A Self-Tuned Simulated Annealing Algorithm using Hidden Markov Model

Mohamed Lalaoui, Abdellatif El Afia, Raddouane Chiheb
National School of Computer Science and Systems Analysis (ENSIAS), Mohammed V University, Morocco

Article Info

Article history:

Received May 22, 2017
Revised Dec 3, 2017
Accepted Dec 17, 2017

Keyword:

Heuristics
Hidden markov model
Machine learning
Prediction
Simulated annealing

ABSTRACT

Simulated Annealing algorithm (SA) is a well-known probabilistic heuristic. It mimics the annealing process in metallurgy to approximate the global minimum of an optimization problem. The SA has many parameters which need to be tuned manually when applied to a specific problem. The tuning may be difficult and time-consuming. This paper aims to overcome this difficulty by using a self-tuning approach based on a machine learning algorithm called Hidden Markov Model (HMM). The main idea is allowing the SA to adapt his own cooling law at each iteration, according to the search history. An experiment was performed on many benchmark functions to show the efficiency of this approach compared to the classical one.

Corresponding Author:

Mohamed Lalaoui,
National School of Computer Science and Systems Analysis (ENSIAS),
Mohammed V University,
Mohammed Ben Abdallah Regragui Avenue, Madinat Al Irfane, BP 713, Agdal Rabat, Morocco.
Email: med.lalaoui@yahoo.com

1. INTRODUCTION

Since the first version of simulated annealing algorithm described by [1], researchers focused on two strategies in order to improve the performance of SA. The first strategy was the implementation of parallel simulated annealing [2-4]. The second one was the optimization of cooling schedule and the adaptation of parameters. The cooling schedule is an important set of parameters that governs the convergence of SA. The set of annealing schedule as defined by [5], includes the cooling factor, the starting and stopping temperature, and the number of moves at each temperature. The cooling factor is the most influential feature among the set of annealing schedule. This factor can be defined as the method for which the algorithm reduces the temperature to its next value. If the temperature is reduced very quickly, a convergence to a local minimum may occur. However, if it is reduced too slowly, the algorithm takes a long time to converge. The most frequently used decrement rule is geometric schedule [6-8] in which the temperature decrease at each step t is governed by the formula $\theta_t = \alpha.\theta_{t-1}$, where $0.85 \leq \alpha \leq 0.96$ and α is a constant. Another method which outperforms the commonly used geometric cooling, was proposed by Lundy [9-13]. Lundy's cooling law uses the flowing formula : $\theta_t = \theta_{t-1}/(1 + \beta\, \theta_{t-1})$ where β is a suitably small value.

The use of machine learning to tune heuristic was adopted by many researchers [14- 17]. Especially, the Hidden Markov Model (HMM) [18]. HMMs success is due to ability to deal with the variability by means of stochastic modeling. It was used to enhance the behavior of metaheuristics by estimating their best configuration [19- 25].

This paper presents a new approach to enhance SA, which consists of tuning the Lundy's cooling law during the run, using the Hidden Markov Chain. The main idea is to predict the best cooling law parameter based on history of the run. To do that, first we train the HMM model by updating its parameters using Baum-Welch algorithm [18]. Then we proceed to a classification process through the Viterbi algorithm [18] which gives the most probable cooling law parameter. The rest of this paper is organized as follows. Section 2 is devoted hybridization methodology of HMM, SA, then section 3 presents and discusses the experimental results, and finally, we conclude the paper in section 4.

2. THE RESEARCH METHOD

To enhance the performance of SA, an hybridization with the HMM was adopted. During the run, the hidden Markov model performs classification based on observable sequence generated from a set of rules. This sequence allows the model to guess the hidden state which can be a slow cooling helping the algorithm to converge to a global minimum, a medium or rapid cooling to speed up the search when no improvement in solution occurs (Figure 1).

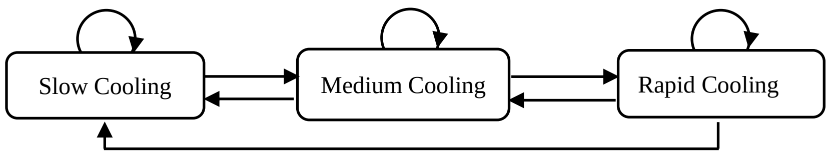

Figure 1. Markov chain for simulated annealing algorithm

The Hidden Markov Model can be defined as 5-tuple (S, O, A, B, π^0) where:

a. S= $\{S_1, S_2, S_3\}$ is set of hidden states, which is respectively: slow, medium and rapid cooling.
b. S_1: is SA with a slow Lundy decrease law, the cooling factor is $\beta = 0.005$.
c. S_2: is the same variant of simulated annealing, where the cooling law is faster than the previous one $\beta = 0.05$.
d. S_3: is Lundy simulated annealing variant with where the rapid cooling law $\beta = 0.1$.
e. $O = (O_1, O_2, \dots, O_5)$ is the set of the observation per state.
f. A $= (a_{ij})$ is a transition probability matrix, where a_{ij} is the probability that the state at time $t+1$ is S_j , is given when the state at time t is S_i
g. $\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0)$ is the initial probability, where π_i^0 is the probability of being in the state S_i.
h. $B = (b_{it})$ is the observation probabilities, where b_{it} is the probability of observing O_t in state S_i. This observations matrix B of hidden markov model is estimated at early stage by Maximum Likelihood Estimation (MLE).

The main purpose of this model is to estimate state sequence S that best explains the observation sequence O. To generate the observable sequence of HMM model. We use a progression rate described in Equation (1), and a measure of the acceptance rate of the proposed solution described in Equation (2).

$$\rho = (Number\ of\ proposal)/(Inner\ loop\ \times Outer\ loop) \quad (1)$$

Where in Equation (1), the number of proposal is the number of solution generated by the neighborhood function in each iteration, inner and outer loop are the maximum number of iterations established for SA to find the best solution.

$$w_t = (Number\ of\ accepted\ solutions)/(Number\ of\ proposal) \quad (2)$$

In Equation (2), the number of accepted solutions at iteration t is the accumulated number of accepted solution until the current iteration; and like the Eq. 1, number of proposal is the number of solution generated during the search. The acceptance rate w_t, and the progression rate ρ are then used to generate a sequence of class from a set of rules as follow:

a. O_1: little decrease of acceptance rate.
b. O_2: no improvement in cost function even if the progression rate is less than 50%.
c. O_3 : a great decrease of acceptance rate.
d. O_4 : a little increase of acceptance rate.
e. O_5 : a huge increase of acceptance rate.

During the run a set of observation is generated as follow:

Algorithm 1: Generate_Observation
```
Input:  $w_t$, ρ, Rule_1($w_t$,$\rho$), Rule_2($w_t$,$\rho$), Rule_3($w_t$,$\rho$), b bRule_4($w_t$,$\rho$), Rule_5($w_t$,$\rho$)
Output: O Current Observation
   If Rule_1($w_t$,ρ)==TRUE then O←1  End
   If Rule_2($w_t$,$\rho$)==TRUE then O←2  End
   If Rule_3($w_t$,$\rho$)==TRUE then O←3  End
   If Rule_4($w_t$,$\rho$)==TRUE then O←4  End
   If Rule_5($w_t$,$\rho$)==TRUE then O←5  End
Return O
```

The purpose of this model is to estimate state S that best explains the observation sequence O. Given the observation sequence $O = O_1 O_2 \dots O_T$ and a model $\lambda = (A, B, \pi)$. Firstly, we estimate the transition and emission probabilities from the first sequence of observation using a supervised training. In which, we count frequencies of transmissions and emissions of the model:

Algorithm 2: MLE
```
Input: $O = (O_1 O_2 \dots O_T)$
Output: A=($a_{ij}$),B =($b_{it}$)
For i = 1 to T-1 do   $a_{O_iO_{i+1}} = a_{O_iO_{i+1}} + 1$   End
For i = 1 to T   do   $b_{O_iO_i} = b_{O_iO_i} + 1$   End
For i = 1 to 3   do   $A_i = \sum_{j=1}^{3} a_{ij}$ and $B_i = \sum_{j=1}^{T} b_{ij}$   End
For $i = 1$ to 3 do
    For j=1 to 3  do $a_{ij} = a_{ij}/A_i$ End
    For t=1 to $T$ do $b_{it} = b_{it}/B_i$  End
End
Return $A$
```

Then we use the Viterbi to select the corresponding state sequence $Q = q_1 q_2 \dots q_T$ that best explains observations, secondly, the Baum Welch adjusts the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$, i.e., the probability of the observation sequence given the model.

2.1. Viterbi Algorithm

After model parameters definition, the Viterbi algorithm is used to build HMM classification process. This algorithm is used to compute the most probable path as well as its probability.

Algorithm 3: Viterbi
```
Input: S= ($s_1$,$s_2$,$s_3$), $O = (O_1 O_2 \dots O_T)$, A = ($a_{ij}$), B = ($b_{it}$) ,  $\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0)$
Output: $s^* = (s_1^*, s_2^*, s_3^*)$ the most probable sequence of states
For i = $1$ to $3$ do $\delta_1(i) = b_i(o_1)\pi_i$ and $\psi_1(i) = 0$   End   {Initialization}
For t = 2 to  T do
    For j=1 to 3  do
        $\delta_t(j) = max_{i=1}^{3}[\delta_{t-1}(i) a_{ij} b_j(o_t)]$ and $\psi_t(j) = argmax_{i=1}^{3}\ [\delta_{t-1}(i) a_{ij}]$
    End
End
$P_{max} = max_{i=1}^{3}\ [\delta_T(i)]$  ;  $s_{10}^* = argmax_{i=1}^{3}[\delta_T(i)]$
For t= $T-1$ to 1 do $s_t^* = \psi_{t+1}(s_{t+1}^*)$   End
Return $s^*$
```

2.2. Baum Welch Algorithm

The Baum–Welch algorithm is used to adjust the parameters of HMM. This training step is based on Forward-Backward algorithm.

2.2.1. Forward Algorithm

The first algorithm used by the Baum-Welch algorithm is the Forward algorithm. This algorithm returns the forward variable $\alpha_j(t)$ defined as the probability of the partial observation sequence until time t, with state S_j at time t, $\alpha_j(t) = P(O_1 O_1 \dots O_t, q_t = S_j | \lambda)$, and we define $P(O|\lambda)$ as the probability of the observation sequence given the model λ.

Algorithm 4: Forward
```
Input: S=($s_1$,$s_2$,$s_3$),O=($O_1 O_2 \dots O_T$),A =($a_{ij}$), B =($b_{it}$),$\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0)$
Output: $\alpha = (\alpha_{t+1}(j))$ , $P(O|\lambda)$
```

```
For $i = 1$ to 3 do $\alpha_i(i) = \pi_i b_{i1}$ End
For $t = 1$ to T-1 do
     For $j = 1$ to 3 do $\alpha_{t+1}(j) = \left(\sum_{i=1}^{3} \alpha_t(i) a_{ij}\right) b_{jt+1}$ End
End
 $P(O|\lambda) = \sum_{i=1}^{3} \alpha_T(i)$
Return $\alpha, P(O|\lambda)$
```

2.2.2. Backward Algorithm

The second algorithm used by Baum-Welch Backward. This algorithm calculates the backward variable $\beta_i(t)$ defined as the probability of the partial observation sequence after time t, given state S_i:
$\beta_i(t) = P(O_{t+1}O_{t+2} \dots O_T | q_t = S_i, \lambda)$

Algorithm 5: Backward
```
Input : S=($s_1$,$s_2$,$s_3$), O=($O_1$ $O_2$ ... $O_T$), A=$(a_{ij})$, B=($b_{it}$), $\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0)$
Output : $\beta = \beta_t(i)$ :the probability of the partial observation sequence
For $i = 1$ to 3 do $\beta_T(i) = 1$ End
For $t = T - 1$ to 1 do
        For $i = 1$ to 3 do $\beta_t(i) = \sum_{j=1}^{3} a_{ij} \beta_{t+1}(j) b_{jt+1}$ End
End
Return $\beta$
```

The Baum-Welch is then used to re-estimate the parameters of the model $\boldsymbol{\lambda}$, which maximizes the probability of the observation sequence. This algorithm is described as follow:

Algorithm 6: Baum-Welch
```
Input: S=($s_1$,$s_2$,$s_3$), O=($O_1$ $O_2$ ... $O_T$), A=$(a_{ij})$, B=$(b_{it})$, $\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0)$ , $\beta = \beta_t(i)$ , $\alpha = (\alpha_t(i))$, $P(O|\lambda)$
Output: $(\bar{\boldsymbol{A}}) = (\bar{a}_{ij})$ , $\bar{\boldsymbol{B}} = (\bar{b}_{it})$
Repeat
 $[\alpha, P(O|\lambda)]$ ←Forward($O$, A, B, $\pi^0$) ; β←Backward(O, A, B, $\pi^0$)
  For t=1 to T do
      For i=1 to 3 do
          For j=1 to 3 do $\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j \beta_{t+1}(j)}{P(O|\lambda)}$ End
          $\gamma_t(i) = \sum_{j=1}^{3} \xi_{ij}(t)$
      End
  End
  For k=1 to T do
     For i=1 to 3 do
        For j=1 to 3 do $\bar{\pi}_i \leftarrow \gamma_i(1)$ ; $\bar{a}_{ij} \leftarrow \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$ ; $\bar{b}_{ik} \leftarrow \frac{\sum_{t=1 \cap O_t = v_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$ End
     End
  End
While ($P(O|\lambda)$ increase)
Return $\bar{A}, \bar{B}$
```

2.3. The Hybridization of HMM and SA

In the following we will implement a variant simulated annealing based on hidden Markov models. The interest behind hybridization the simulated annealing with the HMM is to improve the SA's performance.

Algorithm 7: HMM-SA algorithm
```
Data: The objective function $f$
Initialization O:Empty observation sequence, $\theta_0$:initial temperature, $\theta_f$:final
temperature, $x \leftarrow x_0$:starting point, cmp← 1, $\rho$:progression rate, $w_0$:acceptance rate,
 n← 0:temperature stage

Repeat
    Repeat
       $x_{new} \leftarrow x + u$  {u is a Random vector from the uniform distribution over [0,1)}
       If  $f(x_{new}) - f(x) \leq 0$ then $x \leftarrow x_{new}$
                              else Generate a pseudo-random number $u^*$ {$u^* \in [0,1)$}
                                   If  $u^* < \exp(-\frac{f(x_{new}) - f(x)}{\theta_n})$ then $x \leftarrow x_{new}$ End
       End
    Until equmbrium is approached sufficiently closely at $\theta_n$
    Update($\rho$, $w_n$)
```

```
        $O_n$ ← Generate-Observation($w_n$, ρ, Rule$_1$, Rule$_2$, Rule$_3$, Rule$_4$, Rule$_5$)
        If $cmp \le 10$ then
                $O \leftarrow [O, O_n]$ ; $[A,B] \leftarrow$ MLE($O$) ; state←Viterbi($O,A,B$) ; cmp ←cmp+1 ;
            else
                $O \leftarrow [O_{n-9},\dots,O_n]$ ; [A,B] ←Baum-Welch(O,A,B) ; state←Viterbi(O,A,B)
            End
     Cooling_law ← $H_{state}$
    $\theta_{n+1}$ ← Cooling_law($\theta_n$)
  Until $\theta_{n+1} \le \theta_f$ indicating that the system is frozen
```

3. EXPERIMENT

The experiment was designed to measure the effects of hybridization of HMM and SA and to show how our approach can improve the solution quality, we have chosen five benchmark functions selected from the literature (Table 1).

Table 1. Benchmark functions

Name	Function Formula
Six-Hump Camel	$f_1(\boldsymbol{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$
Levy N° 13	$f_2(x) = sin^2(3\pi x_1) + (x_1 - 1)^2[1 + sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + sin^2(2\pi x_2)]$
Quadric	$f_3(x) = \sum_{i=1}^{D}\left(\sum_{i=1}^{D} x_i\right)^2$
Tablet	$f_4(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$
Sphere	$f_5(x) = \sum_{i=1}^{D} x_i^2$

The proposed hybridization of SA algorithm and HMM was coded in Scilab programming language and experiments were conducted on a PC with an Intel Core i7-5500U 2.40 GHz (4 CPUs) and 8 GB of RAM. The hybridization of SA and HMM have been tested using the benchmark functions presented above. Each function was tested over 30 trials. We eliminated the effects of other factors which play an important role in the performance of algorithm, by choosing the same starting points for all methods (in each run) and their location was chosen to be far from basins of attraction of global minima. Also, we have chosen the same initial acceptance probability and an identical length of the inner and outer loops. The initial temperature, θ_0, have been calculated from mean energy rises Δf during the initialization. Before the start of the SA, the mean value of cost rises is estimated by a constant number of moves equal to 100. Then, initial temperature θ_0 is calculated using the following formula $\theta_0 = \frac{-\Delta f}{\ln P_0}$ [26], where P_0 is the initial average probability of acceptance and is taken equal to 0.95. The length T of observed sequence was chosen equal to 10.

3.1. Numerical Results

The computational results and statistical analyses are summarized in Table 2. It provides the details of the results for the test functions. The overall best solution of the total 30 replications is shown in bold. HMM-SA provides the best solution for the test functions $f_1,..,f_5$. In general, HMM-SA algorithm overcomes the classical variants in all benchmark functions.

Table 2. Results comparisons between HMM-SA and the classical SA

Functions		HMM-SA	CSA
f_1	best	**-1.032E+00**	-1.031E+00
	mean	**-1.032E+00**	-1.031E+00
f_2	best	**3.768E-06**	1.371E-05
	mean	7.864E-02	**6.944E-02**
f_3	best	**2.013E-07**	6.671E-06
	mean	**6.181E-06**	4.000E-03
f_4	best	**9.903E-07**	7.112E-06
	mean	**6.643E-05**	1.432E-03
f_5	best	**4.311E-09**	8.745E-06
	mean	**4.662E-06**	1.155E-03

3.2. Comparison of Convergence Performance

To obtain further insights into the convergence behavior our approach, HMM-SA method was compared to the classical SA. Experiments were designed to measure the effects of the hybridization of SA and HMM presented in the previous section. It was noticed that the HMM-SA can converge rapidly to global minimum. The time gained in early stage can be used to converge to a better solution. This behavior is depicted in Figure 2.

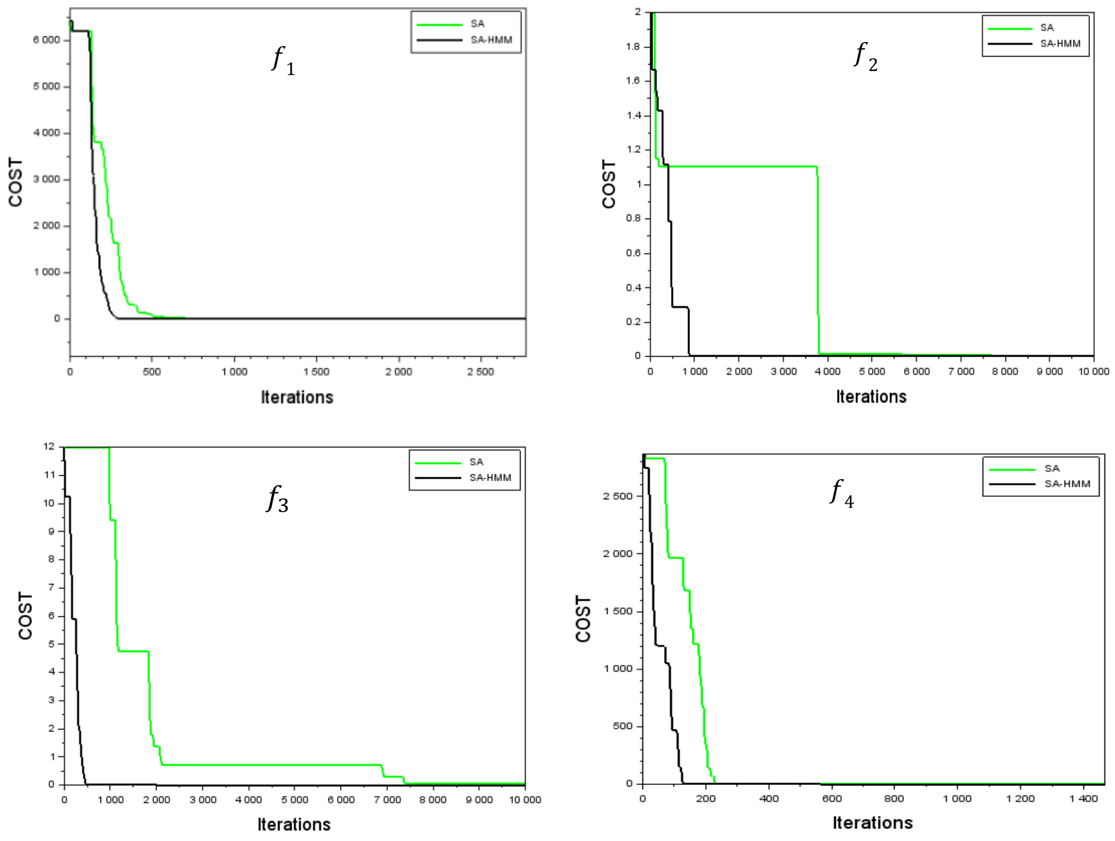

Figure 2. Comparison of HMM-SA and the classical SA

3.3. Statistical Analysis

We performed a Mann–Whitney–Wilcoxon (MWW) test [27] to determine whether the algorithm reach a significant performance. We choose this statistical test because we have two heuristics to compare. The Mann–Whitney–Wilcoxon test compares whether there is any difference from two algorithms. The null hypothesis H_0 says that the two algorithms have the same means ($H_0 : \mu_1 = \mu_2$) and the alternative hypothesis H_1 says that two algorithms have a different means ($H_1 : \mu_1 \neq \mu_2$). According to table 3, for functions f_1, f_3, f_4, f_5, the p-value is less than the significance level of $\alpha = 0.05$. We can reject the null hypothesis, so we can conclude that our hybridization of HMM and SA outperforms the classical instance of SA.

Table 3. Statistical analysis for benchmark functions

	f_1	f_2	f_3	f_4	f_5
$p - value$	1.7E-04	0.24	1.9E-09	2.0E-07	3.7E-09

4. CONCLUSION

In this study, we proposed a self-tuning capability of simulated annealing based on Hidden Markov Model. To test the performance of this approach, it was applied to a number of benchmark functions selected from literature. This approach allows to controls the cooling of SA during the run, based on sequence of state generated from a set of rules. The HMM parameters are calculated and updated at each cooling step. The Viterbi algorithm is then used to classify the observed sequence. The comparisons of the proposed approach and the classical simulated annealing demonstrate that the simulated annealing based on HMM classifier is able to find better solutions in reasonable time. Our approach is able to manage time by rapidly decreasing temperature and thus anticipating exploitation state, this lead to a better convergence. Future research may be compared to SA with fuzzy logic controllers and the application of our method to some optimization problems should be pursued.

REFERENCES

[1] S. Kirkpatrik, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, 671–680, 1983.

[2] E. Aarts, F. de Bont, E. Habers, and P. van Laarhoven, "Parallel Implementations of the Statistical Cooling Algorithm," *Integration, the VLSI Journal*, Vol. 4, 209-238, 1986.

[3] A. Casotto, F. Romeo and A. Sangiovanni-Vincentelli, *"A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells,"* Proceedings of the IEEE International Conference on Computer-Aided Design, 30-33, 1986.

[4] E. Aarts, and J. Korst, "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, " *John Wiley & Sons*, 1989.

[5] A. Kr. Yadav and al., "Design Optimization of High-Frequency Power Transformer by Genetic Algorithm and Simulated Annealing", *International Journal of Electrical and Computer Engineering*, Vol.1, No.2, pp. 102~109, 2011

[6] P. Ilamathi, V. Selladurai, K. Balamurugan, "Predictive Modelling and Optimization of Power Plant Nitrogen Oxides Emission*", IAES International Journal of Artificial Intelligence (IJ-AI)*, Vol. 1, No. 1, pp. 11~18, , 2012.

[7] Q. Chen and al., "Simulated Annealing Algorithm for Friction Parameters Identification", *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, Vol.11, No.1,pp. 245~252, 2013.

[8] M. Lundy and A. Mees, "Convergence of an Annealing Algorithm," *Mathematical Programming*, Vol 34, pp 111-124, 1986.

[9] W. A. Khan a and D. R. Hayhurst, "Computer-aided part program segmentation and reconstruction for minimization of machine tool residence time", *Int. J. Computer Integrated Manufacturing*, Vol. 4, No.5, 300-314, 1991.

[10] A. Afifi and D. R. Hayhurst, "Computer-aided part program optimization of multi-component pallet residence time in a machining centre for canned cycles and cutter tool compensation", *Int. J. Computer Integrated Manufacturing*, Vol. 8, No.1, 1-20, 1995.

[11] A. Afifi, D.R. Hayhurst and W.A. Khan, "Non-productive tool path optimisation of multi-tool part programmes, " *Int J Adv Manuf Technol*, 55:1007–1023, 2011.

[12] R. Bai, J. Blazewicz, E.K. Burke, G. Kendall and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support, " 4OR-Q J Oper Res, 10:43–66, 2012.

[13] S.Yang and J. Marcio Machado, "A Self-Learning Simulated Annealing Algorithm for Global Optimizations of Electromagnetic Devices, " *IEEE Transactions on Magnetics*, Vol. 36, N°. 4, 2000.

[14] S. Sun, F. Zhuge and S. A. Napel, "Learning enhanced simulated annealing, " *Applied Intelligence*, Vol. 28, Issue 1, pp 83-99, 2008.

[15] S.J. Jeong and K-S. Kim, "The efficient search method of simulated annealing using fuzzy logic controller," *Expert Systems with Applications* 36, 70099-7103, 2009.

[16] R. Bellio and S. Ceschia, "Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem, "*Journal of Computers & Operations Research*, 65, 83–92, 2016.

[17] L. Rabiner, *"A tutorial on hidden markov models and selected applications in speech recognition,"* Proceedings of the IEEE, 77(2), 257–286, 1989.

[18] O. Aoun, M. Sarhani and A. El Afia, "Hidden markov model classifier for the adaptive particle swarm optimization," Recent Developments in Metaheuristics, Springer, pp 1-15, 2018.

[19] O. Aoun, M Sarhani and A. El Afia, *"Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems,"* IFAC-PapersOnLine,14th IFAC Symposium on Control in Transportation Systems, Turkey,Vol. 49, Issue 3 p 347–352, 2016.

[20] S. Bouzbita, A. El Afia and R. Faizi, *"A Novel Based Hidden Markov Model Approach for Controlling the ACS Evaporation Parameter,"* submitted to the 5th International Conference on Multimedia Computing and Systems IEEE Conference, ICMCS'16, Marrakech, Morocco, 2016.

[21] M. Lalaoui, A. El Afia and R. Chiheb, *"Hidden Markov Model for a Self-Learning of Simulated Annealing Cooling Law,"* submitted to the 5th International Conference on Multimedia Computing and Systems IEEE Conference, ICMCS'16, Marrakech, Morocco, 2016.

[22] M. Lalaoui, A. El Afia and R. Chiheb, *"Hidden Markov Model for a Self-Tuning of Simulated Annealing Geometric Cooling Law Parameter,"* submitted to the 6th International Conference on Metaheuristics and Nature Inspired Computing, META'2016, Marrakech, Morocco, 2016.

[23] A. El Afia, S. Bouzbita and R. Chiheb, The Effect of Updating the Local Pheromone on ACS Performance using Fuzzy Logic", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 7, No 4, 2017.

[24] S. Bouzbita, A. El Afia, R. Faizi, M. Zbakh, *"Dynamic adaptation of the ACS-TSP local pheromone decay parameter based on the Hidden Markov Model"*, the International Conference on Cloud Computing Technologies and Applications, CloudTech, 344-349, 2016.
[25] P. Kouvelis, W.-C. Chiang, J.A. Fitzsimmons, "Simulated Annealing Procedures for Machine Layout Problems in the Presence of Zoning Constraints, " *Eur J Oper Res* 57(22), 203–223, 1992.
[26] H.B. Mann and D.R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other, " *Annals of Mathematical Statistics*,18 (1): 50–60, 1947.

BIOGRAPHIES OF AUTHORS

Mohamed Lalaoui is currently a PhD candidate at National School of Computer Science and Systems Analysis (ENSIAS), Mohammed V University, Rabat, Morocco. He obtained M.Eng. in 2011 in Computer Science from National School of Computer Science and Systems Analysis. Research areas of interest are Metaheuristics, Machine Learning and Intelligent Manufacturing.

Dr. Abdellatif El Afia is an Associate Professor at National School of Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He received his M.Sc. degrees in Applied Mathematics from University of Sherbrook. He obtained his Ph.D. in 1999 in Operation Research from University of Sherbrook, Canada. Research areas of interest are Mathematical Programming (Stochastic and deterministic), Metaheuristics, Recommendation Systems and Machine Learning.

Dr. Raddouane Chiheb is an Associate Professor at National School of Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He got Ph.D. in 1998 in Applied Mathematics from university of Jean Monnet of Saint-Étienne, France. Research areas of interest are automatic generation of Reticulated Structures, Reticulated Structures optimization, Development of Value Analysis Tools.