THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2018, VOL. 41, NO.2, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

OPTIMAL CONTROL WITH BILOCAL CONSTRAINTS USING SIMULATED ANNEALING ALGORITHM

Viorel MINZU

Department of Automation and Electrical Engineering,"Dunarea de Jos" University of Galati, Romania
e-mail:Viorel.Minzu@ugal.ro

Abstract: This paper is a study on implementing Simulated Annealing (SA) to solve an Optimal Control Problem (OCP) with bilocal constraints related to state variables. The contribution is an implementation of Simulated Annealing Algorithm (SAA) which is enriched with some practical aspects. The presentation of these aspects is made through the agency of two case studies.

Keywords: optimal control problem, metaheuristic, Simulated Annealing, state variable, bilocal constraints.

1. INTRODUCTION

There is an important interest in solving an Optimal Control Problem (OCP) by metaheuristics, because of their large computation complexity. Many practical aspects concerning the implementation of such an algorithm can be found in (Talbi, 2009) and (Valadi 2014). In many practical applications, a dynamic system is modeled by a set of ordinary differential and algebraic equations, associated with a set of constraints.

This work is a study on implementing Simulated Annealing (SA) to solve OCPs with bilocal constraints related to state variables. Hence the implemented algorithm used in this work is Simulated Annealing Algorithm (SAA) that is a single solution metaheuristic. This means that the algorithm proposes a single solution at each step. The series of these solutions is a stochastic process, which under certain conditions is a Markov chain (Kirkpatrick, 1983).

The first contribution of this paper is an implementation of Simulated Annealing Algorithm which is devoted to OCPs. A classical version of SAA, which is a well-known algorithm, is described in (Faber, 2005). In this paper the SAA is enriched with some practical solutions related to the following aspects:

- the recognition of the de facto existence of the algorithm's convergence;
- the encoding of the solution used in the searching process;
- the bilocal constrains treatment when the final state is also set from the beginning;
- the treatment of the constraint involving free final time.

The presentation of these aspects is made through the agency of two case studies that generate OCP's with bilocal constrains. The both of them have not an industrial relevance, but they are useful to

understand easily how to implement the special parts within SAA. Moreover, the example presented in section 3 was the subject of a paper where that OCP was solved with a genetic algorithm. Therefore, we know a priori a very good solution of this one. The second problem described in section 4 admits a known theoretical solution expressed analytically. Hence, the quasi-optimal solutions found out by SAA can be compared with the known solutions. In this way, it is possible to assess the efficiency of SAA in solving the OCP's.

2. DETAILS CONCERNING SAA

In order to ensure the coherence of our work's presentation, the annex A gives the description of a classical implementation of SAA and the notations used in the sequel.

2.1. Integration of the convergence test

As a new element, the pseudo-code description contains a convergence testing bloc aiming to apprehend the situation when the algorithm has converged to the optimal solution of the problem. In this work, this bloc is implemented as below:

(1) $$\textit{if } \left| J\left(x_c^*\right) - J\left(x_c^{k+1}\right) \right| \le \varepsilon_1 \textit{ then } j_c \leftarrow j_c + 1;$$
$$\textit{else } j_c \leftarrow 0;$$

(2) $$\textit{if } \left(j_c \ge N_\varepsilon\right) \& \left(T_A < \varepsilon_2\right) \textit{ then } \mathsf{converg} \leftarrow 1;$$

where j_c is a variable initialized to zero, used to count the number of steps without improvement of the objective function, ε_1, ε_2 and N_ε are constant values set at initialization. If the distance between two values of the objective function is less than ε_1, SAA considers that the current solution was not improved. If $T_A < \varepsilon_2$, the cooling of T_A is sufficient to assert that SAA is in its final state. The value N_ε is the maximum accepted number of steps without improvement of the objective function. The action (2) will set the variable converg to 1 at the end of SAA, when T_A is very low and j_c exceeds N_ε. In our implementation, this means that SAA has converged and the iteration may stop. At the initialization of the algorithm, converg is set to 0.

In our proposition, SAA counts the calls of the objective function and stores the value in the variable N_{eval}. At the end of the running of SAA, its value will be a measure of the algorithm's computational complexity for solving the given problem. During the running of SAA, the value of N_{eval} is tested repeatedly. If this one exceeds a maximum number of calls, which is parameter of the algorithm, the variable converg will take the value 2 that codifies the situation when the algorithm is not convergent.

Taking into account the elements presented before, the three repetitive structures described in annex A employ as final test condition the following three conditions respectively:

condition 1: ($k \le k_{\max}$) *and* (converg =0);

condition 2: ($l \le l_{\max}$) *and* (converg = 0);

condition 3: (converg = 0);

2.2. Implementation of step length adjustment

A very important mechanism for the dynamic of SAA is the adjustment of the step length across the iterative process. In this work, it was used a tuning method that is classical method presented in (Corana, 1987) slightly modified. A key element of this method is the ratio

(3) $$a = \frac{N_{accept}}{N_{eval}},$$

where N_{accept} is the number of steps whose new candidate solution was accepted as current solution, even if it is worse than the current one. The strategy of this method is to keep the value of a as close as possible to 0.5 over the running of the algorithm. If a is too big, then too many steps that are not necessary are accepted and, as a consequence, the computation time is too big. Conversely, if a is too small then SAA fails in a local minimum.

The updating of the search step length used in this work is practically that one presented in (Corana, 1987), according to the next equation:

(4) $$dx^{l+1} = \begin{cases} dx^l \left(1 + \dfrac{a - 0.6}{0.4}\right) & \textit{if } a > 0.6 \\ dx^l & \textit{if } 0.4 \le a \le 0.6 \\ dx^l \left(1 + \dfrac{0.4 - a}{0.4}\right)^{-1} & \textit{if } a < 0.4 \end{cases}$$

Hence the updating is determined by the function (4) depicted in Fig.2.

(5) $$f(a) \overset{D}{=} \frac{dx^{l+1}}{dx^l}$$

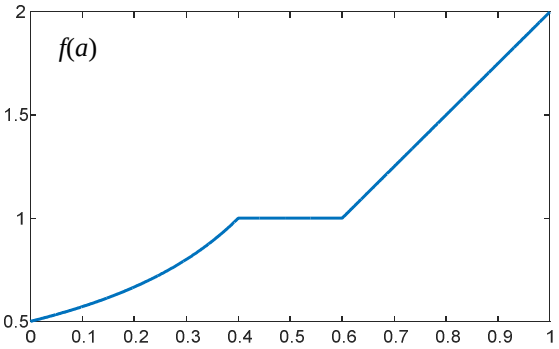

Fig.1. Updating the step length

There is also a relation between dx^l and the temperature T_A. When T_A has a high level, solutions with bigger value of the objective function (in the case of minimization) may be accepted and a greater step length may be adopted in order to cover a larger search zone. That is why the initial value of T_A is big enough to allow that practically all the moves are accepted by Metropolis rule. In this way, at least potentially the algorithm can find out the optimal solution x^*. As fast as T_A decreases, the step length has to be reduced such that degraded values of the objective function would be accepted with a ratio *a* smaller and smaller.

2.3. Implementation of cooling scheme

As it results from fig. 1, for each *m* the temperature T_A^m has a constant value for l_{max} adjustments of the step length. In this way a homogeneous Markov chain in generated. When this algorithm goes well, the optimal value x_c^* corresponding to the current temperature T_A^m is found out within the l_{max} adjustments. After that the iterative process (the repetitive structures after *k* and *l*) starts again from x_c^* with the new adjusted temperature T_A^{m+1}. At the end of the cooling scheme, the optimal solution is found out, i.e. $x_c^* = x^*$.

One of the most popular cooling schemes that could lead to the optimum solution determination is described in (Kirkpatrick, 1982). This one is described by the following geometric series:

$$(6) \qquad T_A^{m+1} = b \cdot T_A^m, 0 \le b \le 1.$$

With this cooling scheme the convergence is guaranteed and the number of objective function evaluations is acceptable. Good results are obtained for $0.5 \le b \le 0.99$. A global optimum may be reached when $b \to 1$, i.e. the cooling is very slow. If *b* has a small value, SAA may stop very fast without finding the global optimum.

3. BILOCAL OPTIMAL CONTROL RELATED TO STATE AND FREE FINAL TIME

This is a special optimal control problem (Minzu, 2017) where the dynamic process has a state representation and the initial and final states are imposed from the beginning, defining thus the bilocal character of the problem. The OCP is also completed by some *equality* and *inequality constraints*. The employment of SAA to solve this kind of problems is illustrated by two examples.

3.1. Case study

As a first case study, we take in consideration un example of OCP described in (Yamashita, 1997), where the authors have used a genetic algorithm to solve this problem. The process is modeled by nonlinear state equations with 2 state variables, x_1 and x_2, and a control input u_1:

$$(7) \qquad \begin{cases} \dot{x}_1 = -5x_1 + (1 - x_2) \cdot u_1 \\ \dot{x}_2 = (1 - x_1) \cdot u_1 \end{cases}$$

The objective function and the performance index are given hereafter:

$$(8) \qquad J = -\frac{1}{t_f}\int_0^{t_f} (3 - 4x_1)u_1 dt, \ \min_{u_1} J.$$

The value t_f is the final time considered free (unknown) in this problem. The initial time is zero. The other constraints that complete the OCP are:

-bilocal constraints:

$$x(0) = [1., 0.]^T;$$

$$x(t_f) = [0.5, 3.0]^T;$$

-bound constraints:

$$0 \le u_1 \le 10$$

-final time

t_f: free

3.2. Encoding of the optimal solution

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 (Minzu, 2017), which cover the time horizon:

$$(9) \ \bar{t} = (t_1, t_2, \cdots, t_n)^T; \text{ with } t_n = t_f$$

When the final time is free, the SAA search its value by applying random variations at every step of the SAA, like for any other unknown component of the solution. Usually the value of *n* is a constant,

set by the algorithm at initialization. Because the value of t_f=t_n changes during the execution, the values of the other time moments also changes accordingly. The main unknown variables form together the control sequence $\bar{u}$ corresponding to these time moments, i.e. the so called *control profile*:

$$(10)\quad \bar{u} = (u_1, u_2, \cdots, u_n);$$

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

$$(11)\quad \bar{x} = (\bar{u}, t_f)^T \text{ or } \bar{x} = \bar{u}^T$$

In conclusion, for bilocal problems the unknown variable t_f is adjusted by the algorithm, but paying attention to the scale factor corresponding to each component of the solution.

3.3. Dynamic system simulation over a variable time horizon

Because the final time is variable, it is necessary to simulate the dynamic system and calculate the objective function with a variable time horizon. The both actions can be simplified through a mathematical artifice: the time-scale transformation. For example, it holds:

$$(12)\qquad \tau = t \cdot \frac{1}{t_f} \Rightarrow dt = t_f \cdot d\tau$$

With this change of the time-variable, the dynamic system and the objective function become as below:

$$(13)\quad \begin{cases} \dfrac{dx_1}{d\tau} = t_f \cdot [-5x_1(\tau) + (1 - x_2(\tau))u_1(\tau)] \\ \dfrac{dx_2}{d\tau} = t_f \cdot [(1 - x_1(\tau))u_1(\tau)] \end{cases}$$

$$(14)\quad J = -\int_0^1 (3 - 4x_1)u_1 d\tau$$

The equation (12) allows the call of the function that makes the dynamic system numerical integration using invariable parameters. In the same time, for the objective function computation, one can call a numerical integration function using invariable parameters, even in the situation of having a variable t_f that doesn't disappear completely from the expression of *J*.

3.4. Bilocal constrains treatment

Let's note that this is a *bilocal optimization problem* with fixed final time. The bilocal character can be treated by adding new penalizing terms in the objective function related to the fixed state variables. In our case, the *extended objective function* may be

$$(15)\quad \min_{u(t),t_f} \left\{ \begin{array}{l} c \cdot (x_1(1) - 0.5)^2 + c \cdot (x_2(1) - 3.)^2 + \\ + \int_0^1 (-3 + 4 \cdot x_1(\tau)) \cdot u_1(\tau) d\tau \end{array} \right\},$$

where *c* is constant chosen by the implementer (for example, *c*=50).

3.5. Test and results

The SAA for solving this problem was implemented and tested under the MATLAB system. In this implementation, the variable t_f will be present in the function that computes the derivatives of the state variables for different time moments.

The values of some parameters used by the SAA for this problem are given here after:

- the number of elements of the control profile *n*=50;
- the number of solution's components, $\dim(\bar{x})$=51;
- the SA temperature T_A=2;
- *b*=0.8;
- k_{max}=20; l_{max}=10;
- ε_1=10^{-6}; ε_2=10^{-6};

The initial solution used for this problem is

$$\underbrace{[0,0,\cdots,0,2]}_{51}.$$

SAA, which is a stochastic algorithm, yields a single solution (control profile) after a single execution. Therefore, it generates a single realization of a stochastic process.

That is why we need an execution series that will be able to characterize the obtained solution. The execution series consists of a number of runs of SAA (e.g., 30–40). The average performance index over the execution series is calculated and a particular execution, whose performance index is the closest to the average, may be considered as the *typical execution*.

The majority of SAA executions carried out with the same parameters shows that the stochastic processes are convergent and meet the bilocal constraints. This fact is also ascertained when different parameters are used.

Fig. 2 and 3 show the results obtained in a typical execution with the parameters indicated before.

The smoothness of the control profile can be improved using the *step control* technique that is not presented in this paper. The decision to apply

this technique is difficult to take when the optimal control profile could have important discontinuities.

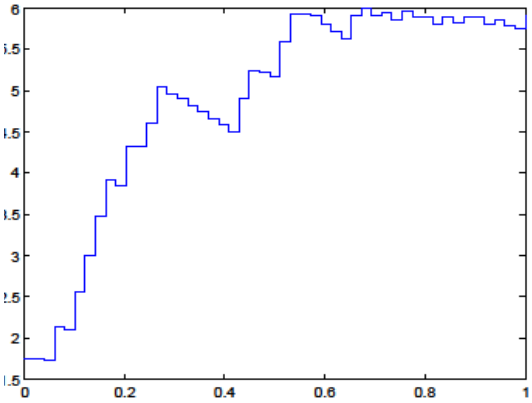

Fig.2. Control profile given by SAA

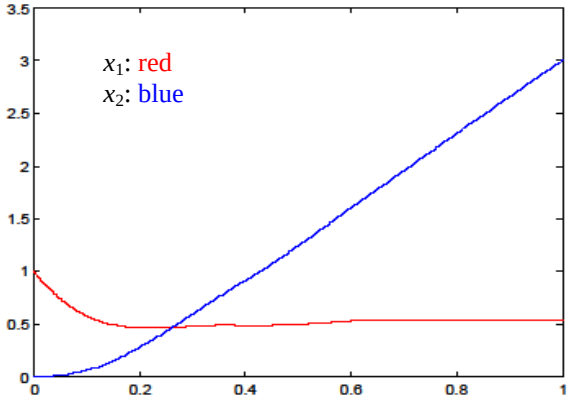

Fig.3. State evolution given by SAA

Table 1. Results of a typical execution of SAA

N_{eval}: 4154	t_f=1.2924
$T_{A\,final}$: $3.05 \cdot 10^{-6}$	J=-4.3457
dx^l_{final}: 0.00022995	$x_1(t_f)$= 0.53773
	$x_2(t_f)$= 3.0019

Table 1 shows the results obtained in a typical execution of SAA that attest the effectiveness of this algorithm in solving the given bilocal OCP. The quasi-optimal solution obtained in (Yamashita, 1997) has the final time t_f=1.3415. It is difficult to compare the two algorithms through the quality of the final solutions. The presented algorithm based on the genetic algorithm is more complex because it makes use of spline interpolation in order to cope with discontinuous control inputs. For our OCP, the optimal control profile has points of discontinuity. That is why, the solution given by the SAA seems to be inferior from the quality point of view, but this good solution is obtained after only 4154 evaluations of the objective function. Hence, the computation complexity of the proposed SAA is much smaller.

4. OBJECTIVE FUNCTION WITH FINAL TERM

Sometimes the performance index has also a final penalization term that depends on the final time t_f, which is added to the integral term..

4.1. OCP with final penalization

This OCP is described in the book (Belea, 1985). It is a problem that can be solved theoretically and consequently the optimal solution can be expressed analytically. This fact will allow us to compare the solution given by SAA with the optimal solution. The process is modeled by linear state equations with 2 state variables, x_1 and x_2, and a control input u:

$$(16) \qquad \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u(t) \end{cases}$$

The objective function and the performance index are given hereafter:

$$(17) \qquad J = \frac{1}{2}\int_0^{t_f} u^2(t)dt + t_f\,,\ \min_u J\,.$$

The value t_f is the final time considered free in this problem. It is also the terminal term of the objective function added to the integral term. The initial time is zero.

In (Minzu 2017), an apparently similar OCP is formulated, having the same state equation for the dynamic system and the same initial values. Even the performance index has the same integral term, but the terminal penalization, t_f, is missing because it is set from the beginning. Therefore, the problem is totally different. Nevertheless, some implementation aspects are the same even if the used metaheuristic is particle swarm optimization.

The other constraints that complete our OCP are:

- bilocal constraints:

$$x(0) = [1.,1.]^T;$$

$$x(t_f) = [0.,0.]^T;$$

- bound constraints:

$$0 \le u \le 5$$

-final time

t_f: free

4.2. Implementation details

We have used the same solution coding given by the equation (10). The time-scale transformation described by (11) can be used as well. The dynamic system and the objective function can be expressed as below:

$$\begin{cases} \frac{dx_1}{d\tau} = t_f \cdot x_2(\tau) \\ \frac{dx_2}{d\tau} = t_f \cdot u(\tau) \end{cases} \quad (18)$$

$$J = \frac{1}{2} \cdot t_f \cdot \int_0^1 u^2(\tau) d\tau + t_f \quad (19)$$

The values of some parameters used in SAA for this problem are given here after:

- the number of elements of the control profile n=50
- the number of solution's components, $\dim(\bar{x})$=51;
- the SA temperature T_A=0.8;
- b=0.8.
- k_{max}=20; l_{max}=10;
- $\varepsilon_1=10^{-5}$; $\varepsilon_2=10^{-4}$;

The initial solution used by SAA is

$$\underbrace{[-2,-2,\cdots,-2,4]}_{51}.$$

In a typical execution, SAA gives the evolution of the control profile and state variables depicted in fig. 4.

Table 2 shows the results obtained in a typical execution of SAA..

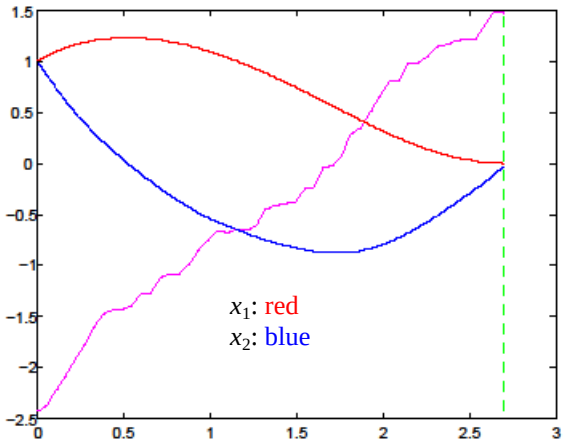

Fig.4. The control and the state variables obtained with SAA

Table 2. Results of a typical execution of SAA for the second problem

N_{eval}: 11329	t_f=2.7811
$T_{A\ final}$: $1.14\cdot 10^{-5}$	J=4.5858
dx^l_{final}: $3.81\cdot 10^{-5}$	$x_1(t_f)$= 0.0166
	$x_2(t_f)$= -0.0154

These values can be compared with the theoretic control and state values given in fig. 5.

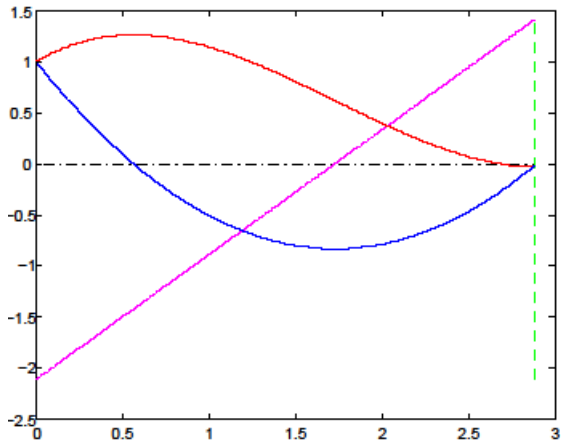

Fig.5. The theoretic control and state variables

The theoretic optimal solution is characterized by the values:

$$t_f^*=2.8848;\ J^*=4.5393$$

Obviously, these values are very close to those given by the SAA. The error of the quasi-optimal performance index is only 1%, while the error of the final time found out by the SAA is -3.5%. Also for this problem, these results show that SAA can be used in order to obtain very good solutions with a small computational complexity.

5. CONCLUSION

We have proposed an enriched SAA with elements devoted to solve OCPs with bilocal constraints. The contributions of the paper consist in underlining some practical aspects related to the implementation. The presentation was made through the agency of two case studies. The first example generates a bilocal optimization problem with fixed final time, while the second one has treated a problem with free final time. The tests were implemented within MATLAB system. and have proved that SAA gives good solutions (control profiles) and has a small computational complexity.

One can take advantage of the SAA's small computational complexity and improve the precision of the control profile through an appropriate setting of the algorithm's parameters. The objective is to make the search process more extensive and intensive in the same time, with a greater computational complexity.

On the other side, the quality of the solution can be improved from the point of view of the smoothness of the control profile. A technique called *step control* can be used in a straightforward manner as in the paper (Minzu, 2017).

As a general conclusion the SAA's implementation presented in this paper to solve OCP's with bilocal constraints turned out to be efficient.

6. ANNEX A

Description of the classical implementation of SAA

SAA is based on Monte Carlo method and it is a single solution metaheuristic. The series of the approaching solutions found out by SAA is a stochastic process that under certain conditions is a Markov chain. This process emulates the natural annealing process. The value of the objective function *J* "plays the role" of the energetic level of the metal. That is why the SAA has an intern variable called annealing temperature (T_A)

For the sake of simplicity, un optimization problem is presented in this annex having a one-dimensional solution, but the formulas are straightforward applicable in the multi-dimensional case.

A pseudo code description of SAA is presented in fig. 6. The search of the optimal solution is made through three loops:

- a loop with the counter variable k ($k \leq k_{max}$), whose inside contains the actions that search the local minimum for a constant temperature T_A^m (at step #*m*) and use the search step length dx^l (at step #*l*);
- a loop with the counter variable l ($l \leq l_{max}$) that updates the search step length dx^l;
- a loop with the counter variable m that updates the temperature T_A^m.

At the initialization of SAA, the initial solution, $x^{k=0}$, of the iterative process and the annealing temperature, $T_A^{m=0}$, are set and the value of the objective function J^k is computed. The main idea is that the optimal solution is iteratively searched, trying to improve the solution's quality at each step k using the recursive equation

$$(20)\quad x_c^{k+1} = x^k + r \cdot dx^l ,$$

where x_c^{k+1} is the candidate solution for the next step, r is random number uniformly distributed in the interval [-1, 1] and dx^l is the current step length. If the value of objective function is better (in the case of minimization), that is

$$(21)\quad J\left(x_c^{k+1}\right) \leq J\left(x^k\right),$$

then the candidate solution is accepted as next solution and possible optimal solution (after test):

$$(22)\quad x^{k+1} \leftarrow x_c^{k+1};\ x_c^* \leftarrow x_c^{k+1};$$

where x_c^* is the candidate optimal solution (the best solution found out until the current step).

If the inequality (18) is nod fulfilled, then the candidate solution may be accepted as next solution using the Metropolis rule:

1. Compute $C = \exp\left(-\frac{J\left(x_c^{k+1}\right) - J\left(x^k\right)}{k_B T_A^m}\right)$;
2. Generate a random number $\bar{p}$ uniformly distributed in the interval [0, 1];
3. *If* $C \geq \bar{p}$ *then* accept x_c^{k+1}

   else keep the same current solution x^k.

Taking into account the additional elements described in section 2, referring to the convergence of SAA, the three loops employ as final test condition the following three conditions respectively:

- *condition* 1: ($k \leq k_{max}$) *and* (converg =0);
- *condition* 2: ($l \leq l_{max}$) *and* (converg = 0);
- *condition* 3: (converg = 0);

```
Simulated Annealing Algorithm
# Set the initial solution;
# Set the algorithm's parameters
m ←1;
do
    l ←1;
    do
      k ←1;
      do
        # Generate a new current solution;
        # Test the convergence of solutions' series
        # Evaluate the objective function
        # Apply the Metropolis rule
        k ← k+1;
      while condition 1
      # Updating of the search step length
      l ← l+1;
    while condition 2
    # Update the temperature T_A
    m ← m+1;
while condition 3
# Display the quasi-optimal solution x_c*
```

Fig.6. The base version of SAA

7. REFERENCES

Belea, C; (1985), in book *Teoria sistemelor* (System Theory), Editura Didactica si Pedagogica, Bucuresti.

Corana, A. et all. (1987), Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Transactions on Mathematical Software*, *13*(3), 262–280.

Faber, R.; Jockenhövelb, Tobias; Tsatsaronis, George; (2005), Dynamic optimization with simulated annealing; *Computers and Chemical Engineering* 29 273–290.

Kirkpatrick, S.; Gelett, C. D. and Vecchi, M. P.; (1983), Optimization by simulated annealing. *Science* **220** 621-630

Mînzu, V.; (2017), Optimal control using particle swarm optimization, The 5th IEEEE International Symposium on Electrical and Electronics Engineering, 20-22 October, Galati, Romania.

Talbi, E.,G.; (2009), in book Metaheuristics from design to implementation, ISBN 978-0-470-27858-1, WILEY.

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

Yamashita, Y. and M. Shimas;(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