THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2017, VOL. 40, NO. 2, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

OPTIMAL CONTROL USING EVOLUTIONARY ALGORITHMS

Viorel MINZU

Department of Automation and Electrical Engineering,"Dunarea de Jos" University of Galati, Romania

Abstract: Optimal Control Problems involve dynamic systems that are subject to algebraic or differential constraints and whose evolution may be characterized by a performance index. Such a problem can be solved by the well known Evolutionary Algorithms. This paper proposes an evolutionary algorithm having usual characteristics concerning the mutation and crossover operators. Generally speaking, the EA gave good results and the convergence was acceptable. But for a specific problem instance, the evolutionary algorithm underperformed on the first simulation series. Therefore, the paper proposes a new mutation operator having adaptive Gaussian standard deviation of genes' values variation.

Keywords— optimal control, Evolutionary Algorithm, mutation, adaptive Gaussian standard deviation

1. INTRODUCTION

An Optimal Control Problem (OCP) involves a dynamic system whose model is supposed to be known. Its evolution is subject to algebraic or differential constraints and initial conditions. This evolution may be characterized by a performance index that quantifies the fulfillment of the control objective.

This work proposes an evolutionary algorithm (EA) for solving OCPs. The genetic algorithm (GA) is a particular case of EAs that uses a special encoding of solutions (usually a binary encoding), which is related to the notions of genotype and phenotype. Because such an encoding is not needed in our case, EA is a natural option for solving an OCP.

There are many works that deal with EAs and GAs in a general manner, among which we recall the books (Kruse et all., 2016), (Siarry Patrick (Ed), 2016;), (Talbi E.G.,2009), (Onwubolu, 2004) and the paper (Abraham et all.,2005). Other group of works deals with the solving of OCPs using metaheuristic algorithms including EA: (Chiang, 2015), (Fogel,1994), (Qian, 2012), (Valadi, 2014), (Minzu, 2017).
The implementation of the algebraic and differential constraints within metaheuristics is a very important topic of the papers (Faber et al., 2004), (Michalewicz et al., 1992) and (Yamashita and Shimas, 1997).

The section 2 describes in a general manner the model of an OCP with its elements: the process model, initial conditions, algebraic equality, path

constraints, terminal equality constraints and bound constraints. The process model is considered as being a discrete recursive equation like in section 4, which treats a case study.

Section 3 is devoted to the description of the EA. The general structure of this algorithm being already well-known, this section gives the details and characteristics that particularize the proposed algorithm: the solutions' encoding, generation of the initial population, mutation operator, crossover operator and the other specific options for an EA.

A case study is presented in section 4, namely the well-known Linear Quadratic Problem (LQP), the discrete version. We chose this OCP as subject of our case study, because its theoretic optimal solution is known and can be compared with the solution given by the EA. In the first stage of our study, a simple MATLAB version of the proposed EA has been implemented using a non-uniform mutation operator and a single-point crossover operator. Four LQP instances defined in the paper made the object of the proposed EA. For each instance, the algorithm has been carried out 30 times due to its stochastic nature. Generally speaking, the EA has converged to good solutions, but very slowly. A huge number of objective function evaluations have been made, fact that decreases the efficiency of the proposed EA. That is why, in the second stage of our study, a new version of the proposed EA has been implemented using a mutation operator with adaptive Gaussian standard deviation. In this way, the genes' values variation is well controlled along the algorithm execution. This new version of the EA gives very good results. The convergence speed increases and the objective function evaluations' number is 15-20 times smaller.

Some conclusions are drawn in section 5 that emphasis the efficiency of the proposed EA.

2. MODEL OF OPTIMAL CONTROL PROBLEM

An OCP has some constitutive elements that are recalled hereafter. Certain mathematical details are avoided in order to simplify the presentations. The statement of an OCP has as a main element the model of a dynamic system. This one is modeled by a general differential equation or a discrete recursive equation

$$(1)\quad x_{k+1} = A\cdot x_k + B\cdot u_k ,$$

$$(2)\quad y_k = C\cdot x_k$$

where *A*, *B* and *C* are matrices with appropriate dimensions.

The variables used in this equation are as follows:

- *t:* the continuous or discrete time, $t \in \boldsymbol{R}$ or $\boldsymbol{Z}$;
- $x(t)$: the vector of state variables;
- $y(t)$: the vector of algebraic variables (usually system output variables).
- $u(t)$: the vector of control variables.

The OCP is also defined by some *equality* and *inequality constraints* listed below:

- initial conditions,
- algebraic equality,
- path constraints,
- terminal equality constraints (t_f is the final time)
- and bound constraints:

$$(3)\quad x^m \le x(t) \le x^M ;$$

$$(4)\quad u^m \le u(t) \le u^M ;$$

$$(5)\quad t_f^m \le t_f \le t_f^M ,$$

The superscripts *m* and *M* are associated with the minimum and maximum values, respectively.

The functioning of the system may be characterized through the value of a specific objective function $J\left(x(t_f), y(t_f), u(t_f), t_f\right)$. The OCP consists in finding the control variables *u* that met all the constraints and minimizes - or maximizes - the objective function

$$(6)\quad \min_{u(t),t_f} J\left(x(t_f), y(t_f), u(t_f), t_f\right).$$

3. IMPLEMENTATION OF THE EVOLUTIONARY ALGORITHM

The implementation of the proposed EA for solving an OCP is characterized first of all by the solution coding.

The optimal solutions are searched in a specific space using a time horizon that is [0, t_f]. The time discretization yields a sequence of *n* time moments, usually equidistant (Faber et al., 2004), which cover the time horizon:

$$(7)\quad \bar{t} = \left(t_1, t_2, \cdots, t_n\right)^T ;\ \text{with } t_n = t_f$$

The main unknown variables form together the control sequence $\bar{u}$ corresponding to these time moments, i.e. the so called *control profile*:

$$(8)\ \bar{u} = (u_1, u_2, \cdots, u_n);$$

Hence, the solution of an OCP $\bar{x}$ may be coded by

$$(9)\ \bar{x} = (\bar{u}, t_f)^T \text{ or } \bar{x} = \bar{u}^T,$$

according as the final time is free (unknown) or fixed.

3.1. Options for the proposed EA

We propose, for the control profile determination, an implementation of EA in which a number of options has been made:

- The population of each generation has μ individuals, which is a parameter of the algorithm; in our simulations we set μ=60;
- The population's individuals are set in increasing or decreasing order according as the objective function is minimized or maximized. Hence, the best solution is in the first position.
- The selection uses the Stochastic Universal Sampling.
- Instead of fitness values, the rank-based selection with linear-ranking is used.
- The number of offsprings generated at each generation, denoted by λ, is a parameter of the algorithm; in our simulations we set λ =30.
- Each chromosome of the selection list together with the best individual of the population is subject to the crossover operator in order to generate a single offspring.
- The crossover in a single point has been initially adopted.
- The offsprings replace the last λ individuals of the current population in order to obtain the next generation. The obtained population is reordered according the objective function values.

These options characterize a very simple version of an EA used to solve OCPs. The efficiency of this version will be analyzed in the sequel.

3.2. Generation of the initial population

Because EA is a population based metaheuristic, it has a natural exploration character. However this character must be ensured through an initial population well spread in the searching space. It is a crucial phase that has as main target the diversification.

A realistic solution is to adopt a random population uniformly distributed in the searching space. This technique is well-known for the global optimization algorithms. It is characterized by some elements:

The initial population P_0 has N individuals (solutions). Each solution $x_i \in P_0, i = 1,...,N$ is a vector with M elements x_{ij}, $j = 1,...,M$. Each element x_{ij} is placed between two limits

$$(10)\ l_j \le x_{ij} \le u_j,$$

where l_j and u_j are respectively the lower bound and upper bound. Every element is randomly generated using the equation:

$$(11)\ \begin{array}{l} x_{ij} = l_j + rand\,[0,1] \cdot (u_j - l_j), \\ i = 1,...,N;\ j = 1,...,M \end{array}$$

where $rand\,[0,1]$ is a value uniformly distributed in the interval [0, 1]. If l_j and u_j are not very well defined, then these bounds must be largely initialized such as the research space would include the optimal solution.

3.3. Mutation for real encoding

In this paper, for the implementation of the EA we adopted two strategies. Firstly we chose a very simple mutation operator having the advantage of a simple implementation and short execution time. Secondly, we adopted a more complicated but more efficient mutation operator.

Non-uniform mutation:

This operator is inspired from „simulated annealing" method and uses the following scheme:

chromosome: offspring:

$$\left(x_1^t \ldots x_i^t \ldots x_M^t\right) \rightarrow \left(x_1^{t+1} \ldots x_i^{t+1} \ldots x_M^{t+1}\right)$$

where:

$$(12)\ x_i^{t+1} = \begin{cases} x_i^t + \Delta(t, UB - x_i^t) \text{ if } a = 0 \\ x_i^t + \Delta(t, x_i^t - LB) \text{ if } a = 1 \end{cases}$$

where:

- $a \in \{0, 1\}$ is randomly chosen value;
- *UB* and *LB* are respectively the upper and lower bound of the variable x_i^t variation range:
- $LB \le x_i^t \le UB$
- $\Delta(t, y)$ is a function that returns a value between 0 and y, which tends to zero as *t* is approaching *T*. Many works use the next function:

$$(13)\ \Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right)^b} \right)$$

where:

- *r* is a real value chosen uniformly in the interval [0, 1]
- *t* is the current generation number
- *T* is the number of generations
- *b* is a parameter of the function

As we mentioned, it holds:

$$(14)\ \lim_{t \to T} \Delta(t, y) = 0 .$$

This means that there is a fine tuning of the variation introduced by the mutation, especially at the end of the searching process when the algorithm must converge to the optimal solution.

3.4. The linear BLX-α crossover

The crossover in a single point has been initially adopted. Because the algorithm's efficiency has not been always acceptable, we have used another operator as well.

For one of the problem treated below, another type of crossover operator was applied: Linear BLX-α (Linear Blend Alpha Crossover). This is described in (Siarry 2016)

If *x* and *y* are two points in $\boldsymbol{R}^n$ representing two individuals belonging to solutions' population, an individual *z* can be produced by applying Linear BLX-α crossover. This one is defined below:

$$(15)\ z = x + (y - x)U(-\alpha, 1 + \alpha)$$

where $U(-\alpha, 1 + \alpha)$ is a random value uniformly distributed in the interval [-α, 1+α]. If L is the length of the segment [*x*, *y*], then the point *z* will belong to a segment, whose length is L(1+2 α) and is centered on the segment [*x*, *y*],

4. CASE STUDY: LINEAR QUADRATIC PROBLEM

4.1. Problem statement

We present hereafter a problem involving a discrete time system that is known as being the Linear Quadratic Problem (LQP). We chose this well-known problem because it has a theoretic solution that can be compared with the solution found by the EA. The statement and the instances of the problem are taken over the work (Michalewicz 1992), where the solution is obtained with a peculiar GA, whose efficiency is not well proved in the paper. Our objective is not to propose a better algorithm producing better solutions, but to emphasize the importance of the mutation operator.

Let us consider the discrete system:

$$(16)\ x_{k+1} = a \cdot x_k + b \cdot u_k ,\ k = 0, 1, \cdots, N-1$$

where the initial state is: x_0 (given value);
The optimum criterion is:

$$(17)\ J^* = \min_{u_k,\, k=0,1,\cdots,N-1} \left[q \cdot x_N^2 + \sum_{k=0}^{N-1} \left(s \cdot x_k^2 + r \cdot u_k^2 \right) \right],$$

where the values a, b, q, s, r are given..

The theoretic solution of LQP is precised below:

$$(18)\ J^* = K_0 \cdot x_0^2$$

where K_k is the solution of the algebraic Riccati equation:

$$(19)\ K_k = r \cdot a^2 \cdot \frac{K_{k+1}}{r + b^2 \cdot K_{k+1}} + s \text{; with } K_N = q$$

LQP has been solved for 4 instances of the problem having *N*=45. These instances are described in table 1.

Table 1. Instances of LQP

problem A	*a=1; b=1; q=1; r=1; s=1; x_0=6.4*
problem B	*a=0.01; b=1; q=1; r=1; s=1; x_0=100*
problem C	*a=1; b=1; q=1; r=10; s=1; x_0=100*
problem D	*a=0.7; b=1.5; q=2; r=0.5; s=1.2; x_0=100*

4.2. Simulation results with the initial version of EA

The EA was implemented using the MATLAB language according the characteristics mentioned in section 1, even with a better crossover operator that engenders the arithmetic crossover as a particular case. We recall hereafter the main characteristics of the EA:

- Selection achieved with Stochastic Universal Sampling using rank-based selection with linear-ranking.
- Replacement of individuals with offsprings produced by the crossover operator and placed in the last λ positions of the population (with N individuals).
- The crossover operator is BLX-0.5; it produces a single offspring.
- The mutation is non-uniform on a single gene that is randomly chosen.
- The main parameters are: N=50; number of genes (*ngene*) =45; x_{min}=-5; x_{max}=5; selection pressure=1.8.

Using the equations (8) and (9), it holds:

(20) K_0=1.61803; J^*=66.2747

In the first simulation series the EA was carried out 30 times because its stochastic character. The typical result is characterized by the following values:

(21) *gen*=10000; *J**=66.981; *Neval* =301850;

where *Neval* is the number of objective function evaluations. In other words, a good control profile is obtained that leads to a value of *J* grater then the optimal value with only 1%, but after the evolution along 10000 generations and a huge number of objective function evaluations. This version of EA converges for all the executions but very slowly.

Therefore, we have adopted another mutation operator that would allow a more rigorous variation's control for the chromosome genes' values ($\Delta(t, y)$). This variation has a random character. In the case of non-uniform mutation, there is a control of the variation's variance (as random variable), but this is monotonically decreasing. That is way we need a mutation operator that would adapt this variance along the EA execution.

The book (Kruse 2016) presents in a comprehensive way the Adaptive Evolution Strategy that allows the global adaptive control of the variance as an alternative to the local adaptive control.

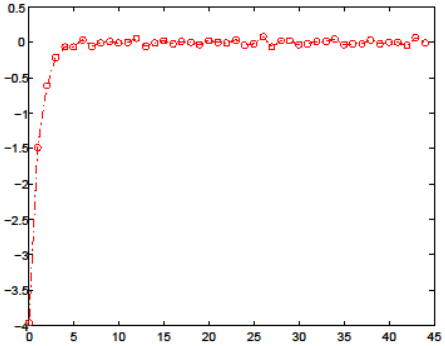
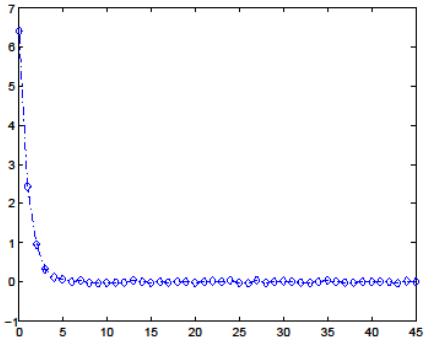
4.3. Mutation with adaptive Gaussian standard deviation

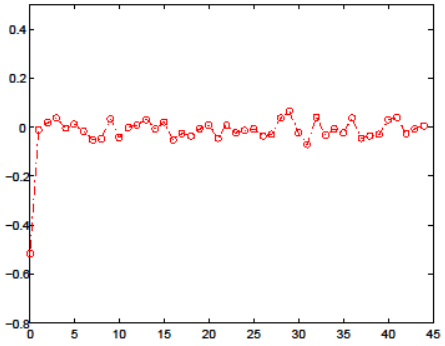
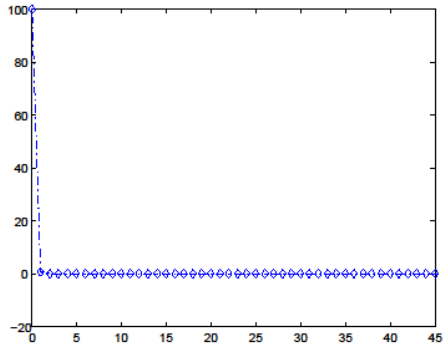
In the book (Siarry 2016), there is a different point of view in comparison with the work (Kruse 2016) concerning the implementation of the standard deviation adaptation. Inspired by the first work, we propose the mutation operator that adapts the standard deviation as it is described in Table 2. The parameter θ, $0 < \theta < 1$, is the threshold from which the standard deviation is modified at every *M* steps. A consecrated value is θ=1/5, when one can say that the "one fifth rule" is applied. Either the exploration or the exploitation is enforced respectively by the increasing or decreasing of the standard deviation, according as the success rate is greater than or less than θ. The mutation is applied to the vector x. The objective function is already calculated in the general algorithm and stored in the vector *fc*, which is meant for all the parents and offspring. This value is referred in line #9 by $fc(i_x)$. The variable *fcfm* (objective function before mutation) will store this value. The objective function value is updated in line #17, after the mutation is carried on. The call EvalFitness(x) evaluates the objective function for the chromosome x.

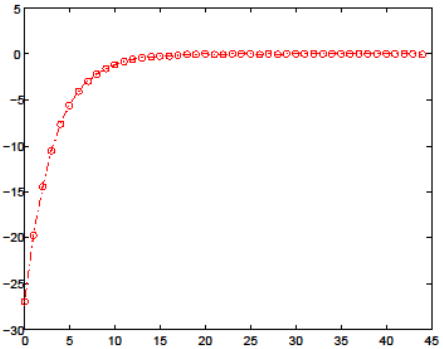
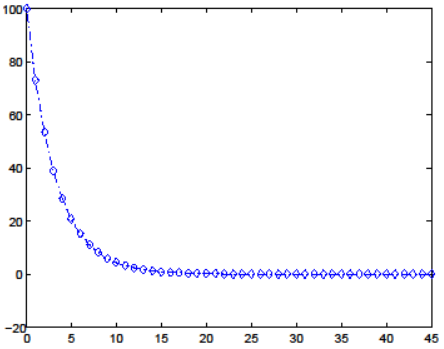
Table 2. Mutation with adaptive Gaussian standard deviation

Mutation_global_adaptive_version(x, σ)

```
1   start
2     if cm ≥ M then /* cm= mutation counter; */
3          rs=cms/cm;
           /* cms=success mutation counter;
           rs=success rate */
4          if rs < θ then σ← σ·a; ■
5          if rs > θ then σ← σ/a; ■
6          cm ←0;
7          cms ←0;
8     ■
9     fcfm←fc(i_x);
    /* i_x =index of the chromosome x */
10    for i=1,..., ngene
        /* ngene = chromosome's dimension*/
11      Δ← Gaussian random variable N(0,σ)
12      x_i← x_i + Δ;
13      x_i← max{x_i, LB_i};
        /* LB_i: lower bound of element i */
14      x_i← min{x_i, UB_i};
        /* UB_i: upper bound of element i */
15    ■
16    cm←cm+1; /*counts mutation */
17  ...fc(i_x)←EvalFitness(x);
18    if fc(i_x) < fcfm then
19         cms← cms+1;
           /* counts success mutation*/
20    ■
21  stop
```

Control profile	State evolution	Problem
		**A** a=1;b=1 q=1; r=1; s=1; x_0=6.4; J^*=66.27; J_{EA}=66.93; *Neval*=19483; *relative error*= =0.01%

		**B**
		a=0.01; b=1; q=1;
		r=1; s=1;
		x_0=100;
		J^*=10000.5;
		J_{EA}=10000.6;
		Neval=6541
		error =0.1

		**C**
		a=1; b=1; q=1;
		r=10; s=1;
		x_0=100;
		J^*=37015.6;
		J_{EA}=37015.9;
		Neval=14622;
		error =0.3

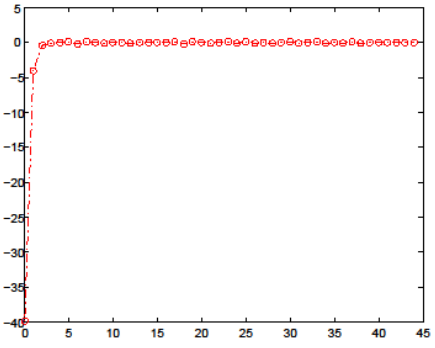
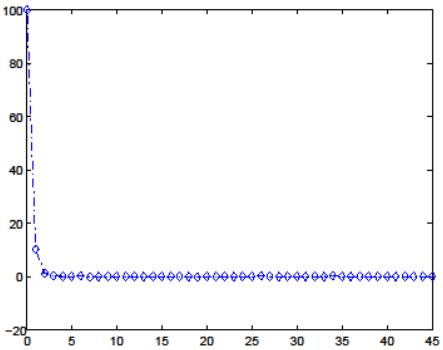
		**D**
		a=0.7; b=1.5;
		q=2; r=0.5; s=1.2;
		x_0=100;
		J^*=12929.2;
		J_{EA}=12929.8;
		Neval=13768;
		error =0.6;

Fig.1 Typical results of the EA using mutation with adaptive Gaussian standard deviation

According to the book (Siarry 2016), one can adopt $M=ngene$ - the research space being $\boldsymbol{R}^{ngene}$ -, meanwhile the work (Kruse 2016) is recommending a more consistent value, $M=\lambda \cdot k$. The second option achieves the adaptation of the standard deviation roughly after k generations. In the proposed EA, the first option was implemented because it gave very good results. The Table 2 describes this version of the mutation operator. The parameter a is place in the interval (0.85,1).

4.4. Simulation results with the modified version of EA

In a second simulation series, the modified EA was carried out 30 times for each instance of the problem. For each of them the results of its typical execution are depicted in Fig. 1. The value of J_{EA} is the best value of the objective function found by the EA.

This time, the typical executions are all convergent as well, but much more efficient than those of the original version. General speaking, *Neval* is 15-20 times smaller. The optimal value of the objective function (J^*) is practically reached for all the four instances of LQP.

The value of *error* is equal to the difference J_{EA}- J^*. The results given in fig. 1 are particularly good and prove a very good behavior of the modified EA.

5. CONCLUSION

An EA was proposed in this paper to solve OCPs. At the beginning, in the first stage of our work, a very simple version of the EA was implemented that is using a non-uniform mutation operator. The algorithm has been tested by solving four instances of LQP. This problem has been chosen as a case study because the theoretic optimal solution is already known and a comparison can be made. Despite the good quality of the found solutions, the EA has a small convergence speed.

The mutation operator changes the value of genes encoding a solution. The variation of these values has a random character. In the case of non-uniform mutation, the variation's variance (as random variable) is monotonically decreasing in accordance with the convergence process. That is way we need a mutation operator that would adapt this variance all along the EA execution. In the second stage of our work, a new version of the EA was implemented that is using an adaptive Gaussian standard deviation of this random variation.

The new EA outperformed the first version of the AE by improving the convergence speed that is 15-20 times smaller. Consequently the number of objective function evaluations is decreasing to the same extent.

As a general conclusion, this second version of the proposed EA can be used to solve other OCPs as well, but some parameters must be updated. A great deal of attention must be given to the choice of the threshold parameter θ because some authors have remarked that the value 1/5 is too optimistic in certain cases.

6. REFERENCES

Abraham, A.; L. Jain, R. Goldberg, (2005). Evolutionary Multiobjective Optimization - *Theoretical Advances and Applications*, Springer ISBN 1-85233-787-7.

Chiang, P-K., Willems, P., 2015. Combine Evolutionary Optimization with Model Predictive Control in Real-time Flood Control of a River System. *Water Resour Manage*, 29: 2527–2542

Faber, R.; T. Jockenhövelb, G. Tsatsaronis, 2004.Dynamic optimization with simulated annealing, *Computers and Chemical Engineering* 29 273–290

Fogel, D.B., (1994); Applying EvolutionaryProgramming to Selected Control Problems. *Computers Math. Applic*. Vol. **27**, No. 11, pp. 89-104, Pergamon.

Kruse, R.; C. Borgelt, C. Braune, S. Mostaghim, M. Steinbrecher; (2016) *Computational Intelligence - A Methodological Introduction*, second edition, Springer.

Michalewicz, Z; Janikow, C.; Krawczyk, J.; 1992; A Modified Genetic Algorithm for Optimal Control Problems; *Computers Math. Applic*. Vol. **23**, No. 12, pp. 83-94

Mînzu, V.; (2017). Optimal Control Using Particle Swarm Optimization, The 5th IEEEE International Symposium on Electrical and Electronics Engineering, 20-22 October, Galati, Romania

Onwubolu, G.; B.V. Babu, (2004). *New Optimization Techniques in Engineering*, Springer, ISSN 1434-9922.

Qian, F., et al., 2012. Novel hybrid evolutionary algorithm for dynamic optimization problems and its application in an ethylene oxide hydration reactor. Ind. Eng. Chem. Res. 51(49) 15974-15985.

Siarry Patrick, (Ed), 2016; *METAHEURISTICS* , Springer, ISBN 978-3-319-45401-6, 2014, ISBN 978-3-319-45403-0 (eBook).

Talbi, E.G.; *METAHEURISTICS- From Design to Implementation*, ISBN 978-0-470-27858-1, WILEY, 2009.

Valadi; J. and P. Siarry -editors, *Applications of Metaheuristics in Process Engineering*, ISBN 978-3-319-06507-6, Springer, 2014.

Yamashita, Y., Shimas, M., (1997). Numerical computational method using genetic algorithm for the optimal control problem with terminal constraints and free parameters. *Nonlinear Analysis, Theory, Methods & Application*, Vol. 30, No. 4, pp. 2285-2290.