\pi x_i) \right) + 20 + \exp
\end{aligned} \tag{25}$$

The simulation of the algorithms have been made in the MATLAB environment with using of LS-SVMlab Toolbox (Pelckmans et al., 2003).

We fixed the control parameters of DE, the same for all functions : $NP = 100$ and $F = 0.5$. The constant of recombination $Cr = 0$ (there is no recombination) in order to make the DE algorithm rotationally invariant (Salomon, 1996; Price, 2003). The maximal number of generations g_{max} is selected for each function separately in order to provide a good visual illustration of the convergence. $n = NP$ – the depth of approximation. The average results of 10 runs are summarized in the table 1. The convergence dynamics of these test functions is shown on the figures 2-4.

As we can see such hybridization gives positive results, the algorithm converges much more quickly. But it is also clear that it consume much more time now. So the next question arise : is the algorithm really effective ? May be during

f_i	D	g_{max}	$\eta, \%$	DE	DE_h
1	20	100	22.1	1.070e+2	4.246e-4
2	20	300	11.6	1.170e+2	2.618e+1
3	30	100	18.8	7.713e+0	1.019e-1

TAB. 1 – Comparison of DE and DE_h approaches, the number of generations is fixed. D – dimension of test function. η – efficiency of approximation. DE and DE_h – the optimal values of the cost (test) function for classical and hybrid (LS-SVM) scheme accordingly.

this time the classical DE could arrive at better, more optimal value? To answer on this question we increased the maximal number of generations g^*_{max} in the classical DE so, that the algorithms work roughly the same time in both cases $t_{DE^*} \simeq t_{DE_h}$. The table 2 summarizes the obtained results.

f_i	g^*_{max}	DE^*	DE_h
1	1000	4.598e+1	4.246e-4
2	2300	8.211e+1	2.618e+1
3	1250	6.127e+0	1.019e-1

TAB. 2 – Comparison of DE^* and DE_h, the algorithm time is fixed.

Nevertheless, as may be seen from the table 2, the hybrid DE algorithm converges better. Thus, after these comparisons we can confirm that such hybridization is quite promising.

6 Remarks

However, in spite of the demonstrated potency of our approach its efficiency of approximation is far from ideal. To estimate it a efficiency measure η have been introduced. η evaluates the percentage of successful approximation, i.e. when the approximated optimum replaces the worst individual.

The average efficiency values is roughly 20%. In others words only 1/5 of iterations with LS-SVM is effective. Moreover, these are the first iterations. So, the rest of iterations do not need such hybridization. It is only loss of computational time.

As the analyse shows this problem is caused by numerical inaccuracies during solving the system (21). When the cost function approaches

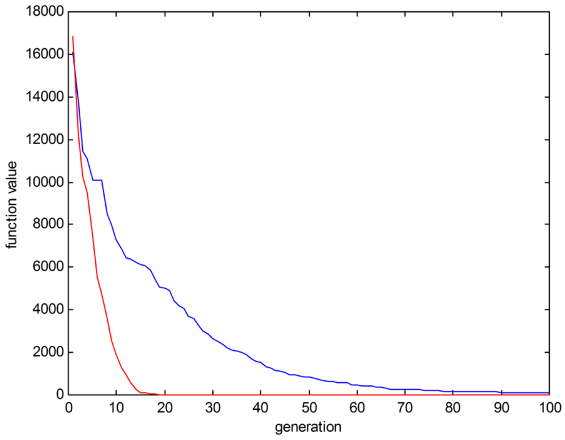

FIG. 2 – Rotated Ellipsoid function.

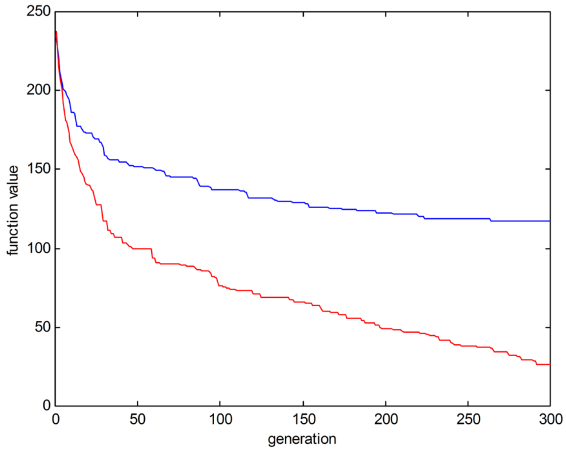

FIG. 3 – Rastrigin's function.

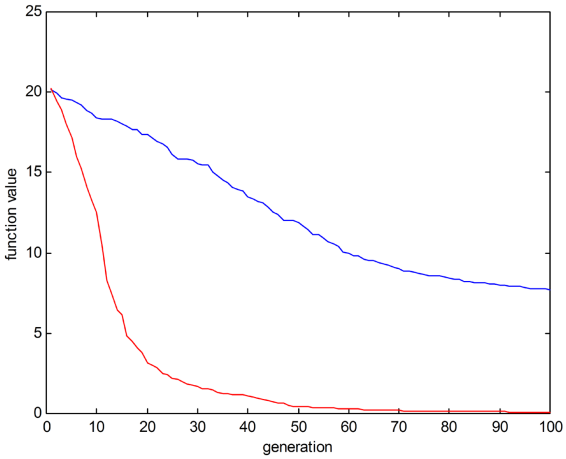

FIG. 4 – Ackley's function.

to its optimum (zero in our case) this numerical effect appears.

We are looking for other methods in order to improve the optimum approximation as well as the numerical solving of our systems.

As an example a very simple approximation could be proposed. Let calculate, for instance, a *barycenter* of n best individuals.

$$X^* = \frac{1}{n}\sum_{i=1}^{n} \hat{X}_i \qquad (26)$$

In comparison with LS-SVM method there are several advantages :

- no numerical peculiarities ;
- no inferior limits on n : $2 \leq n \leq NP$;
- very fast ;
- more efficient : $\eta \simeq 70\%$.

On the other side, its convergence is not as rapid as in the case of LS-SVM's one, but it remains faster than in the classical DE.

To confirm it numerically 10 runs have also been made. The depth of approximation $n = 10$ have been fixed. The results are summarized in the table 3. The convergence dynamics is illustrated on the figures 5-7.

f_i	D	g_{max}	$\eta, \%$	DE	DE_b
1	20	100	100.	9,003e+1	2,089e+2
2	20	300	47.0	1,118e+2	6,588e+1
3	30	100	83.1	7,030e+0	8,589E+0

TAB. 3 – Comparison of the classical DE and the barycenter approximation DE_b approaches.

It is clear that such a simple idea already increases the convergence rate.

7 Conclusion

Hybridization of an iterative population-based stochastic heuristics with approximation techniques presents a rather promising trend. The main idea consists in finding the best solution(s) on basis of selected potentially good individuals, which replace the worth one(s). Such iterative refreshment of population leads to increasing of algorithm convergence. By the example of Differential Evolution an obvious convergence amelioration have been established. Further work is necessary in order to improve numerical performances of our method.

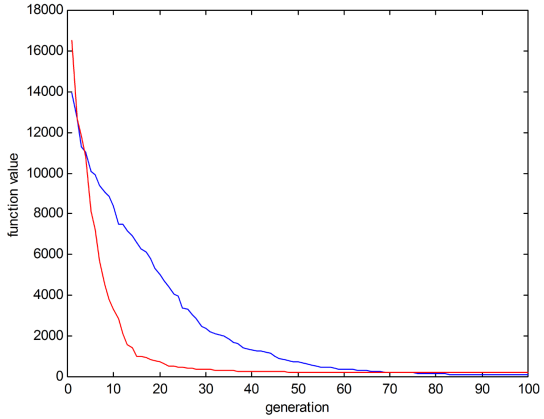

FIG. 5 – Example of barycenter approximation, Rotated Ellipsoid function.

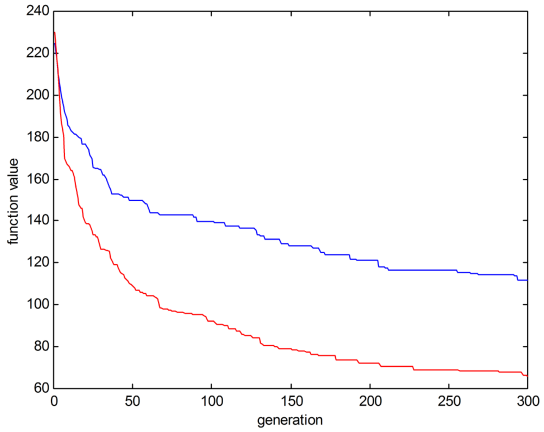

FIG. 6 – Example of barycenter approximation, Rastrigin's function.

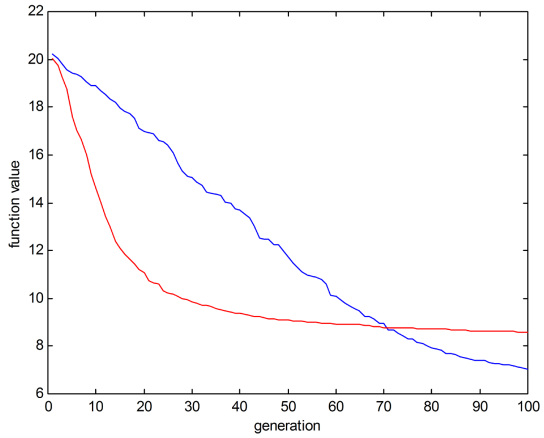

FIG. 7 – Example of barycenter approximation, Ackley's function.

References

Roger Fletcher. 1980. *Practical methods of optimisation*, volume 1 : Unconstrained Optimization. John Wiley and Sons, Chichester - New York - Brisbane - Toronto.

Roger Fletcher. 1981. *Practical methods of optimisation*, volume 2 : Constrained Optimization. John Wiley and Sons, Chichester - New York - Brisbane - Toronto.

K. Pelckmans, J.A.K. Suykens, T.Van Gestel, J.De Brabanter, L. Lukas, B. Hamers, B.De Moor, and J. Vandewalle. 2003. LS-SVMlab Toolbox User's Guide. Technical Report 02-145, Katholieke Universiteit Leuven, Belgium, February.

Kenneth Price. 2003. *New Ideas in Optimization, Part 2 : Differential Evolution*. McGraw-Hill, London, UK.

Ralf Salomon. 1996. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions : A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39 :263–278.

Rainer Storn and Kenneth Price. 1995. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, March.

Rainer Storn and Kenneth Price. 1996. Minimizing the real functions of the ICEC'96 contest by differential evolution. In *IEEE International Conference on Evolutionary Computation*, pages 842–844, Nagoya. IEEE, New York, NY, USA.

J.A.K. Suykens and J. Vandewalle. 1999. Least squares support vector machines classifiers. *Neural Processing Letters*, 9(3) :293–300.

J.A.K. Suykens, T.Van Gestel, J.De Brabanter, B.De Moor, and J. Vandewalle. 2002. *Least Squares Support Vector Machines*. World Scientific, Singapore.

Vladimir Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag, New-York.

Darrell Whitley, Soraya B. Rana, John Dzubera, and Keith E. Mathias. 1996. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2) :245–276.