A Comparative Study of the EEG Signals Big Optimization Problem using Evolutionary, Swarm and Memetic Computation Algorithms

M. A. El Majdouli
Conception & Systems Laboratory
Mohammed V University
Rabat, Morocco
elmajdouli@acm.org

S. Bougrine
Conception & Systems Laboratory
Mohammed V University
Rabat, Morocco
saad.bougrine@acm.org

I. Rbouh
Conception & Systems Laboratory
Mohammed V University
Rabat, Morocco
rbouh.ismail@ieee.org

A. A. El Imrani
Conception & Systems Laboratory
Mohammed V University
Rabat, Morocco
elimrani@fsr.ac.ma

ABSTRACT

This paper investigates the optimization of EEG signals cleaning process by elaborating a comparative study of swarm intelligence, evolutionary and memetic computation techniques. In this context, algorithms from each technique have been selected notably Clonal Selection, Particle Swarm Optimization, Firefly Algorithm, Harmony Search and Fireworks Algorithm. Each algorithm performance has been analyzed and validated by experiments conducted over the CEC Big-OPT EEG datasets. The results conclude a competitive performance of evolutionary and memetic methods in comparison to the swarm intelligence methods.

CCS CONCEPTS

• **Theory of computation** → Evolutionary algorithms • **Applied computing** → Health informatics • **Hardware** → Digital signal processing

KEYWORDS

Medical Data Optimization, Evolutionary Computation, Swarm Intelligence, Big-OPT, EEG Signals

GECCO '17 Companion, July 15-19, 2017, Berlin, Germany

ACM ISBN 978-1-4503-4939-0/17/07…$15.00

DOI: http://dx.doi.org/10.1145/3067695.3082489

1 INTRODUCTION

Most Noninvasive Brain Computer Interfaces (BCI) [1] measure brain activity using special sensors placed on the head to record Electroencephalogram (EEG) signals [2] that allow neurobiologists to monitor the brain electric activity. Recent advances focus on analyzing and correctly interpreting the EEG signals. In one hand, they deliver essential information used by specialists to understand several interesting neurobiological disorders of great importance. For instance, understanding and treating epilepsy which affects 1% of the world's population [3] can have not only a high medical and humanitarian impact, but also an enormous economic opportunity for medical and drugs companies. In the other hand, EEG signals can be interpreted and translated to computer tasks which open wide doors to integrate BCIs into potential applications in the public markets, notably gaming industry; Virtual Reality for in-stance; or military programs. However, achieving such levels is faced with several difficulties. A real-time BCI is characterized to be highly time dependent and captured signals may vary in a milliseconds scale. Also, using many sensors not only raises the amount of recorded data but also question the accuracy rate of each sensor when their manufacturing is not the same. Additionally, captured signals are noised by non-brain electrical sources which affect further the accuracy and augment the processing time. These difficulties can define the 4'v characterizing the Big Data concept [4], which leads to considering the EEG signals as an instance of Big Data.

In the Optimization of Big Data 2015 Competition [5] [6], the problem is modeled as an NP-Hard Big Data optimization problem (Big-Opt). The dimensionality of this problem can be very high as the number of handled variables can reach the multiple of the length of EEG signals by the number of channels used in the capture operation. Besides handling such high dimensionality, the optimization solving algorithm should also

have a rapid convergence time, as EEG signals require a real-time processing.

Many real world optimization problems like Big-Opt are naturally complex and hard to solve using mathematical exact approaches, as they involve complex structures and a very large search space, in which, even exhaustive and random heuristic approaches fails to provide the optimal solution in an acceptable time. Meta-heuristics suggest a compromised approach aiming to deliver good solutions in a reasonable time. This approach is mainly ruled by two mechanisms in order to discover promising regions in the search space: Exploration of the search space and Exploitation of the local space nearby a given solution. Nature has been a great source of inspiration to implement these mechanisms. The evolutionary process of beings reported in the Darwin principles that suggest the survival of the fittest using mutation, crossover and reproduction, started a wide track of optimization known as evolutionary computation. Another important model of optimization solving is the swarm computation. In this approach, the optimization task takes advantage of the group intelligence developed by beings like ants and birds. Memetic computation is the most recent model in optimization. It somehow takes advantage of the evolutionary and swarm computation without being limited to the natural phenomenon. It mainly consists on imitating any real life phenomenon that might serve as a good optimization process that efficiently implements the exploration and exploitation mechanisms. It is mostly represented as an incorporation of a local search method into a global search optimization method such us an evolutionary algorithm.

The goal of this paper is to study the performance of several NP-Hard optimization techniques on solving the Big-Opt problem. As only few works have tackled Big-Opt, this paper delivers a comparative study using some relevant algorithms in the evolutionary, swarm and memetic computation fields, notably Clonal Selection, Particle Swarm Optimization, Firefly Algorithm, Harmony Search and Fireworks Algorithm. The paper will help to provide researchers with a ground truth about the performance of these algorithms to solve Big-opt and also a starting point to ameliorate and enhance the solving process using these algorithms. Additionally, this paper intends to strengthen the motivation for solving Big-OPT-like instances and to help other researchers from outside the optimization field, e.g. medical field researchers, on choosing the appropriate optimization method.

The rest of the paper is organized as follows: Section 2 provides a description of the Big-Opt problem. Section 3 describes the algorithms used in the comparative study. Section 4 details the used solution encoding and initialization. The experimental results are reported in Section 5 at which we conclude right after.

2 BIG-OPT PROBLEM: BACKGROUND & RELATED WORKS

Although several methods to monitor and record the brain activity has been proposed like functional magnetic resonance imaging (fMRI) and magnetoencephalographic signal (MEG), the EEG signal is still the most used because of its convenience to the patients and its economic cost [7]. The challenge in EEG signals analysis lies in successfully recovering true brain signals of a given activity that is noised by some artifacts due to both internal and external factors. While external factors can be summarized in non-brain electric activity sources, the internal factors concern involuntary movements such as eyes blinking or undesirable imagined movements that generate an undesirable electrical brain activity. To remove these artifacts, the signals data is passed through an Independent Component Analysis [8]. The reconstruction is supposed to be performed in real time with the sensors' recording. Considering the high dimensionality of the signals, the reconstruction becomes a time-consuming task. To automatize the noise cleaning process, the problem is mathematically expressed as a Big Data optimization problem.

Let X a matrix of dimension $N \times M$, where N is the number of inter-dependent time series (signals), and M is their length. Assume S, an $N \times M$ matrix with N independent time series of length M such that:

$$X = A \,.\, S \qquad (1)$$

where A is a linear transformation matrix of $N \times N$ dimension. The problem is to decompose S into two matrices S_1 and S_2 of the same dimensionality as S such that:

$$S = S_1 + S_2 \qquad (2)$$

$$X = A \,.\, S_1 + A \,.\, S_2 \qquad (3)$$

The Pearson correlation matrix C between S_1 and X is given by:

$$C = \frac{cov(X, A.S_1)}{var(X).var(A.S_1)} \qquad (4)$$

where *cov(.)* is the covariance matrix and *var(.)* is the variance.

The goal is to find S_1 similar to S where the distance between the two matrices is as minimal as possible. Also, the off-diagonal elements of C should be minimized while the on-diagonal elements have to be maximized. This mathematical expression gave two formulations of the problem. The first one is a multi-objective optimization problem that maximizes two functions f_1 and f_2 such that:

Given S, X and A, find S_1 that minimizes:

$$min \;\; f_1 = \frac{1}{M.N} \sum_{i,j} (S_{ij} - S_{1,ij})^2 \qquad (5)$$

$$min \;\; f_2 = \frac{1}{N^2 - N} \sum_{i,j \neq i} (C_{ij}^2) + \frac{1}{N} \sum_i (1 - C_{ii})^2 \qquad (6)$$

The second formulation is a single objective optimization problem that optimizes:

$$f = min \; (f_1 + f_2) \qquad (7)$$

In order to solve the single objective problem of Big-Opt, some techniques have been used. For instance, Zhang et al. [9] proposed a multi-agent genetic algorithm with redesigned competition and self-learning operators to solve the single objective Big-Opt problem. They combined them with crossover and mutation operators to simulate the learning behavior of agents. An adaptive configuration of differential evolution algorithms (DEA) was introduced in [10]. The idea proposed to

run three variants of DEA in parallel, and evolve only the best performing variant. Elsayed et al. [11] extended the same idea to a differential evolution framework to achieve better results. They mainly focused on analyzing and tuning the differential evolution algorithms parameters.

3 EVOLUTIONARY, SWARM AND MEMETIC COMPUTATION FOR OPTIMIZATION PROBLEMS

In this section, we describe the Immune Clonal Selection algorithms as an evolutionary computation algorithm, Particle Swarm optimization and the Firefly Algorithm to represent the Swarm intelligence computation and Fireworks Algorithm along with Harmony Search for the Memetic computation.

3.1 Immune Clonal Selection

The clonal selection algorithm "*CSA*" [12] is inspired by the clonal selection theory of the natural immune system to perform an optimization process. This latter is mainly based on the idea of proliferation of the *B* cells depending on their maturity degree called affinity. That means the higher the affinity the more clones are produced. Additionally, the mutation rate that a given antibody could suffer is inversely proportional to its parent affinity. In this paper, a fix number of clones is used to preserve a search diversity, along with an inversely proportional hyper-mutation operator, allowing *B* cells with low affinity which is a fitness function value to undergo a high mutation rate. The pseudo-code of the implemented clonal selection algorithm is given below.

```
Begin
 Initialize a Memory Set M with initial Antibodies.
 Initialize the number of clones k.
 While Stopping condition is false do:
  For each Antibody in M do:
     Determine the affinity of the antibody.
     Generate k clones of the current antibody.
     Mutate attributes of these clones.
  End for
  Replace the n lowest affinity antibodies in M with n highest
  affinity mutated antibodies.
End while
End
```

3.2 Particle Swarm Optimization

The Particle Swarm Optimization "*PSO*" is a very popular technique in stochastic optimization [13]. Developed by Eberhat and Kennedy, *PSO* essentially imitates the birds flock and simulates their flying pattern. It exploits the swarm information to update a position *X* of each particle of the swarm. Indeed, each particle updates its velocity *V* according to its best known location P_{best} and the best known location found by the whole swarm G_{best} as described in *eq.8* and *eq.9* where α, C_1 and C_2 are weighting parameters. In this study, the implemented pseudo-code of *PSO* is described below.

$$V_{i+1} = V_i + \alpha . (C_1 . (P_{best} - X_i) + C_2 . (G_{best} - X_i)) \quad (8)$$

$$X_{i+1} = X_i + V_{i+1} \quad (9)$$

```
Begin
 Initialize c1, c2 and α
 Initialize Swarm
 While stop criteria is false do:
    For each particle P in the Swarm do:
      Evaluate P.
      Update the best known location of P. (P_best)
      Update the best known location of the Swarm. (G_best)
    End for
    For each particle P in the Swarm do:
        Calculate the new velocity using eq.8.
        Calculate the new position using eq.9.
    End for
 End while
End
```

3.3 Firefly Algorithm

```
Begin
Initialize a population of n fireflies.
Define maximum number of iterations Max iterations
Calculate the light intensity I_f for each Firefly f.
Define light absorption coefficient γ
While t < Max iterations do:
 For i = 1: n do:
   For j = 1: n do:
     If (I_j > I_i)
        Move firefly i towards j.
     End if
     Evaluate new fireflies' found solutions.
     Update light intensity.
   End for j
 End for i
 Update the best firefly.
End while
End
```

Presented by Yang [14], the Firefly Algorithm "*FA*" is another swarm intelligence algorithm based on imitating the fireflies' patterns and social behavior. Roughly speaking, *FA* is ruled by two essential ideas. The first one develops the attractiveness between fireflies. As fireflies are unisex, each firefly is proportionally attracted to another depending on its brightness only. In other words, less bright fireflies will update their position by moving towards brighter ones using. The algorithm defines the

brightness of a firefly by using the fitness function value of the current firefly position. The second idea is the attractiveness variation which depends on the separating distance between two fireflies. The algorithm above describes the pseudo-code used in our comparison study.

3.4 Harmony Search

Harmony Search "*HS*" algorithm described in the pseudocode below is a metaheuristic based on the concept of mimicking a skilled musician improvisation [15]. Indeed, three possible choices are available to the improvising musician: 1. Playing an already memorized tone, 2. Playing something similar to the memorized tone or 3. Composing a random tone. *HS* introduce the notion of the Harmony Memory *HM*. In order to store a harmony in the *HM*, an accepting rate r_{accept} in [0,1] is used. To formalize a similar tone improvising, a pitch adjustment component is defined by using the b_{range} and a pitch adjusting rate r_{pa}. Now, if the harmony is to be modified, *eq.10* is used to update it.

$$x_{new} = x_{old} + (b_{range} \times \varepsilon) \quad (10)$$

Begin
Initialize **Harmony Memory** ***HM*** with random harmonies
Initialize harmony memory accepting rate $\boldsymbol{r_{accept}}$
Define pitch adjusting rate $\boldsymbol{r_{pa}}$ and $\boldsymbol{b_{range}}$
While ***Stopping condition*** is ***false*** do:
  Initialize an empty ***harmony h***
  For each dimension ***i*** in ***h***
    Generate a random number $\boldsymbol{N_{rand}}$
    If ($\boldsymbol{N_{rand}} < \boldsymbol{r_{accept}}$),
      Choose the $\boldsymbol{i^{th}}$ value of ***the best harmony in HM***
      If ($\boldsymbol{N_{rand}} <$ ***rpa***),
        Adjust the value using ***eq.10***
      End if
    Else Choose a random value from ***HM***
    End if
  End for
  Evaluate ***h***
  Rank ***HM*** Harmonies
  Find the $\boldsymbol{1^{st}}$ ***harmony k*** in ***HM*** with a lower fitness than ***h***
  If found
    Replace ***k with h*** in ***HM***
  End if
End while
End

3.5 Fireworks Algorithm

Fireworks Algorithm (***FWA***) [16] is a memetic approach inspired by fireworks and their explosion process. Actually, the resulted sparks generated by the explosion process are interpreted as solutions revealed by a search mechanism over the local space nearby the explosion's location. During each generation of an explosion, *n* locations (solutions) are selected wherein *n* fireworks are set off. Sparks locations are calculated after the explosion triggering. The next generation explosion locations are selected from the generated sparks and the current fireworks locations relatively to their fitness value. The process runs iteratively until a stopping criterion is satisfied.

To correctly mimic the sparks generation process, many control parameters are defined such as the number of generated sparks for each firework, and the explosion amplitude.

Begin:
Initialize the ***set of Fireworks***
Initialize the ***best location*** Sol_{best}
While ***Stopping condition*** is ***false*** do:
  Initialize Â
  Initialize the number of sparks $\boldsymbol{S_n}$ to generate for all fireworks
    For each Firework $\boldsymbol{x_i}$ do:
      Calculate ***the amplitude*** $\boldsymbol{A_i}$ of x_i using *eq.11*
      For j : 1 ... S_n do:
        Initialize a Spark location $\boldsymbol{y}$ with $\boldsymbol{x_i}$
        Calculate the displacement $\boldsymbol{h_i}$ using $\boldsymbol{x_i}$ amplitude:
        $\boldsymbol{h_i = A_i}$ ***. rand (-1,1)***
        Generate uniformly a ***random k*** : $1 < k < d$
        For j : 1 ... k do:
          Select a ***random dimension t***.
          $\boldsymbol{y_t = x_t + h_i}$
        End For
      End for
    End for
    Evaluate all Fireworks and generated Sparks.
    Update the ***best location*** Sol_{best}
    Select the ***best Firework location*** F_{best}
    Select the ***best Spark location*** SP_{best}
    Initialize the ***next generation*** starting location:

$$ST_{next} = \begin{cases} F_{best} & if\ SP_{best} > \alpha\ .\ F_{best} \\ \square & \\ SP_{best} & Otherwise \end{cases}$$

    For each next generation Firework $\boldsymbol{x_i}$ do:
      $\boldsymbol{x_i = ST_{next}}$
    End For
End while
Stop

These parameters control the search process and the explorative behavior of the algorithm. The *FWA* algorithm used in our study, a fix number of sparks is defined for all fireworks in order to promote diversity. As for the explosion amplitude for each firework, the equation is given by:

$$A_i = \hat{A} \times \frac{f(x_i) - y_{min} + \varepsilon}{\sum_{i=1}^{n} f(x_i) - y_{min} + \varepsilon} \quad (11)$$

where $\hat{A}$ is the maximum explosion amplitude, and represents the best value of the objective function:

$$y_{min} = min\ f(x_i) \quad ; \quad i = 1, 2, 3, \ldots, n \quad (12)$$

4 SOLUTION ENCODING & INITIALIZATION

For all algorithms used in this study, solutions are represented using a matrix scheme where each decision variable is real encoded and is mapped using a couple *(i,j)*, representing its row and column position. The size of each solution is the number of dimensions in the decision space.

The problem of Big-OPT aims to find a matrix S_1 similar to S. the similarity is measured using the second norm distance d_2. This means:

$$d_2(S_1, S) = 0 \;\Rightarrow\; S_1 \equiv S \qquad (13)$$

The initialization process is then facing two choices: Random initialization of solutions with uniformly distributed values in the decision variables ranges or initialize vectors using the values of S, as S is the optimal solution of the minimization problem of the first objective function. The pseudo-code below is used to initialize each algorithm where $Vector(rand(-1,1))$ returns a vector of size equal to the number of the problem dimension, initialized with values in [-1,1] . We distinguish two cases. In the first case, a starting solution ST is initialized with random values in [-1,1]. In the second case, ST is initialized around the values of S. In this study, we have tested both scenarios, however, due to space limitation of the paper, we will present the results collected by the first case only. The initialization process is described below.

```
Begin
Initialize the population (or Memory) with its corresponding size
Initialize a constant ε
For each solution x_i do:
        Initialize x_i :  x_i = Vector(rand(−1, 1)) x ε
End For
End
```

During the initialization tests of all algorithms, the successful parameter initializations are set as reported in the Table 1. It is noticed that for evolutionary and memetic algorithms, the selected parameters orient the search to be more exploitative than explorative. As for swarm intelligence methods *PSO* and *FA*, various configurations have been tested. The best ones essentially required a weak weighting on the local best position and a strong weighting on the global best position. We've noticed also that the fitness function minimization is continuously achieved using infinitesimal changes with small mutation amounts in evolutionary and memetic methods.

Table 1: Parameters initialization

Algorithm	Parameters
CSA	*Iterations* = 10000 ; *Memory Set* Size = 3 ; k = 2 ; *Hyper-mutation* ρ = 2
PSO	*Iterations* = 10000 ; *Swarm Size* = 5 ; α = 0.07 ; $C_1 = C_2 = 2$
FA	*Iterations* = 10000 ; *Swarm Size* = 5 ; $\beta_0 = 1$; $\gamma = 3$
HS	*Iterations* = 50000 ; HM Size = 15; r_{accept} = 0.5 ; r_{pa} = 0.5 ; ε = 0.05
FWA	*Iterations* = 30000 ; *Population Size* = 2 ; m = 2 ; $\hat{A}$ = 0.02

5 EXPERIMENTAL RESULTS & DISCUSSION

5.1 Datasets & test protocol

To compare each method's performance, six datasets of different sizes are used as described in the table below:

Table 2: Big-OPT instances details

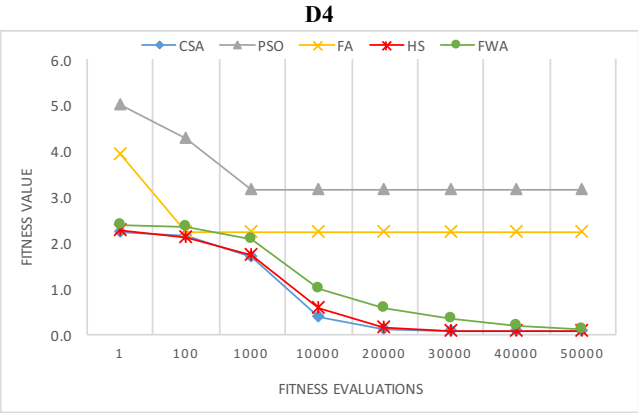
Dataset instance	D4	D4N	D12	D12N	D19	D19N
# channels	4	4	12	12	19	19
Data length	256	256	256	256	256	256
Size	1024	1024	3072	3072	4864	4864
Noise	No	Yes	No	Yes	No	Yes

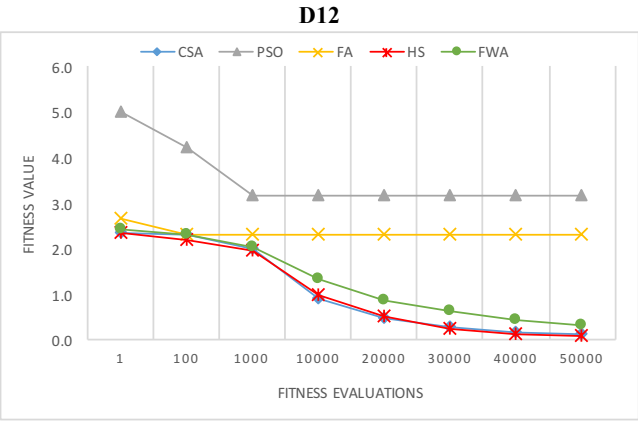
The experiments are executed on a laptop with a *2.3 GHZ CPU* and *8 GB of RAM*. No parallel computing has been performed. To avoid the "random effect", experiments are repeated ten times on each dataset. Each algorithm has been allowed a maximum of *50000 fitness function evaluations* as a stopping criterion.

5.2 Detailed results

The figures Fig.1.a and Fig.1b below shows respectively the convergence of the used algorithms in noise-free and noised datasets. It is clear that both swarm intelligence algorithms *PSO* and *FA* could not converge to a good optimum in comparison to *CSA*, *HS* and *FWA*.

Figure 1.a: Fitness convergence of each algorithm on noise-free datasets.





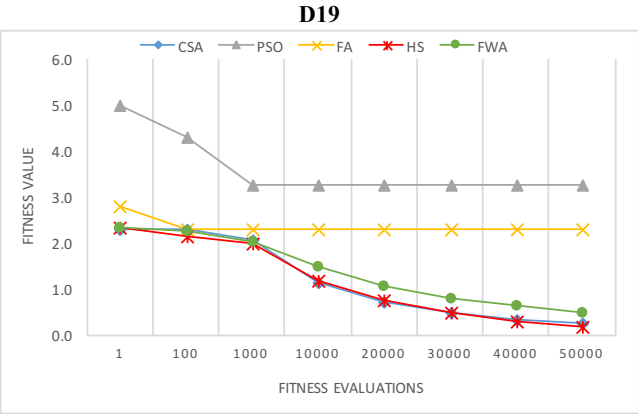
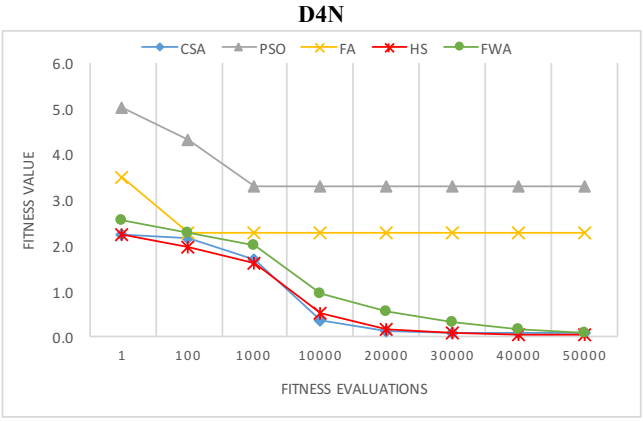
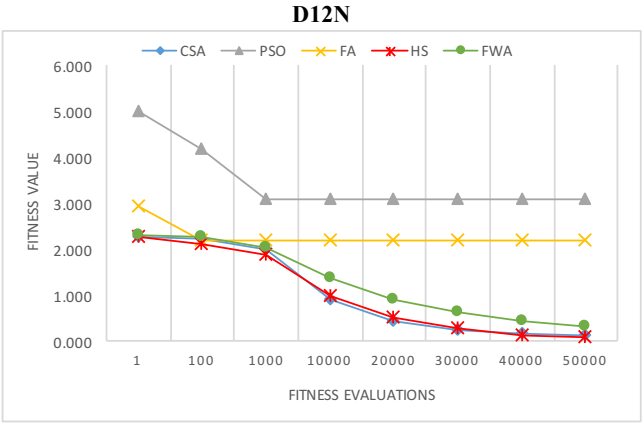


Figure 1.b: Fitness convergence of each algorithm on noised datasets.





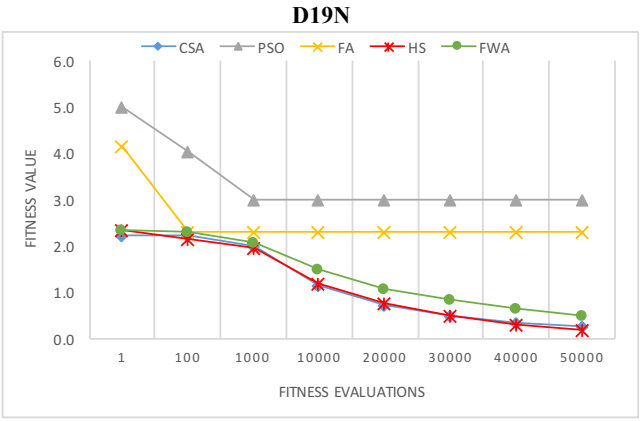


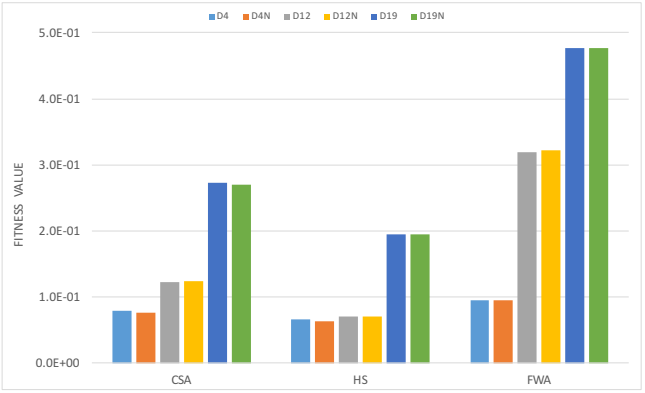
Table 3: Detailed results on all datasets

	Alg.	Best	Average	SD	Avg. Time (s)
	CSA	7.75E-02	7.86E-02	6.34E-04	8.09E+00
	PSO	2.86E+00	3.12E+00	1.23E-01	7.82E+00
D4	FA	2.13E+00	2.25E+00	5.34E-02	**4.81E+00**
	HS	**6.57E-02**	**6.59E-02**	**1.16E-04**	1.28E+01
	FWA	9.00E-02	9.52E-02	2.31E-03	6.40E+00
	CSA	7.58E-02	7.64E-02	4.62E-04	8.07E+00
	PSO	2.89E+00	3.07E+00	1.33E-01	7.81E+00
D4N	FA	2.18E+00	2.26E+00	3.85E-02	**4.80E+00**
	HS	**6.37E-02**	**6.39E-02**	**1.76E-04**	1.25E+01
	FWA	8.89E-02	9.49E-02	2.93E-03	6.64E+00
	CSA	1.18E-01	1.22E-01	3.39E-03	4.01E+01
	PSO	3.05E+00	3.15E+00	6.24E-02	4.69E+01
D12	FA	2.25E+00	2.29E+00	2.73E-02	**3.04E+01**
	HS	**6.70E-02**	**7.05E-02**	**2.58E-03**	5.42E+01
	FWA	3.12E-01	3.19E-01	5.56E-03	3.28E+01
	CSA	1.17E-01	1.23E-01	**2.95E-03**	4.02E+01
	PSO	2.94E+00	3.15E+00	9.63E-02	4.84E+01
D12N	FA	2.24E+00	2.29E+00	2.51E-02	**3.13E+01**
	HS	**6.42E-02**	**7.03E-02**	3.27E-03	5.59E+01
	FWA	3.09E-01	3.22E-01	6.93E-03	3.40E+01
	CSA	2.63E-01	2.72E-01	**6.01E-03**	9.55E+01
	PSO	3.10E+00	3.20E+00	6.16E-02	1.02E+02
D19	FA	2.25E+00	2.30E+00	2.25E-02	7.68E+01
	HS	**1.80E-01**	**1.95E-01**	7.22E-03	1.15E+02
	FWA	4.66E-01	4.77E-01	6.92E-03	**7.44E+01**
	CSA	2.65E-01	2.69E-01	**2.75E-03**	9.51E+01
	PSO	3.14E+00	3.21E+00	4.68E-02	1.08E+02
D19N	FA	2.27E+00	2.30E+00	2.49E-02	7.80E+01
	HS	**1.89E-01**	**1.94E-01**	5.96E-03	1.20E+02
	FWA	4.71E-01	4.77E-01	4.66E-03	**7.54E+01**

Indeed, *HS* and *CSA* record almost the same performance with a slight difference to *FWA* while both *PSO* and *FA* suffer from stagnation. Furthermore, the Table 2 presents the detailed results of all algorithms in all datasets. *CSA* and *HS* show a very stable performance as the standard deviation *SD* is very low in the order of 10^{-4}. *FWA* also scores the third good performance while *PSO* and *FA* record bad optimums in all datasets. To illustrate the results presented in Table 1, the figures below compare the average solutions of all methods.

As swarm intelligence methods performance is very weak, the Fig.2 compares only the evolutionary technique "*CSA*" to memetic techniques *HS* and *FWA*. the *HS* algorithm has the best quality performance among all algorithms for all datasets. However, it is found that *HS* is computationally expensive as it takes more time to accomplish the optimization process. *CSA* and *FWA* seem to deliver a good compromise between quality and computational time as *FWA* is the most rapid among all methods followed by *CSA*, as shown in Fig.3.

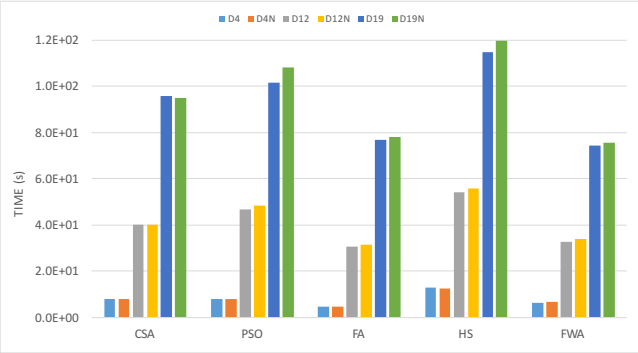
Figure 2: Average fitness of CSA, HS and FWA on each dataset.



These results are found to be very promising especially when comparing them with techniques specially developed for Big-OPT such as the Differential evolution variants. For example, *HS* can reach optimums that are near to optimums found by the variant *ACDE* in [10]. Additionally, *HS* is twice as faster as *ACDE*. Also, *HS*, *FWA* and *CSA* obtains better results than *JADE*, *SHADE*, *DECC* and *NSGAII* [16] in terms of quality and computational time.

This is exactly the main purpose of this paper, where the implementation of simple metaheuristics reported in this study can be very helpful for readers and relatively fruitful than implementing adapted or more complex methods to solve Big-Opt. Moreover, the simplicity of the successful methods in this study allows a straightforward parallel GPU implementation to fulfill the very short computational time constraint.

Figure 3: Computational time comparison in all datasets.



6 CONCLUSIONS

This paper introduced a comparative study for the Big-Opt problem. The comparison involves methods from evolutionary, swarm and memetic computation fields notably Clonal Selection Algorithm, Particle Swarm Optimization, Firefly Algorithm, Harmony Search and Fireworks Algorithm. Intensive initialization tests conclude that the exploitative configuration of the used methods is the best adapted to Big-OPT. The reported experimental results showed that memetic and evolutionary techniques are much more efficient than swarm intelligence methods which suffer from stagnation. As perspectives for this work, a good tuning of the used evolutionary and memetic techniques in a parallel implementation could be of interest in order to efficiently solve the problem in real time. Also, it is still a major work to test back the recorded performance on a real world application although the used benchmarks have been collected from real data records.

REFERENCES

[1] S. G. Mason, A. Bashashati, M. Fatourechi, K. F. Navarro, G. E. Birch. A comprehensive survey of brain interface technology designs. Annals of biomedical engineering.35 (2) (2007) 137-169.

[2] He, L., Liu, B., Hu, D., Wen, Y., Wan, M., & Long, J. (2015). Motor Imagery EEG Signals Analysis Based on Bayesian Network with Gaussian Distribution. Neurocomputing.

[3] Murthy, J. M. K. (2003). Some problems and pitfalls in developing countries. Epilepsia, 44(s1), 38-42.

[4] Big Data Infographic: Solve your Big Data Problems, http://www.intel.in/content/www/in/en/big-data/solving-big-data- problems-infographic.html.

[5] http://www.husseinabbass.net/BigOpt.html. Accessed: 25 April 2016.

[6] Goh, S. K., Tan, K. C., Al-Mamun, A., & Abbass, H. A. (2015, May). Evolutionary big optimization (BigOpt) of signals. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 3332-3339). IEEE.

[7] E. Parvinnia, M. Sabeti, M. Zolghadri Jahromi, R. Boostani, (2013). Classification of EEG Signals using adaptive weighted distance nearest neighbor algorithm. Journal of King Saud University-Computer and Information Sciences.

[8] Goh SK, Abbass HA, Tan KC, Al-Mamun A (2015) Decompositional independent component analysis using multi-objective optimization. Soft Computing, pp 1–16.

[9] Zhang, Y., Zhou, M., Jiang, Z., & Liu, J. (2015, May). A multi-agent genetic algorithm for big optimization problems. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 703-707). IEEE.

[10] An adaptive configuration of differential evolution algorithms for big data. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 695-702). IEEE. DOI: 10.1109/CEC.2015.7256958.

[11] Elsayed, S., & Sarker, R. (2016). Differential evolution framework for big data optimization. Memetic Computing, 1-17.

[12] Cutello, V., Nicosia, G., & Pavone, M. (2006, April). Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator. In Proceedings of the 2006 ACM symposium on Applied computing (pp. 950-954).

[13] Kennedy, J. (2011). Particle swarm optimization. In Encyclopedia of machine learning (pp. 760-766). Springer US.

[14] Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. In International Symposium on Stochastic Algorithms (pp. 169-178). Springer Berlin Heidelberg.

[15] Yang, X. S. (2009). Harmony search as a metaheuristic algorithm. In Music-inspired harmony search algorithm (pp. 1-14). Springer Berlin Heidelberg.

[16] El Majdouli, M. A., Rbouh, I., Bougrine, S., El Benani, B., & El Imrani, A. A. (2016). Fireworks algorithm framework for Big Data optimization. Memetic Computing, 8(4), 333-347.