STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
*Stat., Optim. Inf. Comput.*, Vol. **5**, March 2017, pp 19–34.
Published online in International Academic Press (www.IAPress.org)

# A Direct Local Search Method and its Application to a Markovian Model

Zhivko Taushanov, André Berchtold *

*Institute of Social Sciences & National Centre of Competence in Research LIVES,University of Lausanne,Switzerland.*

(Received: 28 September 2016; Accepted: 23 January 2017)

**Abstract** While the hidden mixture transition distribution (HMTD) model is a powerful framework for the description, analysis, and classification of longitudinal sequences of continuous data, it is notoriously difficult to estimate because of the complexity of its solution space. In this paper, we explore how a new heuristic specifically developed for the HMTD performs compared to different standard optimization algorithms. This specific heuristic can be classified as a hill-climbing method, and different variants are proposed, including a jittering procedure to escape local maxima and measures to speed up the convergence. Different popular approaches are used for comparison, including PSO, SA, GA, NM, L-BFGS-B, and DE. The same HMTD model was optimized on different datasets and the results were compared in terms of both fit to the data and estimated parameters. Even if the complexity of the problem implies that no one algorithm can be considered as an overall best, our heuristic performed well in all situations, leading to useful solutions in terms of both fit and interpretability. The principles presented in this paper can be easily applied to other similar statistical models with complex solution spaces.

**Keywords** Hidden Mixture Transition Distribution (HMTD) model, optimization, heuristic, hill-climbing method, longitudinal data

**AMS 2010 subject classifications** 62M05,65C60

**DOI:** 10.19139/soic.v5i1.253

## 1. Introduction

Many studies deal with the problem of finding the optimum of a function without using its derivatives. Although numerous methods cope with this problem, with some more popular than others, there is no unique method that can optimally cope with all the situations. In this paper, we present a search method with hill-climbing features specifically designed to deal with the maximization of the log-likelihood of a hidden mixture transition distribution (HMTD) model for continuous variables, but which could also be used in many other problems that have similar characteristics. One advantage of this method appears when we do not have a fixed set of constraints, but we cannot accept all mathematically correct solutions. For instance, an extremely high value for the autoregressive coefficient of the mean appears occasionally as the most likely solution through the Nelder-Mead method, but such a value is non-interpretable with regard to the data meaning. This occurs especially in the case of datasets containing either short sequences or a small number of sequences. As a local search, our approach begins to explore the neighbourhood of the initial solution without going too far in the parameter space, thus avoiding aberrant solutions or numerical irregularities in the objective function.

In this paper, we explore the potential of the aforementioned method to improve the estimation of the parameters of an HMTD model for continuous variables. This Markovian model is very versatile, because it can be used to both

*Correspondence to: (Email: Andre.Berchtold@unil.ch). Institute of Social Sciences & National Centre of Competence in Research LIVES,University of Lausanne. Géopolis / SSP, Lausanne, Switzerland(CH-1015).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

describe longitudinal data and perform a classification of multiple sequences. The most time-consuming part of the modelling is the estimation of the parameters that maximize the log-likelihood. As the log-likelihood equation is not easy to derive explicitly, we need a procedure that allows us to rapidly maximize our log-likelihood function without using any derivatives. In the following sections, we briefly present the HMTD model, followed by our heuristic estimation procedure, which we compare with some other well-known heuristic methods, and finally, we present and analyse the results of different numerical experiments.

## 2. HMTD model

### *2.1. General presentation of the model*

The HMTD model is a two-level model: a visible level that models the sequence of successive observations and a latent level that drives the visible one [5]. The visible part is a mixture transition distribution (MTD) model. It was introduced by Raftery in 1985 as a modelling of high-order Markov chains [21] and developed later, among others, by Berchtold [2, 3]. Berchtold & Raftery [4] reviewed the different versions of the model. The major benefit of this model is that it allows an approximation of high-order Markov chains with fewer parameters than the full model. This is done by considering the effect of each lag on the present value as independent from the effect of other lags. Let $Y$ be a discrete random variable taking values in a finite set, and consider a $l$-th order dependence among successive observations of $Y$. Then, the full $l$-th order Markov chain is approximated as

$$P\left(Y_t = y_0 | Y_{t-1} = y_1 \dots Y_{t-\ell} = y_\ell\right) = \sum_{i=1}^{\ell} \lambda_i P_i \left(Y_t = y_0 | Y_{t-i} = y_i\right)$$

For continuous variables, the MTD becomes a mixture of Gaussian distributions and has the following form:

$$F\left(y_t | y_{t-1} \dots y_{t-\ell}\right) = \sum_{g=1}^{k} \lambda_g G_g \left(y_t | y_{t-1}, \dots, y_{t-r_g}\right)$$

where

$$G_g \left(y_t | y_{t-1}, \dots, y_{t-r_g}\right) = \Phi\left(\frac{y_t - \mu_{g,t}}{\sigma_{g,t}}\right)$$

is a Gaussian distribution with expectation $\mu_{g,t}$ and standard deviation $\sigma_{g,t}$.

The dependence between successive observations is taken into account by considering the mean $\mu_{g,t}$ of each component (and possibly its standard deviation $\sigma_{g,t}$) as a function of the past $l$ values:

$$\mu_{g,t} = \phi_{g,0} + \sum_{j=1}^{\ell} \phi_{g,j} y_{t-j}$$

$$\sigma_{g,t} = \sqrt{\theta_{g,0} + \sum_{j=1}^{\ell} \theta_{g,j} y_{t-j}^2}$$

The latent part of the model is a homogeneous Markov chain $X$ of any order, determining the component (or Gaussian distribution) used to model the current observation. We denote by $k$ the number of components, also identified as the number of possible states of the hidden Markov chain. The parameters of the latent part are the matrix of transition probabilities $A = [a_{ij}]$ between the $k$ hidden states, and $\pi$, the matrix of initial probabilities for each state. Since each state of the latent variable implies a different modelling of the observations at the visible level, the model belongs to the class of non-homogeneous Markovian modellings.

### 2.2. *Log-likelihood of the model*

The most important problem with the HMTD is to estimate the parameters that maximize the log-likelihood function. As for most hidden Markov models, this computation is based on the forward-backward algorithm introduced by Rabiner [20] which computes the marginal distribution of the latent states knowing the entire observed sequence, i.e. $P(X_t|Y_1...Y_T)$ for every $t \in (1, ..., T)$. The computation is carried out a first time forward, starting the inference from $t$=1, and then a second time backward, starting from $t = T$. Both sets of probabilities are then combined and smoothed.

Starting from known values for the parameters of the visible part of the model, we estimate the latent parameters by using the following equations (here in the case of a first-order hidden Markov chain), $\alpha_t(j)$ denoting the forward probabilities and $\beta_t(i)$ the backward ones:

$$\alpha_t(j) = P(Y_0, \ldots, Y_t, X_t = j) = \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{y_t - \mu_{j,t}}{2\sigma_j^2}\right) \sum_{i=1}^{K} a_{ij}\alpha_{t-1}(i)$$

$$\beta_t(i) = P(Y_{t+1}, \ldots, Y_T|Y_t, X_t = i) = \sum_{j=1}^{K} a_{ij}\beta_{t+1}(j) \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{y_{t+1} - \mu_{j,t+1}}{2\sigma_j^2}\right)$$

$$\gamma_t(i) = P(X_t = i|Y_0, \ldots, Y_T) = \frac{\alpha_t(i)\beta_t(i)}{L(Y_0 \ldots Y_T)}$$

$$\epsilon_t(i,j) = P(X_t = i, X_{t+1} = j|Y_0, \ldots, Y_T) = \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)\frac{1}{\sigma_j\sqrt{2\pi}} \exp(-\frac{y_{t+1}-\mu_{j,t}}{2\sigma_j^2})}{L(Y_0 \ldots Y_T)}$$

$$\hat{a}_{ij} = \sum_{t=1}^{T-1} P(X_{t+1} = j|X_t = i, Y_0, \ldots, Y_T) = \frac{\sum_{t=1}^{T-1} \epsilon_t(ij)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

where $Y_t$ is the observed data at time $t \in (1, ..., T)$, $X_t$ is the latent variable, $\gamma_t(i)$ is the marginal probability of a latent state $i$ and $\epsilon_t(i, j)$ is the joint probability of two successive latent states.

After the re-estimation of the latent parameters, we can compute the log-likelihood as

$$L(Y_0 \ldots Y_T) = \sum_{i=1}^{K} \alpha_t(i) \sum_{j=1}^{K} a_{ij}\beta_{t+1}(j) \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{y_{t+1} - \mu_{j,t}}{2\sigma_j^2}\right)$$

For instance, if all components have two lags for both the mean and the standard deviation, then

$$L(Y_0 \ldots Y_T) = \sum_{i=1}^{K} \alpha_t(i) \sum_{j=1}^{K} a_{ij}\beta_{t+1}(j) \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{y_{t+1} - (\phi_{j,0} + \phi_{j,1}y_t + \phi_{j,2}y_{t-1})}{2(\theta_{j,0} + \theta_{j,1}y_{t-1}^2 + \theta_{j,2}y_{t-2}^2)}\right)$$

This equation is not easily derivable with respect to all parameters, especially when the components use unequal numbers of lags for their mean and standard deviation. This is why we explored the use of heuristic methods as an alternative to optimize the log-likelihood function.

### 2.3. *Maximization of the log-likelihood function*

The procedure for log-likelihood maximization follows the general principle of the expectation-maximization (EM) algorithm. An alternative Markov Chain Monte Carlo approach for hidden Markov models estimation was discussed by Ryden [22] and Scott [23].

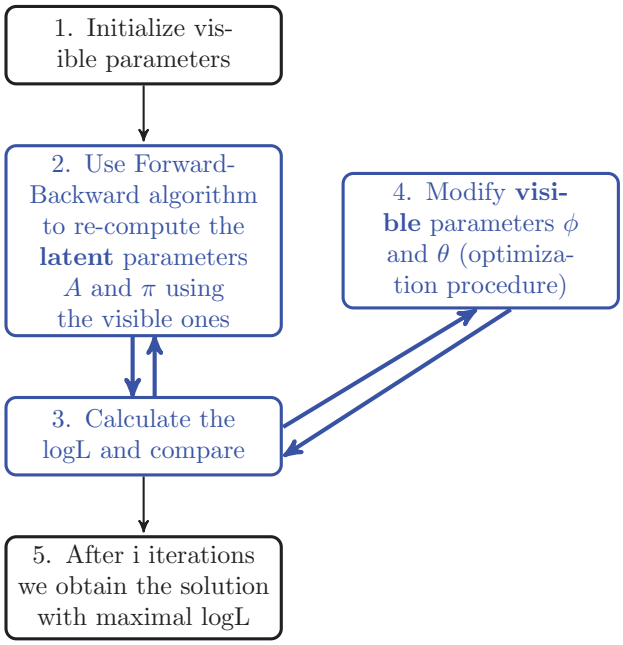


Figure 1. Steps in the estimation procedure.

In Figure 1, we briefly illustrate the main steps of the estimation procedure. After initializing the visible parameters (step 1), we apply the aforementioned forward-backward algorithm in the E-step (2) to re-estimate the parameters of the latent part of the model on the basis of the parameters of the visible part. Then, we calculate the resulting log-likelihood (3). During the maximization step (4), we try to improve the log-likelihood by re-estimating the parameters of the visible part of the model, and our heuristic procedure is used here. These re-estimated values of the visible parameters are in turn used in step 2 to compute the corresponding log-likelihood value, which indicates whether the changes in the visible parameters were beneficial. The algorithm iterates until a stopping criterion is satisfied.

The difficulties in deriving the log-likelihood equation compels us to use a heuristic (or direct) estimation procedure (in step 4) to find parameter estimates that do not decrease the log-likelihood. Because of that modification of the M-step (optimization methods instead of having equations to maximize), our procedure is qualified as a generalized EM algorithm (GEM) rather than a standard EM.

This procedure can also be applied on multiple sequences or panel data. Towards this, computations are performed separately on each independent sequence, and results are then averaged over all sequences.

## 3. Visible parameters estimation procedure

We start with an unoriented search of the maximum of the log-likelihood function. This implies the introduction of an arbitrarily chosen initial point in our parameter space. This point is chosen without any information, but by making some "semi-educated guess" by using our knowledge of the nature of the data and parameters. In other words, we try to ensure that the initial solution is not too unlikely to avoid falling in a region of the solution space corresponding to a near-zero likelihood.

### *3.1. Limits in the solution space*

As the process describing the mean (and possibly the standard deviation) of the HMTD model is autoregressive (AR), we need to define the possible solution space for our parameters. We have to consider here only the parameters of the visible part of the model, since the latent parameters are re-estimated using elements calculated by the forward-backward procedure, before returning the log-likelihood of the model. For each component of the model, we have parameters $\phi_0 \dots \phi_p$ for the mean and $\theta_0 \dots \theta_q$ for the variance. Since the model could be used with any continuous variable, we cannot exclude the possibility to have a non-stationary AR process. Therefore, we cannot define any strict bounds on our parameter space. As an example, we can imagine a process in which one hidden state implies a constant increase in the observed variable. For what concerns $\phi_{0i}$ and $\theta_{0i}$, we also have little prior knowledge. However, we know that especially in the case of a constant standard deviation, $\theta_{0i}$ cannot be negative and should be "comparable" to the standard deviation of the data. One good strategy is to fix some initial arbitrary bounds around the empirical variance of the data, and further modify them whenever the current solution provided by the estimation algorithm approaches these limits. This will be beneficial especially at the beginning of the estimation procedure because it will allow us to start from a most probable region of the solution space and prevent the algorithm from immediately spending time in exploring highly unlikely areas. In other terms, we can start in a narrow parameter space and broaden it gradually when the current solution approaches the limits.

In practice, after having set limits, we can calculate the log-likelihood of a number of randomly chosen potential solutions. The initial values for the whole estimation procedure can then be chosen either as the parameter values giving the best log-likelihood, or as a centroid of the best solutions.

Another important point to be considered is the likely presence of dependence between (some of) the parameters. Noting that

$$\phi_{i0} + \phi_{i1} y_{t-1} \cdots + \phi_{ip} y_{t-p} \approx \mu_i$$

and

$$\theta_{i0} + \theta_{i1} y_{t-1} \cdots + \theta_{iq} y_{t-q} \approx \sigma_i^2$$

we observe that increasing all the $\phi_i$ parameters simultaneously leads to a larger mean for the component $i$ (if the past observations are positive), which at certain points may diminish its probability. Therefore, all $\phi$ are interdependent. This information could also be incorporated in the estimation procedure in order to improve its efficiency. The same finding is valid for the variance of each component.

### *3.2. Searching the optimal solution*

We describe here in detail the heuristic procedure used to re-estimate the visible parameters of the model (box 4 in Figure 1). Notice that after each use of this heuristic, the estimation procedure has to also re-estimate the latent parameters (box 2). Our heuristic is related to the algorithm implemented by Berchtold [2] in the case of the discrete MTD. An important feature is that we do not make use of any derivatives of the underlying log-likelihood objective function. First, we need to introduce an initial vector of parameters. As discussed above, a good guess for the $\phi$ and $\theta$ parameters is one that gives us a value close to respectively the mean and variance of our data. From this initial vector, we evaluate the change in the log-likelihood when each of the parameters is modified. Therefore, we consecutively increase and diminish each parameter before measuring the $\Delta$logL corresponding to the change. After computing all these changes, we modify the entire vector of parameters in the direction that is optimal for each parameter separately.

We considered different versions of this procedure. The first one consisted in modifying only the parameter that enhances the most the log-likelihood and then re-estimate again the influence of the other parameters. Such a procedure make sense if our parameters are strongly dependent, but according to our experience, this dependence does not generally play a central role. The assumption of such a dependence costs too much in terms of computational time, and it appeared to be not sufficiently useful during our experiments.

If we modify all parameters simultaneously, in the direction that is optimal for them independently of all other parameters, in most of the cases we obtained a faster convergence towards the local maximum. However,

this approach ignores any dependence between the parameters, what could be problematic. For instance, if one component with larger mean could fit the data better, we would see that an increase in $\phi_{i0}$ would improve our log-likelihood, and an increase in $\phi_{i1}$ or $\phi_{i2}$ would also be beneficial. According to the aforementioned approach, we should then increase all of these parameters simultaneously. Although we make several beneficial steps simultaneously, this could lead to a decrease of the log-likelihood, because we amplify the effect of increasing the mean, making it too large.

To fix this problem, we can include the modification of the previous parameters before testing the influence of the next one on the log-likelihood (a method that we name "**S**" in our outputs). For instance, we test the effect upon the log-likelihood of a change in parameter $\phi_{i2}$ with respect to the solution obtained after saving the modifications made on the previously tested parameters (i.e., $\phi_{i0}$ and $\phi_{i1}$ if the order of optimization is not permuted). By doing this, we account for the dependence between the parameters without introducing any additional computational costs.

One potential problem that remains after this change is that we may improve the log-likelihood by modifying not the "most important" parameter (in terms of log-likelihood increase) at first place, and therefore compel the most important one to adjust to the last modification of less important parameters (increase $\phi_{i1}$ to increase the mean and adjust $\phi_{i0}$ to it, instead of proceeding in the opposite order when the latter coefficient increases the most the fitness). In other words, we make a step in the less important dimension of the solution space before making a step in the most important one. This could slow down the speed of convergence of our approach. Introducing a permutation in the order of update of our parameters can solve this problem. Such a permutation is indicated by "**P**." Changing the order of modification after each iteration, according to the absolute improvement of the log-likelihood ($\Delta$logL) leads us faster to the optimal solution.

We can rarely improve the log-likelihood by both increasing and decreasing the same parameter. This would be possible only if the current value of this parameter corresponded to a (local) minima of the solution space, a very rare situation. Introducing a one-direction improvement check ("**E**") can help us spare calculations: if an increase of a parameter increases the fitness, we update the parameter without checking what happens if we decrease it. However, if the fitness value decreases in the first case, we also need to see what happens when we decrease the value of the parameter.

Finally, we also need to sacrifice some precision by introducing a minimal change step for each parameter according to its absolute value and initial limits ("**M**"). This would prevent us from spending too many computations unnecessarily, considering infinitesimal updates of the parameters. Moreover, it seems logical to allow the step to vary during estimation. Noting that all parameters need to evolve in a different manner, we also need to allow an independent variation of their step size.

The logic of this procedure is simple: assuming that our initial guess is arbitrary, in many cases, it will be very far from the optimum. That means that once the good direction for re-estimating a parameter is found, we need to accelerate the convergence by increasing the relative change of the given parameter. In order to keep the procedure stable, we introduce a limit to the change rate. When we approach the optimal value of one parameter (i.e., when a further big leap worsens our log-likelihood), we shrink its relative change considerably (up to a limit) in order to improve its estimation accuracy. If the modification of the other parameters changes the optimal value for this parameter (interdependence), we increase its amplitude of change once again, and so on until convergence. The different amplitudes of the change may play a role in determining the order of parameter re-estimation.

We implemented two types of limits for the step of each parameter: in relative (min and max limits) and in absolute (only min limits) values. The relative value limits are measured by fractions and are the same for all parameters (e.g., between 0.5% and 30%). However, for the autoregressive parameters, a precision of more than 0.01 (in absolute value) does not seem necessary. Consequently, we fixed at 0.005 times the initial value for the mean and the variance (normally distributed around the mean and variance of the data) as the lower absolute limits for the corresponding parameters.

As said before, the heuristic procedure used to re-estimate the visible part of the model is used alternatively with the forward-backward algorithm used for the latent part of the model. In practice, there are two main possibilities for the visible parameters: either we try to reestimate each of them once before going to the latent parameters, or we allocate a fixed number of function calls to the heuristic, and we go to the latent part when this number is

reached. This second possibility means that some parameters could be reestimated several times by the heuristic before going back to the reestimation of the latent parameters. When some visible parameters are far from the optimum, this method could speed up the convergence.

### *3.3. Stopping criterion*

The estimation procedure continues until a local optimum is reached, that is, the value of the log-likelihood remains the same for two consecutive iterations, which implies that no further change of the parameters improves the fitness. However, this situation is often reached too quickly and does not necessarily imply an optimal solution. Therefore, we need to "jitter" the solution in order to continue the procedure. This is done by adding random noise to the solution that we scale to 0.1 of the value of the corresponding parameter (making sure that the variance parameters remain positive). Then, we repeat the procedure until the maximal number of iterations is reached.

For most versions of the algorithm, the optimal solution is one of those achieved immediately before one of the jitters, and therefore, we need to store the parameter values only at this moment.

### *3.4. Pseudo-code*

An illustration of the whole procedure including the four above-discussed improvements (S, E, M, and P) is given by the following pseudo-code:

---

**Pseudo-code for SEMP procedure:**

```
WHILE number of function calls < max function calls
    FOR each parameter i of the solution V in the given Order
        Increase V(i) and calculate logL
        IF new logL is higher than the saved value
            save the new V(i) and increase its future change step Change(i)
        ELSE
            decrease V(i) and calculate logL
            IF new logL is higher
                save and increase Change(i)
            ELSE
                decrease Change(i)
            END
        END
    END
    Change the Order of the parameters according to the logL increase
    IF last logL = previous logL (no change of any parameter improves the fitness)
        save the parameters and logL
        jitter the vector of parameters to escape local optimum
    END
END
```

---

Notice that before using the heuristic procedure for the first time, the maximal and minimal percentage change and minimal absolute change for the AR parameters have to be chosen. Refer to Section 4 for examples.

### *3.5. Alternative procedures*

The heuristic approach developed in this paper can be compared to other existing search methods described in the literature. In the remaining of this paper, we will consider the following alternatives: Particle Swarm Optimization (PSO) [11, 25, 8], Simulated Annealing (SA) [12, 7], Genetic Algorithm (GA) [10, 27], Differential Evolution (DE)

[28], Nelder-Mead simplex algorithm (NM) [16, 26], and the Limited memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (L-BFGSB) [6]. Although there also exist many hybrid procedures combining several of the previous algorithms that are reported to work well (for instance, PSO-SA [9], PSO-GA [17] etc.), we have no evidence of any advantages in our particular case, and hence, we do not consider them further.

## 4. Numerical experiments

We describe in this section the different numerical experiments performed to evaluate the performance of our heuristic when used with the HMTD model. All computations were made in the open source R language [19]. The Mersenne Twister pseudorandom number generator [13] was used for generating random values when required. A personal computer under Windows 10, with an Intel Xeon E5-2650 running at 2.00 GH with 8 physical cores was used for the simulations. All syntaxes are available on GitHub: https://github.com/ztau/5352

### *4.1. Comparison between several versions of the heuristic*

We performed a first numerical experiment to compare different versions of our heuristic procedure. We used the following HMTD model specification: a hidden Markov chain of order two, two hidden states with constant variance, and autoregressive mean with one lag. Therefore, the visible parameter vector has the following content: $(\theta_1, \theta_2, \phi_{1,0}, \phi_{2,0}, \phi_{1,1}, \phi_{2,1})$. All datasets are available and documented in R [19]. For each dataset, we compare 6 different specifications of the model, starting with a standard implementation without S, E, M, and P options, and going up to a specification including these 4 refinements (SEMP). The initial solution is the same for all procedures, because a same seed was used for the random generator. We used the first stopping criterion, that is, until a local optimum is reached (two consecutive iterations with same log-likelihood), without jittering. Results are provided in Table 1. We observe that after including all aforementioned improvements, our approach becomes more efficient and precise in most cases. Even if we cannot clearly identify from our experiments one variant of the algorithm that would always be better than the others, the methods with most of the above-mentioned features generally have better performances. Our choice would then be the "SEMP" method, because it gives the most consistent results.

Table 1. Comparison between different variants of the heuristic. Use of the first convergence to a local optimum as stopping criterion, without jittering. For each computation, we provide the log-likelihood and the required number of iterations. Source of the datasets: R.

| | **Different variants of the heuristic** | | | | | |
|---|---|---|---|---|---|---|
| **R dataset** | **Standard** | **S** | **SE** | **SEP** | **SEM** | **SEMP** |
| UKDriverDeaths | -141.60 | -138.95 | -127.95 | -128.05 | -128.08 | -128.70 |
| | 52 | 65 | 162 | 177 | 123 | 111 |
| sunspot.month | -127.19 | -127.19 | -120.21 | -117.85 | -120.20 | -113.12 |
| | 52 | 52 | 108 | 165 | 108 | 202 |
| faithful | -89.16 | -91.23 | -83.79 | -85.46 | -83.31 | -87.90 |
| | 104 | 52 | 189 | 144 | 231 | 95 |
| JohnsonJohnson | -41.54 | -47.33 | -39.64 | -39.39 | -39.62 | -39.68 |
| | 143 | 52 | 141 | 160 | 142 | 96 |
| sunspots | -205.70 | -217.57 | -203.39 | -207.08 | -203.53 | -204.01 |
| | 156 | 52 | 185 | 96 | 198 | 144 |
| Seatbelts | -86.66 | -84.29 | -78.52 | -79.10 | -78.55 | -78.05 |
| | 52 | 65 | 153 | 128 | 153 | 176 |

### *4.2. Comparison between the new heuristic and standard optimization procedures*

In order to compare the performances of our heuristic with other methods, we ran a second set of simulations using the same HMTD model as in Section 4.1. Again, we fitted the model on several time series available in R. The

different tested procedure are four versions of our heuristic (S, SE, SEM, SEMP) as well as the SA, GA, NM, DE, and PSO procedures.

Most of the optimization methods are not too difficult to implement. However, in order to avoid any influence of the coding upon our results, we performed our comparisons by using the most common package available in R for each method: "SA" [29], "GA" [24], "PSO" [1], "optim" (package "stats" included in the base distribution of R), and "DEoptim" [15]. The number of iterations of most algorithms was limited in order to obtain comparable results. As we used different datasets with different characteristics, it was difficult to calibrate the constants and parameters of each optimization procedure (for instance, the velocity constants for PSO, the $\alpha$, $\gamma$, $\rho$, and $\sigma$ parameters for the Nelder-Mead algorithm, etc.). Therefore, we chose to leave all these parameters to their default values as chosen by the conceptors of the R implementation of each algorithm. The only parameters for which we chose the initial values were the limits of the parameter space, the initial solution for the HMTD parameters, and the maximal number of calls of the objective log-likelihood function.

The variants of our heuristic differed from each other by the presence of the different options previously described. They all include the modification of the previous parameters (S), the second variant adds the one-directional check (E) allowing us to spare function calls. The third one includes the minimal absolute parameter change (M) defined as 1/300 of the initial $\theta$ parameters, 1/200 of the initial $\phi_{i,0}$ parameters and 0.01 for the autoregressive $\phi$ parameters. The last variant adds the permutation of the parameters during the procedure (P).

*4.2.1. Maximization of the log-likelihood* For each procedure, we measured the time to convergence, the maximal log-likelihood reached and the number of calls to the function computing the log-likelihood of the HMTD model. The latter measure gives us the best indicator of efficiency of each procedure because the effectiveness of the implemented code is not considered and the time of evaluation of the log-likelihood is much more important than the one of the creation of a new parameter vector (or solution) by each method. Globally, the execution time appears to be proportional to the number of function calls.

Table 2 provides for each dataset and each estimation algorithm the maximum log-likelihood, the number of calls of the objective function (or the corresponding number of iterations for some algorithms) required to achieve this maximum, and the optimization time in seconds. We can see that among the existing methods, PSO has a good overall performance in terms of both speed and achieved maximal log-likelihood values. Its closest concurrent is the Nelder-Mead simplex optimization algorithm, whose main drawback is the lack of possibility to introduce constraints. Unfortunately, this is a major issue in our case, because some infeasible solutions may spuriously achieve higher log-likelihood values (for instance: autoregressive coefficients above 100'000). It happened several times during our experiments, therefore we had to ban the Nelder-Mead method despite its overall good performance.

The genetic algorithm was too slow in our experimets, and it was surpassed by all other methods. A possible reason is that GA appears to perform well in very difficult problems, and not well enough in simple ones, as suggested by Pukkala & Kurttila [18]. The simulated annealing, L-BFGS-B, and differential evolution performed rather well, but not better than PSO and NM. These results suggest that PSO remains the biggest competitor of our hill-climbing heuristic.

Among the different versions of our heuristic, it is difficult to determine an overall best method. It appears that the "one-directional" check enhances the procedure, allowing us to use the function calls elsewhere instead of wasting them to check both directions. Introducing a minimal parameter change is definitely better even if we have less precision in the answers. However, the permutation of the parameters according to their importance for the log-likelihood improvement does not seem to improve the performance. Therefore, our choice tends to the third method (SEM).

*4.2.2. Acceptability of the solutions* A higher log-likelihood of a given solution does not necessarily imply its superiority over another solution. It is also very important to discuss the usefulness of a potential solution in terms of its interpretability before we accept it. Therefore, we need to examine the values of the hidden parameters ($A$ and $Pi$) that one solution implies. For instance, if we test a model with two hidden components but the hidden transition matrix $A$ of our solution suggests that one of the states is improbable, we may reject that solution because a simpler

Table 2. Comparison between the different versions of the hill-climbing heuristic, PSO, SA, GA, L-BFGS-B, Nelder-Mead, and DE: For each computation, we provide the log-likelihood, the number of function calls, and the running time in seconds (between brackets). NA means that the algorithm was unable to converge to a usable solution. The best solution found for each dataset is in bold.

| R dataset | S | SE | SEM | SEMP | PSO | SA | GA | LBFGSB | NM | DE |
|---|---|---|---|---|---|---|---|---|---|---|
| Seatbelts | -176.93 | **-171.48** | -171.70 | -172.46 | -174.87 | -174.98 | -178.69 | -179.82 | NA | -175.05 |
| | 507(35.2) | 503(37.9) | 503(38.1) | 500(38.0) | 490(36.9) | 768(58.0) | 1000(63.2) | 35(34.3) | | 42(94.7) |
| UKDriverD. | -130.22 | **-127.79** | -128.36 | -128.27 | -132.66 | -132.66 | -135.47 | -139.44 | NA | -132.74 |
| | 507(17.9) | 506(19.1) | 502(18.8) | 508(19.8) | 490(18.7) | 716(26.7) | 1000(30.9) | 10(4.9) | | 42(47.9) |
| sunspot.m. | -119.45 | -122.70 | **-112.18** | -118.51 | -117.86 | -117.86 | -122.56 | -122.40 | NA | -117.28 |
| | 507(26.4) | 509(28.6) | 510(28.7) | 511(28.8) | 490(28.2) | 547(32.1) | 1000(49.2) | 36(26.2) | | 42(69.7) |
| faithful | -69.03 | -67.82 | **-57.34** | -60.99 | -71.40 | -85.10 | -73.07 | -60.07 | -58.18 | -71.42 |
| | 507(44.8) | 500(48.4) | 501(48.1) | 511(49.2) | 490(51.3) | 287(29.2) | 1000(77.1) | 40(49.5) | 501(47.8) | 42(126.8) |
| JohnsonJ. | **-39.60** | -41.20 | -39.63 | -39.75 | -41.19 | -41.17 | -44.33 | -40.29 | -39.94 | -41.35 |
| | 507(17.4) | 504(18.7) | 508(18.9) | 501(18.8) | 490(18.0) | 300(11.0) | 1000(30.8) | 42(20.1) | 501(18.4) | 42(46.3) |
| lh | -29.53 | -33.40 | -32.71 | -32.40 | -29.08 | -29.06 | -34.50 | -21.46 | **-20.88** | -29.91 |
| | 507(51.9) | 508(56.3) | 510(56.5) | 509(56.8) | 490(59.2) | 500(56.5) | 1000(94.1) | 36(51.4) | 501(55.2) | 42(139.9) |
| ldeaths | -535.03 | -534.47 | -533.39 | -537.41 | **-523.81** | -592.85 | -591.14 | -554.66 | NA | NA |
| | 507(78.4) | 509(85.7) | 504(83.3) | 507(83.7) | 490(80.6) | 500(78.1) | 1000(137.0) | 7(14.9) | | |
| nottem[1:10] | -26.25 | -22.59 | -22.60 | -22.59 | -22.91 | -22.89 | -26.78 | -23.22 | **-22.51** | -23.49 |
| | 507(7.9) | 509(8.8) | 508(8.6) | 519(8.9) | 490(8.4) | 500(8.4) | 1000(14.0) | 41(8.9) | 501(8.4) | 42(21.2) |
| nottem[1:30] | -92.75 | -85.17 | **-85.15** | -85.25 | -88.77 | -88.77 | -97.13 | -88.89 | -95.01 | -89.35 |
| | 507(25.4) | 506(27.7) | 507(27.9) | 509(28.5) | 490(26.8) | 500(27.3) | 1000(45.8) | 42(29.8) | 501(27.0) | 42(69.6) |
| lynx | -157.05 | -157.34 | -157.72 | **-156.65** | NA | NA | NA | -166.79 | NA | NA |
| | 507(18.0) | 504(19.2) | 505(19.3) | 509(19.9) | | | | 31(15.3) | | |

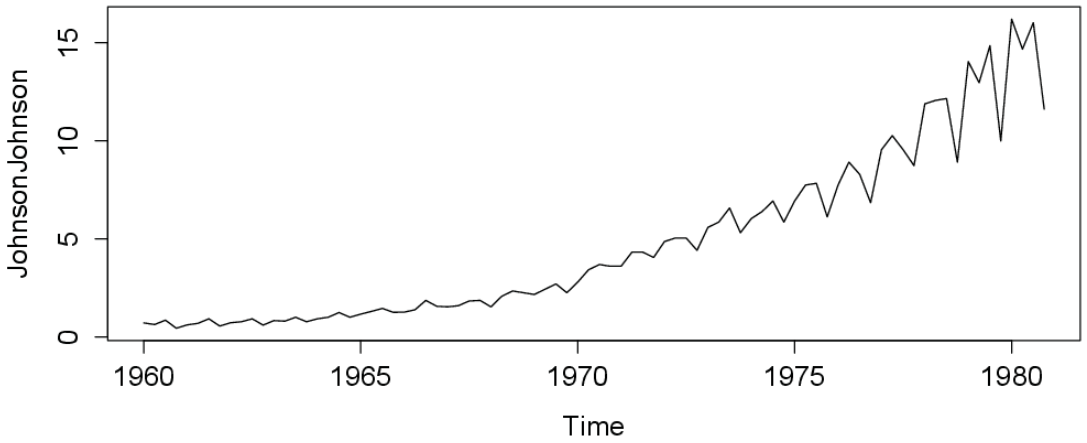


Figure 2. The "JohnsonJohnson" dataset.

model could be more appropriate, the improbable state being probably associated to only a very few number of (maybe extreme) observations.

As an example, we chose our experiment with the "JohnsonJohnson" data. This choice was made because of the availability of answers from all procedures, and because of the proximity of the best log-likelihood values achieved. These data represent the quarterly earnings in US dollars per Johnson & Johnson share during the period 1960-1980 (Figure 2). The Figure suggests a different behavior before and after 1970, with a higher variability in the second part of the series. A two-component HMTD model could then prove to be appropriate.

Let us first explore the visible parameters of the solutions of each procedure (Table 3). If we take a close look at it, we can see that some of them appear to be relatively close to each other. For instance, the solutions of our four

hill-climbing heuristic all assume one component with a low standard deviation ($\theta_{0,1}$ around 0.5 except for "S"), and a slightly negative autocorrelation for the mean ($\phi_{1,1}$ between [-0.107;-0.069]). The other component has a relatively large standard deviation and positive autocorrelation for the mean.

Table 3. Optimal solutions: visible parameters for each method on the Johnson and Johnson dataset.

| | $\theta_{0,1}$ | $\theta_{0,2}$ | $\phi_{0,1}$ | $\phi_{0,2}$ | $\phi_{1,1}$ | $\phi_{1,2}$ |
|---|---|---|---|---|---|---|
| S | 12.20 | 8.079 | 8.871 | 9.194 | -0.069 | 0.181 |
| SE | 0.506 | 4.538 | 8.151 | 9.358 | -0.107 | 0.154 |
| SEM | 0.512 | 4.082 | 7.823 | 8.626 | -0.069 | 0.192 |
| SEMP | 0.562 | 4.086 | 8.153 | 8.581 | -0.099 | 0.182 |
| PSO | 3.674 | 9.629 | 3.953 | 60.26 | 0.611 | 0.033 |
| SA | 5.148 | 3.594 | 55.02 | 4.175 | 0.225 | 0.585 |
| GA | 6.250 | 6.216 | 8.056 | 41.30 | 0.165 | 0.265 |
| L-BFGS-B | 5.113 | 1.970 | 7.782 | 4.190 | 0.437 | 0.482 |
| NM | 3.118 | 0.238 | 3.042 | 8.702 | 0.754 | 0.051 |
| DE | 3.747 | 10.27 | 4.892 | 51.24 | 0.497 | 0.408 |

If we look at the solutions (Table 3), we can also observe a common part in the solutions of PSO, SA, GA, and DE. However, in this case, only one component appears similar in their solutions: the second one for SA and the first for PSO, GA, and DE, with $\theta_0$, $\phi_0$, and $\phi_1$ parameters in the intervals, respectively [3.59;6.25], [3.95;8.05], and [0.16;0.61]. Note that the order of the two components is random and that we chose not to change it in presenting the results. This logic is fully confirmed if we look at the transition matrix $A$ and the initial probabilities $Pi$ (Table 4). Values of zero and one indicates that only one component of the model is used for the entire sequence of observations, whereas the hill-climbing heuristic, NM, and L-BFGS-B algorithms use both components equally to model the data sequence. To go one step further and to decide whether a simpler one-component model should be used to model this particular dataset, we could compute different HMTD models, and rely on the BIC information criterion.

Table 4. Optimal solutions: hidden parameters for each method on the Johnson and Johnson dataset. Since each row of both the transition matrix $A$ and the probability distribution of the first two hidden states $Pi$ is a probability distribution of two elements summing to one, we only provide the first one of each of them. $A_{11,1}$ is the transition probability from the state defined by $X_{t-2} = 1$, $X_{t-1} = 1$ to $X_t = 1$; $A_{21,1}$ is the transition probability from the state defined by $X_{t-2} = 2$, $X_{t-1} = 1$ to $X_t = 1$, $Pi_1$ is the unconditional probability of the first hidden state to be $X_1 = 1$, and so on.

| | $A_{11,1}$ | $A_{21,1}$ | $A_{12,1}$ | $A_{22,1}$ | $Pi_1$ | $Pi_{1,1}$ | $Pi_{2,1}$ |
|---|---|---|---|---|---|---|---|
| S | 0.38 | 0.60 | 0.40 | 0.37 | 0.59 | 0.58 | 0.59 |
| SE | 0.53 | 0.69 | 0.41 | 0.06 | 0.85 | 0.84 | 0.88 |
| SEM | 0.49 | 0.66 | 0.39 | 0.07 | 0.83 | 0.82 | 0.86 |
| SEMP | 0.50 | 0.65 | 0.38 | 0.07 | 0.79 | 0.78 | 0.82 |
| PSO | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| SA | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| GA | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| L-BFGS-B | 0.40 | 0.53 | 0.44 | 0.33 | 0.18 | 0.22 | 0.17 |
| NM | 0.56 | 0.84 | 0.70 | 0.71 | 1.00 | 1.00 | 1.00 |
| DE | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

Observing the log-likelihood values, we also see that even if they are all indeed very close (which is the reason why we chose this dataset), all methods from the first group achieved values similar to each other and slightly higher than those from the second group. That lead us to two suggestions. The first one is that the second group was probably trapped in a local optimum using only one component and it was not able to escape from it. The second is

that the model specification with two components is slightly more appropriate for this dataset compared to another one using only one component (in which case the latent part of the model would not have been necessary). In other words, using two components, and thus using the hidden layer of the HMTD model, improved the modelling.

### *4.3. Sequence length and speed of convergence*

We performed a last set of numerical simulations to evaluate the influence of sample size on the different heuristics, and to compare their speed of convergence. We generated data sequences according to a two-component HMTD model. Both components followed a Gaussian distribution with variance $\sigma = 0.5$ and mean $\mu_t = 1 + 0.2 * x_{t-1}$ for the first one, and $\sigma = 2$ and $\mu_t = 3 + 0.6 * x_{t-1}$ for the second one. The probability to start with the first component was set to 0.75, and the hidden transition matrix between components was

$$A = \begin{bmatrix} 0.75 & 0.25 \\ 0.40 & 0.60 \end{bmatrix}$$

We considered 5 sequence lengths (15, 25, 50, 100, 200 and 300 data points), and we generated 200 sequences of each length.

*4.3.1. Number of function calls and convergence* The various optimization methods follow very different procedures, and rely on very different convergence and stopping criteria. Therefore, they also require very different amounts of time to reach an optimum, what is crucial in comparing them, because a procedure that requires more calls of the objective function has higher chances of reaching a better solution, but at the cost of more computing time. For instance, even a slight change in the stopping criterion of one specific method could result in a slightly better performance in exchange of a higher execution time. The speed of convergence to the optimum is therefore an important criterion when choosing an optimization method.

We decided to compare the different heuristic by allocating them a fixed maximal number of log-likelihood function calls (500). Often, the different methods managed to converge with this number of function calls (perhaps because of the bounded parameter space), but some did not succeed. This raises the question of the presence of error due to the non-convergence of some methods during some iterations. A possible solution is to only analyse the solutions that have converged. However by doing this we would omit the fact that for a given dataset, some methods did simply not succeed to find an acceptable solution, whereas others did. As we would inevitably obtain different proportions of solutions, we would arrive in a situation where we compare only the successful solutions, what could benefit the method with the less tolerant stopping criterion. Therefore, we chose to include all solutions obtained with a given number of function calls. This probably introduces a non-convergence error in our analysis, but results allow us to fulfill the main objective of the simulation, that is to find the fastest procedure that offers acceptable solutions by treating equally all the methods.

*4.3.2. Results* Table 5 summarizes our results by providing for each sequence length and each optimization method the mean and standard deviation of the log-likelihood of the 200 datasets.

Results show that Nelder-Mead appears to be the best method to maximize the log-likelihood of the model, right before the SEM and PSO. However, even if very popular in numerical optimization, this method suffers from a major issue: the impossibility to fix constraints. This may often be problematic, especially if we analyze small datasets where spurious optima exist. Among examples that we experienced during the optimization are negative standard deviations, and exceedingly high autocorrelation values, leading to non-interpretable solutions, even if they may be better from a strict mathematical point of view. For these reasons NM should be used only if the datasets are large enough and there are no hard constraints imposed by the model or the nature of the data.

The other methods compared here do not suffer from such a limitation, and among them the SEM and PSO appear as the best choices, the advantage in favor of SEM increasing with the length of the data sequence. On the other hand, it appears that permuting the re-estimation order of the coefficients is not useful (SEM leads to better results than SEMP). Overall, our new heuristic behaves well against its competitors allowing to reach good and interpretable results in all situations.

Table 5. Results of the simulations with 500 log-likelihood function calls. We generated 200 series of each data length, and we provide the mean and standard deviation of the 200 log-likelihood.

| Data length | | SEM | SEMP | NM | PSO | L-BFGS-B | GA | DE |
|---|---|---|---|---|---|---|---|---|
| 15 | $\mu$ | -25.42 | -25.87 | -24.80 | -25.36 | -26.13 | -27.97 | -27.74 |
| | $\sigma$ | 6.05 | 6.75 | 6.17 | 6.03 | 6.06 | 10.85 | 6.97 |
| 25 | $\mu$ | -45.83 | -46.85 | -44.93 | -45.67 | -46.96 | -53.84 | -49.74 |
| | $\sigma$ | 8.05 | 10.13 | 8.05 | 8.09 | 7.74 | 86.21 | 9.84 |
| 50 | $\mu$ | -97.62 | -100.28 | -96.84 | -98.41 | -100.25 | -110.43 | -107.60 |
| | $\sigma$ | 10.59 | 13.99 | 10.44 | 11.25 | 10.12 | 85.15 | 20.42 |
| 100 | $\mu$ | -200.62 | -204.38 | -199.18 | -202.21 | -205.23 | -234.00 | -219.43 |
| | $\sigma$ | 15.45 | 19.21 | 14.60 | 15.64 | 14.50 | 388.30 | 29.92 |
| 200 | $\mu$ | -408.95 | -416.68 | -406.25 | -414.84 | -417.99 | -450.80 | -450.95 |
| | $\sigma$ | 21.64 | 31.83 | 20.27 | 25.23 | 22.31 | 270.48 | 78.72 |
| 300 | $\mu$ | -615.56 | -626.65 | -612.02 | -625.61 | -630.34 | -668.43 | -678.95 |
| | $\sigma$ | 26.44 | 40.70 | 25.13 | 31.97 | 30.60 | 151.42 | 138.86 |

## 5. Discussion

This paper proposed a new heuristic approach for the optimization of the HMTD model. Our motivation was that the HMTD model is very complex, with many constraints on the solution space, and hence, standard available algorithms may have difficulties in finding acceptable solutions. Of particular importance is the fact that the log-likelihood function of the HMTD is difficult to differentiate. Thus, our approach does not use any derivatives.

Since our method can be qualified as a hill-climbing method, it can be compared to other neighbourhood search or hill-climbing methods (stochastic hill climbing, random restart hill climbing, etc.). Therefore, it also shares some of the issues of these methods. Among them is the ascension of *ridges* (or descend of *alleys*): All of these methods update each dimension separately, and therefore, if the direction of the ridge (or alley) is not aligned to the axis of one dimension, the algorithm has to progress in zig-zag, spending more time. Such problems may be solved, but we need to be able to detect the problem first in order to eliminate it. A typical situation of ridge is manifested when a change in one dimension introduces a possibility to improvement in another dimension that was not possible until then. If this relation between two variables continues on the same sense for more than two consecutive iterations, we can include the *simultaneous change* feature that we developed before. This feature should spare a lot of unnecessary steps. However, when the number of dimensions increases, the detection of such situations becomes more difficult. We need to highlight that the possible inclusion of simultaneous change distinguishes such procedure from standard hill-climbing methods, because it allows the solution to evolve in more than one dimension simultaneously.

Similar to the majority of optimization algorithms, our heuristic procedure requires to start from a good set of initial values. Therefore, it is a good idea to draw a number of randomly chosen points in the parameter space, evaluate them, and select either the fittest one or the centroid (mean or median point) of the best five solutions. This solution may be beneficial for our procedure despite the additional computations. Our approach is also better adapted to smooth functions, and its drawbacks are obvious if we test it on more rough functions with many local optima. In this case, our approach brings us only to the nearest optimum. The solution then is to add some noise to the parameters after reaching an optimum in order to escape this optimum in the case it is a local one, what we called "jittering".

During a test with the Rastrigin function with 40 parameters, we also observed another interesting particularity: the re-estimation step needed to be higher when we attempted to optimize high-dimensional problems. The reason was probably the diminishing sensibility of the function to the modification of a single parameter as dimensionality increases. When the outcome remained unmodified after changing one parameter, the algorithm was trapped into the current solution. A possible remedy to this situation could be to introduce higher initial values for the re-estimation step and to rise the maximum possible value of this step as the number of dimensions increases.

Another issue is that for some solutions, the computed log-likelihood decreases even below the minimal number that the machine can consider (especially if we calculate it using highly unlikely values for the parameters). As it is easier to work with log-likelihood instead of the likelihood itself, we need to ensure that the returned value of the

likelihood is not rounded to 0 (log(0)=-Inf). To fix this problem, we impose the likelihood to be greater or equal to the minimal number for the machine that we use. This problem also shows the importance of the initial solution to our approach in order to avoid areas where the objective function is flat (i.e., the log-likelihood around remains null despite the changes in the parameters). These flat areas are also an issue for the local search methods, and they are one more reason to include bounds.

In addition to the different approaches used throughout this paper, another alternative would be the meta-optimization, implemented first by Mercer & Sampson [14], which applies one optimization method to tune the parameters of another one. This procedure has been previously applied to many situations, but in our case, it does not guarantee an improvement of performance, especially if we modify the configuration of our model (number of lags, latent states, etc.). The notion of hyper-heuristics is also worth mentioning. These are techniques that are applied to heuristic methods in order to determine the most appropriate of these for a given problem or, alternatively, to generate a new heuristic method by combining existing heuristics. The aim is to find a more general optimization procedure. However, again, we do not know whether the objective function remains similar when we change the specification of the model. Moreover, as observed, the best optimization method changes in function of the chosen specification of the model, even when applied on the same dataset.

Even if our heuristic shows good performance, it could be improved in different ways. First of all, and as already mentioned, the performance of our hill-climbing approach is influenced by the initial solution. As the speed of convergence to a local optimum is rather fast, in simple problems we may want to introduce parallel computing starting from different points instead of using the "jitter" procedure. That would allow the algorithm to explore the presence of any local optima, without consuming additional time in practice, provided that today, most computers have multi-core processors nowadays. The performances of most of the procedures tested here are also influenced by the limits of the parameter space. Since the choice of these limits is very arbitrary, an introduction of "floating" limits (limits that serve as orientation, but are broadened as soon as the current best solution approaches them) may be a solution. Another possibility, directly related to the structure of the HMTD model, would be to estimate simultaneously, instead of sequentially, the visible and latent parameters. In this case we would need to introduce constraints on the hidden transition matrix $A$ in order to ensure that it will remain a proper transition probability matrix. Such estimation procedure appears to be computationally demanding taking into account the increasing number of dimensions.

The complexity of the solution space arising from statistical models such as the HMTD (especially when the number of components, lags and/or covariates increase) and the additional specificities associated with each particular dataset imply that no one optimization algorithm can be demonstrated to be always the best, and that even with a good algorithm, the fine tuning of its parameters can have a very high impact on the final result. That being said, the contingencies of applied research imply that finding the best overall solution in terms of fit to the data, hence of log-likelihood, is not always required. In most situations, increasing the log-likelihood by one or two points is useless, since it will not imply a dramatic change in the parameters, hence in the interpretation that will be made of the model. The focus then has to be put on the speed of convergence (what disqualifies GA) and on the probability to find an acceptable solution, that is one that is not influenced by the boundaries of the solution space and that is sounds in regard of the dataset. Regarding these requirements, the new heuristic presented in this paper for the HMTD model works well by minimizing the number of situations with useless results, and it should be applicable to many similar statistical models.

## Acknowledgements

This publication benefited from the support of the Swiss National Centre of Competence in Research LIVES - Overcoming vulnerability: Life course perspectives, which is financed by the Swiss National Science Foundation (grant number: 51NF40-160590). The authors are grateful to the Swiss National Science Foundation for its financial support. The funding bodies had no role in the design and conduct of the study; collection, analysis and interpretation of data; or preparation, review, and approval of the manuscript. The authors have no conflicts of interest to disclose.

## REFERENCES


[1] Bendtsen, C. (2012) pso: Particle swarm optimization. R package version 1.0.3. Available on https://cran.r-project.org/web/packages/pso/index.html.

[2] Berchtold A. (2001) Estimation in the mixture transition distribution Model. *Journal of Time Series Analysis* 22(4): 379-397.

[3] Berchtold, A. (2003) Mixture transition distribution (MTD) modelling of heteroscedastic time series. *Computational statistics and data analysis* 41(3): 399-411.

[4] Berchtold, A. and Raftery, A. (2002) The mixture transition distribution model for high-order Markov chains and non-Gaussian time series. *Statistical Science* 17(3): 328-356.

[5] Bolano D. and Berchtold A. (2016) General framework and model building in the class of Hidden Mixture Transition Distribution models. *Computational Statistics and Data Analysis* 93: 131-145.

[6] Byrd, R. H., Lu, P., Nocedal, J. and Zhu, C. (1995) A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16(5): 1190-1208.

[7] Cerny, V. (1985) Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications* 45: 41-51.

[8] Elbeltagi, E., Hegazy, T. and Grierson, D. (2005) Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics* 19: 43-53.

[9] Fang, L., Chen, P. and Liu S. (2007) Particle swarm optimization with simulated annealing for TSP. *Proceedings of the 6th WSEAS International Conference on Artificial Intelligence*, Knowledge Engineering and Data Bases (AIKED 07).

[10] Holland, J. H. (1992) Genetic algorithms. *Scientific American* 267(1): 66-72.

[11] Kennedy, J. and Eberhart, R. (1995) Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks* IV: 1942-1948.

[12] Kirkpatrick, S., Gelatt Jr, C. D. and Vecchi, M. P. (1983) Optimization by simulated annealing. *Science* 220: 671-680.

[13] Matsumoto, M. and Nishimura, T. (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation* 8(1): 3-30.

[14] Mercer, R.E. and Sampson, J.R. (1978) Adaptive search using a reproductive metaplan. *Kybernetes* 7(3): 215-228.

[15] Mullen, K., Ardia, D., Gil, D., Windover, D. and Cline, J. (2011) DEoptim: an R package for global optimization by differential evolution. *Journal of Statistical Software* 40(6): 1-26.

[16] Nelder, J.A. and Mead, R. (1965) A simplex method for function minimization. *Computer Journal* 7: 308-313.

[17] Premalatha, K. and Natarajan, A.M. (2009) Hybrid PSO and GA for global maximization. *International Journal of Open Problems in Computer Science and Mathematics* 2(4): 597-608.

[18] Pukkala, T. and Kurttila, M. (2005) Examining the performance of six heuristic optimization techniques in different forest planning problems. *Silva Fennica* 39(1): 67-80.

[19] R Core Team (2015) R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria. URL: https://www.R-project.org/.

[20] Rabiner, L. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2): 257-286.

[21] Raftery, A. (1985) A model for high-order Markov chains. *Journal of the Royal Statistical Society, series B* 47(3): 528-539.

[22] Ryden, T. (2008) EM versus Markov chain Monte Carlo for estimation of hidden Markov models: a computational perspective. *Bayesian Analysis* 3(4): 659-688.

[23] Scott, S. (2002) Bayesian methods for hidden Markov models. *Journal of the American Statistical Association* 97: 337-351.

[24] Scrucca, L. (2013) GA: A package for genetic algorithms in R. *Journal of Statistical Software* 53(4).

[25] Shi, Y. and Eberhart, R.C. (1998) A modified particle swarm optimizer. *Proceedings of IEEE International Conference on Evolutionary Computation* 69-73.

[26] Singer, S. and Nelder, J. (2009) Nelder-Mead algorithm. *Scholarpedia* 4(7): 2928.

[27] Srinivas, M. and Patnaik, L. (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on System, Man and Cybernetics* 24(4): 656-667.

[28] Storn, R. and Price, K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11: 341-359.

[29] Xiang, Y., Gubian, S., Suomela, B. and Hoeng J. (2013) Generalized simulated annealing for global optimization: the GenSA package. *The R Journal* 5(1).