A parallel strategy for an evolutionary stochastic algorithm: application to the CP decomposition of nonnegative N-th order tensors

Souquières Laura
Université de Toulon,
Aix Marseille Université,
CNRS, LIS, UMR 7020
Toulon, France,
F-83957 La Garde, France
souquieres@univ-tln.fr

Cyril Prissette
Université de Toulon,
Aix Marseille Université,
CNRS, LIS, UMR 7020
Toulon, France,
F-83957 La Garde, France
prissette@univ-tln.fr

Sylvain Maire
Université de Toulon,
Aix Marseille Université,
CNRS, LIS, UMR 7020
Toulon, France,
F-83957 La Garde, France
maire@univ-tln.fr

Nadège Thirion-Moreau
Université de Toulon,
Aix Marseille Université,
CNRS, LIS, UMR 7020
Toulon, France,
F-83957 La Garde, France
thirion@univ-tln.fr

***Abstract*—In this article, we address the problem of the Canonical Polyadic decomposition (a.k.a. CP, Candecomp or Parafac decomposition) of N-th order tensors that can be very large. In our case, this decomposition is performed under nonnegativity constraints. While this problem is often tackled thanks to deterministic approaches, we focus here, on a stochastic approach based on a memetic algorithm. It relies on the evolution of a population and a local search stage. The main drawback of such algorithms can be their relative slowness. It is the reason why we suggest and implement a parallel strategy to increase the acceptance rate of the original algorithm and thus to accelerate its convergence speed. Numerical simulations are performed in order to illustrate the effectiveness of our approach on simulated 3D fluorescence spectroscopy tensors.**

***Index Terms*—Tensor decomposition, stochastic optimization, memetic algorithms, parallel strategy**

I. Introduction

A particular attention has been dedicated to tensor decompositions with the raise of data arranged in multi-way arrays. There exist many application fields where tensors are considered (telecommunication, statistics, data mining, chemometrics, linguistics, (audio) signal processing and so on); an overview can be found in [1].

By analogy with the singular value decomposition for matrices, the Canonical Polyadic Decomposition (CPD) has been introduced for tensors by Hitchcock in [2] and rediscovered several times afterwards. To solve this decomposition problem, different algorithms have been suggested: i) direct methods (e.g. GRAM-DTLD), ii) deterministic methods (alternating or descent algorithms), among which are (Hierarchical) Alternating Least Squares algorithms (ALS and HALS) [3], proximal gradient approaches [4] and more recently researchers have also been interested in iii) stochastic approaches [5], [6] on which we focus in this article. Most methods of the first two classes of approaches are well described in [3].

This article is dedicated to the last category of methods, and more specifically to memetic algorithms [7], which can be regarded as hybrid methods between genetic evolution algorithms and local search approaches. More precisely, we focus, here, on an algorithm called Collaborative Evolution of Population (CEP) [6] [8]. The main advantages of this kind of methods are their robustness and their ability to address different problems without requiring substantive changes or complicated calculations. However, they are known for being relatively slow regarding the convergence speed. To improve the CEP algorithm, a parallel strategy is suggested here. Our aim is to overcome some issues that might occur during the local search stage. In fact, in the original version of the algorithm, intermediary created solutions are often discarded. Our aim is to improve the acceptance rate and thus to accelerate the algorithm.

In the balance of this article, the following notations will be used: scalars are denoted in (lowercase or capital) italic letters, vectors in bold lowercase letters, matrices in bold capital letters. Tensors, cost functions and equations sets are in calligraphic uppercase letters. Finally, this article is organized as follows. After an introduction, the second section is dedicated to the statement of the CP decomposition problem. The principle of memetic algorithms and more specifically of the CEP algorithm as well as the various tricks on which it relies, are introduced in the third section. In the fourth section, we explain the principle of the parallel strategy used to improve the convergence rate and speed up the algorithm. Then, numerical simulations are provided to illustrate the robustness and effectiveness of the algorithm. Finally, a conclusion is drawn and perspectives are delineated.

II. Problem statement

A. The CP model

There exists numerous tensor decompositions (Tucker, Block-term, Tensor train). We focus, here, on the CP decomposition. This compact decomposition turns a tensor into a sum of rank-

1 tensors. Let $\overline{\mathcal{T}}$, a N-way tensor of size $I_1 \times I_2 \times \ldots \times\ I_N$ and assume its rank is R, its CP decomposition writes:

$$\overline{\mathcal{T}} = \sum_{r=1}^{R} \bar{\mathbf{a}}_r^{(1)} \otimes \bar{\mathbf{a}}_r^{(2)} \otimes \ldots \otimes \bar{\mathbf{a}}_r^{(N)} = [\![\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \ldots, \bar{\mathbf{A}}^{(N)}]\!], \tag{1}$$

with $\otimes$ the outer product or equivalently element-wise for all $(i_1, i_2, \ldots, i_N) \in \mathcal{I} = \{1, \ldots, I_1\} \times \{1, \ldots, I_2\} \times \ldots \times \{1, \ldots, I_N\}$

$$\bar{t}_{i_1,i_2,\ldots,i_N} = \sum_{r=1}^{R} \bar{a}_{i_1,r}^{(1)} \bar{a}_{i_2,r}^{(2)} \ldots \bar{a}_{i_N,r}^{(N)}, \tag{2}$$

where the N matrices $\bar{\mathbf{A}}^{(n)} = (\bar{a}_{i_n r}^{(n)}) = [\bar{\mathbf{a}}_1^{(n)}, \bar{\mathbf{a}}_2^{(n)}, \ldots, \bar{\mathbf{a}}_R^{(n)}] \in \mathbb{R}^{I_n \times R}$, with $n \in \{1, \ldots, N\}$, are the so-called *loading (or factor) matrices*, whose R columns $\bar{\mathbf{a}}_r^{(n)}$, with $r \in \{1, \ldots, R\}$, are the *loading vectors*. The tensor rank is unknown in most practical applications.

B. The CP decomposition

Given an observed tensor $\mathcal{T}$, the purpose of the CP decomposition is to find the best rank-one approximation of $\mathcal{T}$ (i.e. $\mathcal{T} \sim \overline{\mathcal{T}}$) and thus to estimate the loading matrices. A usual way is to minimise a well-chosen objective function $\mathcal{F}(\mathbf{x})$, which measures if the estimated CP model fits the observed tensor, $\mathbf{x}$ standing for the $L = (I_1 + \ldots + I_N) \times R$ unknowns written in a vector form

$$\mathbf{x} = \begin{pmatrix} \text{vec}\{\mathbf{A}^{(1)}\} \\ \vdots \\ \text{vec}\{\mathbf{A}^{(N)}\} \end{pmatrix} \in \mathbb{R}^L.$$

The operator vec$\{\cdot\}$ vertically stacks the columns of a matrix into a vector. The most commonly used objective function is the squared Euclidian distance between the estimations and the observations (fidelity criterium):

$$\begin{aligned} \mathcal{F}(\mathbf{x}) &= \sum_{(i_1,i_2,\ldots,i_N)\in\mathcal{I}} |t_{i_1,i_2,\ldots,i_N} - \sum_{r=1}^{R} a_{i_1,r}^{(1)} a_{i_2,r}^{(2)} \ldots a_{i_N,r}^{(N)}|^2 \\ &= \sum_{(i_1,i_2,\ldots,i_N)\in\mathcal{I}} |e_{i_1,i_2,\ldots,i_N}|^2 \end{aligned} \tag{3}$$

where $e_{i_1,i_2,\ldots,i_N}$ is the entry-wise error for each tensor term. Yet, the more terms are involved in the calculation of the cost function, the more computationally expensive this calculation can become. Moreover, in some applications (such as 3D or 4D fluorescence spectroscopy [9] [10]), the optimization is performed under non-negativity constraints because the loading vectors are standing for physical non negative quantities (spectra and concentrations).

III. A STOCHASTIC ALGORITHM

Nowadays, more and more stochastic optimisation algorithms are used (stochastic gradient descent, Monte Carlo methods, evolutionary algorithms, genetic algorithm (GA), and so on), especially in the field of Artificial Intelligence (see Convolutional Neural Networks (CNN) for example). In the tensor decomposition field, stochastic approaches have already been used either to create randomly smaller sub-tensors than the original problem [5] or to build randomly compressed cubes [15], but the optimisation algorithm often remains deterministic. Here, the optimisation algorithm itself is stochastic since we use a memetic algorithm. Such algorithms were introduced by Mocasto in [7]. As a metaheuristic technique, memetic algorithms are iterative. Like GAs, memetic algorithms are based on the convergence of all the elements of a population to the same solution. But to avoid the problem of convergence to a local extremum (which might occur with GA), a local search stage is added. Thus, each iteration *a.k.a a generation*, contains three main stages: a selection, a search and a replacement.

A. The CEP algorithm

We focus on the Collaborative Evolution of Population (CEP) algorithm, which has been adapted for CP decomposition in [6]. It consists of a competition between two candidates randomly picked among the population. Their respective cost functions are compared, the best candidate is kept and a new candidate is created around this one while the worst is discarded. In the context of CP decompositions, a candidate represents the set of all the unknowns *i.e.* the components of all the loading matrices. The principle of memetic algorithms for the minimization problem $\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$ where $\Omega \subset \mathbb{R}^L, f : \Omega \to \mathbb{R}$ can be summarized as follows:

Algorithm 1 Principle of the CEP algorithm

Step 1: Initialisation (at iteration $k = 0$) of a random population of $N \geq 2$ candidates $\{\mathbf{x}^1[0], \ldots, \mathbf{x}^N[0]\} \subset \Omega$. Compute $f(\mathbf{x}^1[0]), \ldots, f(\mathbf{x}^N[0])$ to sort the population.

Step 2: At iteration $k > 0$,

- Selection stage: choose, according to a uniform law, two different candidates in the population. The candidate with the smallest cost function is kept while the other one is discarded from the population.
- Local search: create a new candidate in the neighbourhood of the previously selected candidate
- Replacement: introduce this new candidate in the population

Step 3: Return to the Step 2 and set $k \to k + 1$ until the stopping criterium is reached

During the local search stage, we opt for a really simple rule. Only one component $x_{\bar{l}}$ of the selected candidate $\mathbf{x}$ is randomly picked ($\bar{l} \in \{1, \ldots, L\}$). It is modified by simply adding a quantity μ to its value (or $\mu[k]$ when iteration k is considered). Variable $x_{\bar{l}}$ corresponds to one single variable $a_{i_{\bar{l}};r_{\bar{l}}}^{(n_{\bar{l}})}$ among all the unknown parameters that have to be estimated which also means that the only term which is modified is at the position $(i_{\bar{l}}, r_{\bar{l}})$ in the $n_{\bar{l}}$-th loading matrix. So the elements of the different loading matrices $\tilde{a}_{i_n,r}^{(n)}[k+1]$

of the new candidate $\tilde{\mathbf{x}}[k+1]$, can be written as:

$$\tilde{a}^{(n)}_{i_n,r}[k+1] = \begin{cases} |a^{(n)}_{i_n,r}[k] + \mu[k]| & \text{if } n = n_{\bar{l}}, i_n = i_{\bar{l}} \text{ and } r = r_{\bar{l}} \\ a^{(n)}_{i_n,r}[k] & \text{otherwise} \end{cases} \tag{4}$$

The absolute value $|.|$ is here to ensure the imposed non-negativity constraint. The choice of the step size is crucial and is discussed in the next section. The main inconvenient of stochastic optimisation algorithms is that they are computationally expensive and converge after a huge amount of iterations.

B. Optimisation tools of the CEP algorithm

To reduce the computational cost of each iteration, two numerical tools have already been developed in [8]: 1) the use of a partial cost function and 2) the use of a more efficient calculation of the cost function.

1) Use of a partial cost function: In the context of CP decompositions, the number of tensor terms which is equal to $I_1 \times I_2 \times \ldots \times I_N$ is much more larger than the number of unknowns $(I_1 + I_2 + \ldots + I_N) \times R$. Therefore, all tensor elements are not taken into account in the calculation of the cost function to decrease its computational cost. Only M tensor elements are picked randomly making it possible to build a new set $\mathcal{I}_M \subset \mathcal{I}$, on which the partial cost function is effectively calculated:

$$\mathcal{F}_M(\mathbf{x}) = \sum_{\mathcal{I}_M \subset \mathcal{I}} |t_{i_1,i_2,\ldots,i_N} - \sum_{r=1}^{R} a^{(1)}_{i_1,r} a^{(2)}_{i_2,r}, \ldots, a^{(N)}_{i_N,r}|^2 \text{with}$$
$$\mathcal{I}_M = \{(i_1, i_2, \ldots, i_N) \in \mathcal{I} \mid \text{eq.}\,(i_1, i_2, \ldots, i_N) \text{ is chosen}\}.$$

Such a partial cost function is also very useful in the particular case of outliers or missing data. It makes it possible to not take them into account whereas a modified weighted cost function has to be introduced with most approaches considering block of unknowns [11] [12].

2) The choice of the step size: A too large step size μ in Eq. (4) may lead to divergence while a too small one may decrease the convergence speed. In the following, a stochastic step size is drawn from a uniform law $\mathcal{U}(-b[k], b[k])$. $b[k]$is calculated thanks to two heuristics detailed in [8]:

$$b[k] = \sqrt{\frac{\mathcal{F}(\mathbf{x}[k])}{M}} \Big/ \tau^{N-1} \tag{5}$$

$$\text{where } \tau = \sqrt[N]{\frac{\sum_{(i_1,i_2,\ldots,i_N)\in\mathcal{I}} t_{i_1,i_2,\ldots,i_N}}{MR}}$$

where the integer M corresponds to the number of equations among the $I_1 \times I_2 \times \ldots \times I_N$ available ones that are selected in (2). We can notice that $b[k]$ decreases when $\mathcal{F}(x[k])$ decreases.

3) Economic cost function calculation: In [8], it has been shown that a population of two candidates is more suited for the CP decomposition. So, the algorithm sums up in a local search around a candidate $\mathbf{x}[k]$ and if the new candidate $\tilde{\mathbf{x}}[k+1]$ is better, it is kept and the other discarded ($\mathbf{x}[k+1] = \tilde{\mathbf{x}}[k+1]$); otherwise, the new one is discarded ($\mathbf{x}[k+1] = \mathbf{x}[k]$) and a local search around the previous candidate is again performed. Since there is only one variable characterized by $(n_{\bar{l}}, i_{\bar{l}}, r_{\bar{l}})$ that is modified between the two candidates, only a few residual errors $e_{i_1,i_2,\ldots,i_N}[k+1]$ are impacted by this modification when $i_{n_{\bar{l}}} = i_{\bar{l}}$, although the others remain the same. Thus, the new modified residual errors can be calculated:

$$e_{i_1,\ldots,i_{\bar{l}},\ldots,i_N}[k+1] = e_{i_1,\ldots,i_{\bar{l}},\ldots,i_N}[k] - \mu[k] z^{(-n_{\bar{l}})}_{i_1,\ldots i_N, r_{\bar{l}}} \tag{6}$$

where

$$z^{(-n_{\bar{l}})}_{i_1,\ldots i_N,r_{\bar{l}}} = a^{(1)}_{i_1,r_{\bar{l}}} \ldots a^{(n_{\bar{l}}-1)}_{i_{n_{\bar{l}}-1},r_{\bar{l}}} a^{(n_{\bar{l}}+1)}_{i_{n_{\bar{l}}+1},r_{\bar{l}}} \ldots a^{(N)}_{i_N,r_{\bar{l}}}$$

The difference between the two cost functions $\mathcal{G}_M(\tilde{\mathbf{x}}[k+1], \mathbf{x}[k]) = \mathcal{F}_M(\tilde{\mathbf{x}}[k+1]) - \mathcal{F}_M(\mathbf{x}[k])$ can be computed by:

$$\mathcal{G}_M(\tilde{\mathbf{x}}[k+1], \mathbf{x}[k]) = \sum_{(i_1,\ldots,i_N)\in\mathcal{I}_{M,i_{n_{\bar{l}}}=i_{\bar{l}}}} [|e_{i_1,\ldots,i_N}[k+1]|^2 - |e_{i_1,\ldots,i_N}[k]|^2] \tag{7}$$

$\mathcal{I}_{M,i_{n_{\bar{l}}}=i_{\bar{l}}}$ is a subset of $\mathcal{I}_M$ where the tensor index $i_{n_{\bar{l}}}$ is fixed at $i_{\bar{l}}$.
The set on which $\mathcal{G}_M(\tilde{\mathbf{x}}[k+1], \mathbf{x}[k]$ is calculated, is then far smaller than $\mathcal{I}_M$. Therefore, a cheap computation of the new candidate cost function can be performed by $\mathcal{F}_M(\tilde{\mathbf{x}}[k+1]) = \mathcal{F}_M(\mathbf{x}[k]) + \mathcal{G}_M(\tilde{\mathbf{x}}[k+1], \mathbf{x}[k])$. If $\mathcal{F}_M(\tilde{\mathbf{x}}[k+1]) < \mathcal{F}_M(\mathbf{x}[k])$, then the update of the residual errors is taken in account otherwise not.

IV. A PARALLEL STRATEGY TO IMPROVE THE ALGORITHM

With the rise of "Big Data" tensors, the optimisation problems can become very large and the scale-up of the algorithms can be a real challenge. This raises issues linked to hardware constraints such as memory capacity and processor speed. To overcome these constraints, the asynchronous parallel methods have been considered for solving optimisation problems [13]. For tensor decomposition with non-negativity constraints, a parallel algorithm via Alternating Direction Method of Multiplier has already been studied in [14] and another approach consisting of analysing in parallel randomly compressed cubes has been suggested in [15].

A. Purpose of the parallel strategy used in the CEP algorithm

Due to their inherent "parallel nature", it is easy to apply a parallel strategy to memetic algorithms. In fact, they rely on the notion of a population of candidates which means that the cost function of each candidate can be calculated separately. Parallel memetic algorithms have already been considered in [16] but for other problems.
In the context of the CEP algorithm, a parallel strategy is introduced in order to improve the acceptation rate after the local search milestone. A "reject" instead of an "acceptation" results from the fact that the new created candidate often has a cost function larger than the one of the previous candidate. In some situations, the rejection rate can be close to 100% (in the following example, it was roughly equal to 98%). The aim

is to improve the acceptation rate, so the convergence speed of the algorithm will increase. In our case, the parallel part of the algorithm is coded thanks to OpenMP.

B. Description of our parallel strategy

We start our algorithm with just one candidate. At each iteration, a certain number *thread* of new candidates are created independently in parallel during the local search stage in the neighbourhood of the previous one. More precisely, we apply a different step size on a randomly picked variable thanks to Eq. (4), for each new candidate. After comparing the value of their cost function, the candidate with the smallest cost function is kept (or the previous candidate if no new candidate is able to improve the value of the cost function). At each iteration, the probability to find an “acceptable” new candidate increases and thus, we are more likely to improve the acceptation rate even if each iteration becomes more time consuming.

Evaluating the cost function value after each move consumes about the same time because we can consider that the number of selected tensor terms related to each variable is in the same order of magnitude. Therefore, each thread created in the parallel section is associated with a new candidate. The resulting algorithm can be summed up as follows:

Algorithm 2 Principle of a parallel strategy for the CEP algorithm

```
Step 1: Initialisation (iteration $k = 0$) of a random candidate $\mathbf{x}[k]$ and compute $\mathcal{F}_M(\mathbf{x}[k])$
Step 2: At iteration $k > 0$,
  • Parallel section (local search stage):
      – Create a population of thread new candidates $\tilde{\mathbf{x}}_0[k+1], \ldots, \tilde{\mathbf{x}}_{thread\text{-}1}[k+1]$ from $\mathbf{x}[k]$ with different moves (variable that is moved and the used step size)
      – Compute $\mathcal{F}_M(\tilde{\mathbf{x}}_0[k+1]), \ldots, \mathcal{F}_M(\tilde{\mathbf{x}}_{thread\text{-}1}[k+1])$
      – Keep the best candidate $\tilde{\mathbf{x}}^*[k+1]$ with its cost function value $\mathcal{F}_M(\tilde{\mathbf{x}}^*[k+1])$
  • Replacement: if $\mathcal{F}_M(\tilde{\mathbf{x}}^*[k+1]) < \mathcal{F}_M(x[\mathbf{x}])$ , then $\mathcal{F}_M(\mathbf{x}[k+1]) = \mathcal{F}_M(\tilde{\mathbf{x}}^*[k+1])$ and $\mathbf{x}[k+1] = \tilde{\mathbf{x}}^*[k+1]$ otherwise the candidate and its cost function value remains the same (rejection case)
Step 3: Return to the Step 2 and set $k \to k+1$ until the stopping criteria are reached
```

V. Simulations and Comparison

The application targeted, here, is the 3D fluorescence spectroscopy. This kind of data set constitutes a 3-way tensor in which are saved emitted light intensities for different excitation wavelengths and for several observations. The tensor is a juxtaposition of the so-called Fluorescence Emission-Excitation Matrices (FEEM). This application is interesting, because, at low concentrations of fluorophores (fluorescent components that are present in the different observations/mixtures), the non-linear model predicted by the Beer-Lambert law can be linearized and it has been shown that it follows the CP model [9]. The columns of the loading matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ stand for physical quantities: excitation and emission spectra of the different fluorophores and the evolution of the relative concentration of each fluorophore along the different experiments. In this case, the tensor rank R in the CP model, corresponds to the number of fluorophores that are present in the studied solutions. Since we are dealing with intrinsically non-negative data (spectra, concentrations), it is quite relevant to impose non-negativity constraints during the optimization scheme. In this section, all the considered tensors are simulated synthetic fluorescence spectroscopy like data.

The Relative Reconstruction Error (RRE) is defined as:

$$\text{RRE} = \frac{||\mathcal{T} - \tilde{\mathcal{T}}||_F^2}{||\mathcal{T}||_F^2} \text{ and } \text{RRE}_{dB} = 10\ log_{10}(\text{RRE}) \qquad (8)$$

where $|| \cdot ||_F$ stands for the Frobenius norm. The stopping criteria for each experiments are RRE$< 10^{-8}$ with a maximum number of iterations 2.4×10^8 and for all memetic algorithms an another stopping criterium is added: all 40 000 iterations the ratio between the two values of cost function $\frac{\mathcal{F}_M(\mathbf{x}[k])}{\mathcal{F}_M(\mathbf{x}[k-40000])} <$ 0.99999. For all tested parallel strategies, the relative reconstruction errors and the CPU time are recorded each 40000 iterations, although in the classic case, each 400000 iterations. In the following, the results for parallel strategies are averaged on five executions because the random generator behaviour is different at each execution.

To compare the performances of the different algorithms, the studied tensors, in this example, are $100 \times 100 \times 100$ and their rank is 5. Only $M = 10 \times L$ (with L the number of latent variables *i.e.* 1500) tensor terms are taken into account for the partial cost function.

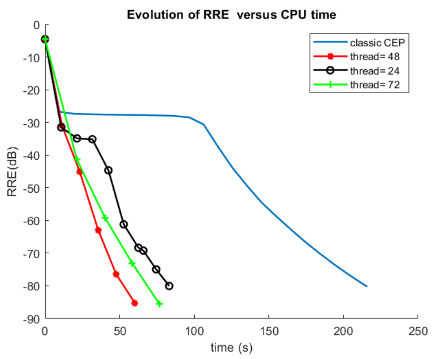

Figure 1. Performance vs CPU time for different number (*thread*) of candidates created in parallel

In Fig. 1 and Tab. I, we clearly observe that the application of a parallel strategy on the CEP algorithm allows to reduce the computation time. Moreover, all variants of the algorithm were able to reach the true solution as it appears on the Figure 3. The best choice among the different values of *thread* seems to be here *thread* = 48 for 96 available processors.

The lasting time between two records becomes shorter when the number of thread is decreasing and inversely. It makes

Table I

COMPARISON OF THE MEAN TIME CONSUMED DURING 40000 ITERATIONS AND MEAN NUMBER OF ACCEPTATIONS BY ITERATION VERSUS THE NUMBER OF THREADS CREATED IN THE PARALLEL SECTION

Number of threads	24	48	72
Mean time during 40 000 iterations	9.24s	12.05s	19.11s
Mean number of acceptations	17.88%	19.58%	22.49%

sense because *thread* cost function values for the new created candidates are computed at each iteration.

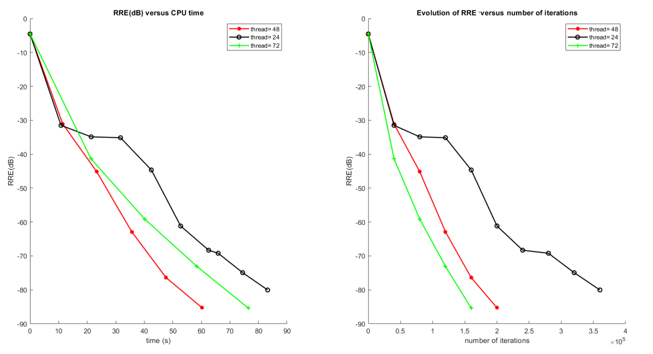

Figure 2. Impact of the number of new created candidates *thread* on the perf.

As illustrated on Fig. 2, the parallel strategy version with *thread* $= 72$ leads to convergence in less iterations. Indeed, the more created threads there are, the less iterations to achieve the converge are required. However, this version of the algorithm is not the fastest since each iteration takes much more time to be completed. As a conclusion, there is a compromise to find between the time taken by each iteration and the total number of iterations required to reach the stopping criteria but with this kind of stochastic algorithm it remains obviously advantageous to opt for a parallel strategy.

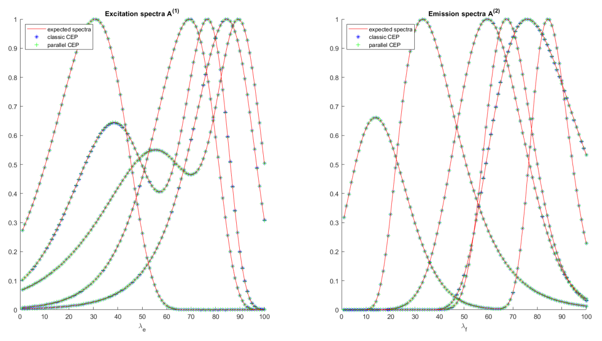

Figure 3. Comparison of the emission and excitation spectra estimated with the different variants of the algorithm superimposed with the ground truth spectra.

VI. CONCLUSION & PERSPECTIVES

The parallel strategy for the CEP algorithm used to solve the CP decomposition problem is less time consuming than the standard version of the CEP algorithm (about 3,6 times) while the same accuracy can be reached. However, we have shown that the number *thread* of new candidates must be carefully chosen. In [8], a locally optimal step size strategy has been developed on a random variable to reduce the rejection rate in the original version of the CEP algorithm. Our aim will be now to use both this parallel strategy and the locally optimal step size strategy. New candidates will be created by calculating and applying this "optimal step size" for different variables picked in parallel. Finally, many other parallel strategies could be considered, among which are the application of all the accepted moves on the previous candidate, or the use of the second best move in the previous iteration at the local search stage in the parallel region to avoid the loss of important acceptances.

REFERENCES

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications", SIAM Review, 51:3, pp. 455-500, 2009.
[2] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products", Journal of Mathematics and Physics, 6, 1927.
[3] A. Cichocki, R. Zdunek, A. H. Phan, and S. I. Amari, "Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation". Wiley Publishing, 2009.
[4] X. T. Vu, C. Chaux, N. Thirion-Moreau, S. Maire and E. M. Carstea "A new penalized nonnegative third order tensor decomposition using a block coordinate proximal gradient approach: application to 3D fluorescence spectroscopy" in *Journal of Chemometrics*, Wiley & Sons, special issue on penalty methods, Vol. 3, CEM 31.1, March 2017.
[5] N. Vervliet, L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors". In IEEE Journal of Selected Topics in Signal Processing 10.2, pp. 284–295, 2016.
[6] X. T. Vu, S. Maire, C. Chaux and N. Thirion-Moreau, "A new stochastic optimization algorithm to decompose large nonnegative tensors", in IEEE Signal Processing Letters, vol. 22, no. 10, pp. 1713-1717, Oct. 2015.
[7] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts - Towards memetic algorithms". Caltech Concurrent Computation Program, Report. 826, 1989.
[8] S. Maire, X. Vu, Caroline Chaux, C. Prissette, N. Thirion-Moreau. "Decomposition of large nonnegative tensors using memetic algorithms with application to environmental data analysis". 2019. hal-02389879
[9] J. R. Lakowicz, "Principles of fluorescence spectroscopy", 3rd edition Springer ISBN: 978-0-387-31278-1 Hardcover, 2006.
[10] J.P. Royer, N Thirion-Moreau, P. Comon, R. Redon and S. Mounier "A regularized nonnegative canonical polyadic decomposition algorithm with preprocessing for 3D fluorescence spectroscopy", Journal of Chemometrics 29 (4), 253-265, March 2015
[11] J.-P. Royer, N. Thirion-Moreau, P. Comon, "Nonnegative 3-way tensor factorization taking into account possible missing data", in Proc. *EUSIPCO*, pp. 71-75, Bucharest, Romania, August 2012.
[12] G. Tomasi, R. Bro, "PARAFAC and missing values", Chemometrics and Intelligent Laboratory Systems, Vol. 75, Issue 2, pp. 163-180, 2005.
[13] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. "Asynchronous parallel algorithms for nonconvex big-data optimization: model and convergence". ArXiv, abs/1607.04818. 2016.
[14] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers", in IEEE Trans. on Signal Proc., vol. 63, no. 20, pp. 5450-5463, Oct. 2015.
[15] N. D. Sidiropoulos, E. E. Papalexakis and C. Faloutsos, "A parallel algorithm for big tensor decomposition using randomly compressed cubes (PARACOMP)", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 1-5, 2014.
[16] J. Tang, M. H. Lim, Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems - Soft Computing", Springer, 2007.