Automatic Estimation of Differential Evolution Parameters using Hidden Markov Models

Marwa Keshk, Hemant Singh, Hussein Abbass

University of New South Wales, School of Engineering and Information Technology, Canberra, ACT 2600, Australia.
{m.keshk,h.singh,h.abbass}@adfa.edu.au

Abstract. Differential Evolution (DE) has been successful in solving practical optimization problem. However, similar to other optimization algorithms, the search performance of DE depends on the efficacy of the adopted search operators. The ability to adapt these operators within an evolutionary run enhances their ability to find better quality solutions. This adaptation process requires learning algorithms capable of compressing the information embedded within a population into meaningful estimates to adapt the search operators.
Hidden Markov Models (HMMs) are learning algorithms designed to estimate parameters by compressing information collected from on a state space. In this paper, we use HMMs to compress the information within a population and use the model for adapting the DE parameters. The resultant DE-HMM algorithm dynamically adjusts the two basic parameters of DE. After a thorough testing of this method and conducting an extensive comparison of its performance on the CEC2005 and CEC2014 dataset, it is shown that the proposed DE-HMM algorithm is able to achieve better results compared with the classical DE and other state-of-the-art methods. On average, the algorithm can achieve this performance faster than other methods in the literature.

Keywords Differential Evolution (DE), Hidden Markov Models, self-adaptive parameter control

1 Introduction

The Differential Evolution (DE) algorithm, first proposed by Storn and Price in 1995 [1], has been successful for solving many real-life problems, including data clustering [2, 3], robotic system control [4, 5], wireless networks sensing [6], hydropower production [7], flow shop scheduling [8], and chemical engineering [9] among others.

DE has three basic operators (mutation, crossover, and selection) and three control parameters (a scaling factor, crossover rate and population size), which guide its evolutionary search. The Scaling Factor (F) is a positive-valued control parameter that affects the amplification of difference vectors (*i.e.,* approximate

gradients). In essence, it is similar to a step length in classical optimization and generates a mutant child by moving a parent in the direction(s) of the difference vectors. The Crossover Rate (CR) controls the rate of genetic transmission from a parent to the mutant vector to generate a potential new child in the population. This child is only a potential one because the replacement strategy may reject placing it in the new population if its performance is inferior to that of its parent. The Population Size (PS) represents the number of solution vectors sampled in the first generation, a setting which is maintained throughout the run in subsequent generations.

The performance of DE is sensitive to the control parameters [10]. The settings of these parameters can, in itself, be a complex optimization problem, as discussed below:

1. Setting the parameters is problem-dependent and normally discussed in the context of the 'no free lunch' theorem for optimization problems. As their configurations represent different search biases, a parameter setting may be optimal for one optimization problem but may obtain inferior results for another [11].
2. The value of parameters during search are evolution-dependent, with different stages in the search process possibly requiring a different setting of parameters [12].
3. As the parameters interact nonlinearly in a complex manner, it is challenging to understand how F and CR interact together to improve an evolutionary search [13].

To address the above challenges, we propose a novel self-adaptive parameter control algorithm for improving the performance of DE using Hidden Markov Models (HMMs). The resultant algorithm is referred to as DE-HMM, in which the DE parameters are adjusted in each generation. The first challenge is overcome by using HMM to dynamically set the parameters for DE. The second challenge is overcome by coupling each chromosome in evolution with its own set of parameters (F and CR). The third challenge is overcome by estimating F as a function of CR.

Our use of HMMs is motivated from two perspectives. First, many studies have used finite Markov models to analyze the behavior of evolutionary computation methods. As early as 1980s, Goldberg and Segrest [14] used finite Markov chains to model the behavior of simple genetic algorithms. Mahfoud [15] built on that work and used finite Markov models to predict the expected drift time for a Boltzmann tournament selection strategy. Davis and Principe [16] proved that there is a unique stationary distribution for simple genetic algorithm when mutation is used for parameter control. In essence, the mutation operator acts as a source of perturbation to the population, while selection impacts the transitions from one population to another. Markov chains have been used extensively to analyze the behavior of evolutionary computation in recent times. An old, but very useful survey is presented in [17], and more recent studies on the topic including [18, 19].

Second, and despite the significant amount of work that went into using Markov Chains to analyze the behaviour of evolutionary computation techniques, no study has actually used Markov Chains to assist evolutionary computation to improve its performance. To the contrary, differential evolution was used to improve Markov chains by building a population of chains [20].

HMM are known to be more accurate and produce smaller models than simple Markov models. The only study that we are aware off on using HMM for evolutionary computation is the work by Rees and Koehler [21]. In their work, they showed that HMM can accurately predict the parameters for evolutionary computation. Their work ran many evolutionary computation algorithms with fixed parameters and used HMM to estimate these parameters in an offline mode. It was able to accurately estimate the parameters using HMMs. This work motivates this study by asking the question of whether HMM can in fact be used to estimate the parameters in real-time and use these estimates to adapt the parameters of DE to improve the performance of the optimization process?

HMMs require three matrix-based parameters of their own: the initial state probability, the transition probability, and the emission matrices. All three matrices are formed by direct estimations drawn from the DE population; thus, no parameters for HMMs are needed in our method.

The rest of this paper is organized as follows: Section 2 reviews the literature on DE and its different adaptive variants; Section 3 discusses the basics of the HMM; Section 4 presents the framework of the proposed DE-HMM algorithm; Section 5 provides details of the experimental study conducted; and Section 6 presents summary and conclusions of the study.

2 Background

2.1 The Basic DE Algorithm

DE was designed primarily for continuous optimization problems [22, 23]. In a minimization problem, the lower fitness value is taken to be a better solution. For example, minimize the objective/fitness function $f : \mathbb{S} \rightarrow R$, where $\mathbb{S} \subsetneq \mathbb{R}^D$, and each data point $(\mathbf{x} = (x_1.....x_D) \in \mathbb{S})$ is called a decision vector. $\mathbb{S}$ is the space of feasible solutions, whereby feasibility in unconstrained optimization is defined with boundary (also called boxing) constraints alone. Thus, the feasible space is $\mathbb{S} = \prod_{j=1}^{D} [L_j, U_j]$, where L_j and U_j are the lower and upper bounds of x_j, respectively.

For a population of PS individuals, we denote each individual by $x_{i,j,G} = \{x_{i,j}.....x_{PS,D}\}$, where i is the index for an individual in a population with $i = 1, \ldots, PS$, j is the index for a decision variable with $j = 1, \ldots D$; and G is the index for the current generation.

The DE algorithm starts by initializing the population, typically with a uniform random distribution within the search space $\mathbb{S}$, as follows:

$$x_{i,j,G=0} = L_j + rand() * (U_j - L_j) \tag{1}$$

where $0 < rand() < 1$ is a uniformly distributed random number.

Each individual in the population is evaluated using the objective function to be optimized. Then, the algorithm applies three basic operations: mutation, crossover and selection [1].

1. *Mutation*: a mutant vector, $(v_{i,G})$, corresponding to each target vector, $(x_{i,G})$, can be generated using different strategies [24]. In this paper, we use the basic DE/rand/1 strategy due to its exploration capabilities that can improve the search process;

$$v_{i,G} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) \tag{2}$$

   where $r1$, $r2$, and $r3$ are distinct integer indices randomly chosen within the range of $[1, PS]$ and are different from i, and F is the amplification factor which is a positive real number within the range of $[0, 1]$ for scaling the difference vectors.
2. *Crossover*: the target vector $(x_{i,G})$ is combined with the mutant vector $(v_{i,G})$ to produce the trial vector (i.e., offspring) using different schemes, with DE algorithms commonly using either a binomial or an exponential scheme. In this paper, we adopt the most basic of the two: the binomial scheme as in [1]:

$$u_{i,G} = \begin{cases} v_{i.G} & if(rand() \leq CR) \mid\mid (j = K) \\ x_{i,G} & otherwise \end{cases} \tag{3}$$

   where $K \in \{1, 2, ...D\}$ is a randomly chosen integer for ensuring that there is at least one dimension in the trial vector that is different from its corresponding target vector $(x_{i,G})$, and CR is a user-defined crossover probability $\in [0, 1]$.
3. *Selection*: a greedy selection competition is applied whereby the best of the target (parent) and trial (offspring) vectors with the minimum fitness value survives to the next generation. In DE, a simple cycle involving mutation, crossover and selection stages is repeated until some termination conditions are satisfied.

2.2 Control parameter settings for DE

The DE algorithm has attracted a great deal of attention given its robustness, high searching accuracy, ease of implementation and use, and fast convergence. However, the canonical DE is very sensitive to its parameters (i.e., its population size, mutation and crossover strategies, as well as their corresponding control parameters) [25]. According to Storn et al. [1], choosing the values of the DE's associated control parameters F and CR is a non-trivial task although there are different empirical recommendations for them. For example, Storn et al. [1, 24] suggested that $F \in [0.5, 1]$ and $CR \in [0.8, 1]$, Liu et al. [26] advocated $F = 0.9$ and $CR = 0.9$ with respect to [1] and, Gamperle et al. [25] proposed $F = 0.6$ and $CR \in [0.3, 0.9]$. Ronkkonen et al. [27] stated that the F should set between 0.4

and 0.95, and the CR should be sampled within $[0.9, 1]$ for separable problems and within $[0.9, 1]$ for non-separable and multimodal problems.

Setting these control parameters manually is both inefficient and time consuming, therefore many adaptive DE variants that dynamically update them during the evolutionary search have been proposed [10]. The idea of parameter adaptation was first introduced in the context of a genetic algorithm (GA) [28]. As it is difficult to classify DE parameter control methods according to a well-defined taxonomy, we will categorize them in the following three classes based on a 'how' criterion:

- **deterministic** - the control parameters are altered using deterministic rules regardless of any information obtained as feedback from the system [29];
- **adaptive** - the control parameters can be updated dynamically and incorporate some form of feedback from the search procedure to guide their adaptation and determine the direction and/or magnitude of any parameter change [30, 31]; and
- **self-adaptive** - the parameters are directly encoded into an algorithm to co-evolve as part of the optimization problem [32].

Usually, adaptive and self-adaptive methods can achieve better results than the classical DE algorithms because their parameters are automatically and favorably updated with respect to the ongoing evolutionary process [33, 12].

The PS parameter has also received its fair share of attention in the literature. Most of the time, it is set to a value predetermined by the user according to the problem's dimensionality. Different suggestions have been made regarding the setting, which is typically maintained invariant throughout the DE process. Storn et al. [1], Gamperle [25], and Ronkkonen [27] suggested PS to be fixed between $5D$ and $40D$, $3D$ and $8D$, and $2D$ and $40D$, respectively. As usual, choosing an appropriate parameter setting is not a straightforward task because of a parameter's dependencies and the complex interactions during the search [34], and adaptive/self-adaptive techniques are essential.

2.3 Adaptive selection of multiple mutation strategies and control parameters

Here, the main idea is to combine different strategies with different parameter settings into a parameters' candidate pool which could be appropriate for different problems or different stages in a particular problem [30]. In the self-adaptive DE method proposed by Qing et al. [30], called SaDE, both the trial learning strategies and control parameters (F and CR) are probabilistically self-adapted based on the promising solutions from the previous generations. It relies on a pool of four learning strategies for producing trial vector solutions, with the F measured by a normal distribution $N(0.5, 0.3)$, and the CR estimated using an independent normal distribution with a mean (CR_m) and a fixed standard deviation of 0.1 which have to be updated every 25 successive CRs.

To choose the best combination of offspring strategies and control parameter settings, Wang et al. [35] proposed a composite DE (CoDE) that reveals different

options that could improve the search procedure. It randomly imposes three trial vector generation strategies with three random control parameter (F and CR) settings to generate new solutions in each generation.

Another adaptive method for developing a parameter candidate pool introduced in [36] is called Two-Stage DE (TSDE). In it, the evolutionary process is divided into two stages using the number of fitness evaluations, with different mutation strategies and their corresponding control parameter settings implemented to achieve equilibrium between the exploration and exploitation requirements in different stages. More recently, Tang et al. [37] defined four distinct mutation strategies with some parameter values (F and CR) assigned to the population based on the ordering of their individual ranking, computed by a component depending on their fitness information being either superior or inferior.

Mallipeddi et al. [38] proposed an ensemble DE of offspring generation strategies and control parameter settings. It involves one pool of diverse strategies and another of different values for each parameter, whereby a combination of strategy and parameter settings is selected based on their successes in previous generations to produce better offspring. The empirical results indicated an outstanding performance compared with those of other peer algorithms.

Elsayed et al. [39] developed a self-adaptive method for parameter control called UMOEA in which the population is divided into three sub-populations of the same size, and then a multi-operator algorithm is applied individually to each sub-population. The success rate for a predefined number of generations is recorded to determine the best multi-operator. This method was tested on different functions and produced reasonable results with relatively less evaluation cost than other approaches.

2.4 Strategy for generating offspring

Another interesting research direction is to improve the DE's strategy for generating offspring. Zhang and Sanderson [40] implemented an extension of the mutation strategy DE/current-to-best, called 'DE/current-to-pbest', in the JADE algorithm, formulated as follows:

$$v_{i,G} = x_{i,G} + F.(x_{pbest,G} - x_{i,G}) + F.(x_{r1,G} - x_{r2,G}), \tag{4}$$

where $x_{pbest,G}$ is chosen randomly from $100p\%$ of the population vectors through the current generation, with $p \in [0, 1]$, $x_{r1,G}$ and $x_{r2,G}$ adopted randomly from the combination of the current population and optional archive of previously generated offspring. Moreover, the F and CR are independently updated for each target vector using the Cauchy and normal distributions, respectively.
Qiu et al. [41] developed an adaptive cross-generation DE for multi-objective optimization. This study suggested two mutation strategies for efficiently preserving the population's diversity, and a new cross-generation adaptive technique (CGA) for automatically adjusting the F and CR via information regarding the objective space and evolutionary process information. Wang et al. [42] used the

cumulative distribution information of the population, which helps to implement an Eigen coordinate system for the crossover operator. This operator is executed in both the original and Eigen coordinate systems with better offspring surviving for the next generations.

2.5 Configuring control parameter settings

Liu and Lampinen [26] proposed a new DE, called the fuzzy adaptive DE (FADE). Its parameters are adapted while running the evolutionary process using fuzzy logic controllers, the inputs for which are provided with their relative function values and successive generations' individuals. These parameters respond to the population information, i.e., parameter vectors, function values and their changes during the search process, with its experimental results demonstrating that it outperforms the conventional DE on high dimensional problems. Yu et al. [43] designed an individual dependent parameter adaptation in which the parameters are updated at the two levels, that of population and individual. The former depends on the optimization state (explorative or exploitative) and the latter relies on the population's characteristics.

Some optimization algorithms rely on probabilistic models, such as Covariance Matrix Adaptation (CMA) [44] which aims to maximize the progress of reproduction in a search's evolutionary path. Wang et al. [45] introduced a new mechanism called CoBiDE for dynamically adjusting the control parameters F and CR based on a covariance matrix learning and binomial distribution parameter settings with the purpose of achieving a balance between DE's exploration and exploitation. The covariance matrix of CoBiDE uses a coordinate system for the crossover operator, and the binomial distribution with two Cauchy distributions. On the one hand, F is randomly generated by a Cauchy distribution, with locations of 0.65 and 1.0 and scales of 0.1 and 0.1 for the first and second halves of the time period. On the other hand, CR is randomly generated by a Cauchy distribution, with locations of 1.0 and 0.95 and scales of 0.1 and 0.1 for the first and second halves of the time period.

The experimental results showed that CoBiDE performed effectively compared with other DE variants and state-of-the-art algorithms. A covariance matrix with the ES method called CMA-ES was developed by Hansen and Ostermeier [46]. It aims to adapt the covariance matrix of the sampling population based on previously accepted samples with the purpose of maximizing the likelihood of the search process by generating successful mutations (i.e., step sizes and search directions).

A DE with self-adaptive parameter control based on probabilities called jDE was designed by Brest et al. [13]. It encodes the control parameters into a population of individuals by initializing F to 0.5 and CR to 0.9 for each individual. Then, the F and CR are regenerated based on uniform distributions of $[0.1, 1]$ and $[0, 1]$, respectively. With regard to the probabilities $\tau_1 = \tau_2 = 0.1$, new F and CR are calculated in subsequent generations by using the following equations.

$$F_{G+1} = \begin{cases} F_l + rand_1.F_u & if rand_2 \leq \tau_1 \\ F_{i,G} & otherwise \end{cases}$$

$$CR_{G+1} = \begin{cases} rand_3 & if rand_4 \leq \tau_2 \\ CR_{i,G} & otherwise \end{cases}$$

Here, $rand_1$, $rand_2$, $rand_3$, and $rand_4$ are random numbers which follow a uniform distribution of [0, 1], F_l andF_u equal 0.1 and 0.9, respectively.

Abbass [47] proposed a self-adaptive Pareto DE algorithm for multi-objective optimization problems in which the F and CR are regenerated using a Gaussian distribution. Competitive results were obtained in the study compared with other contemporary multi-objective optimization algorithms. In the last few years, solving problems with multiple optimal solutions has become a big challenge; for instance, Basak et al. [48] proposed a DE technique called MOBIDE, which uses a bi-objective formulation for multimodal optimization. It showed outstanding performance compared to a number of other single and bi-objective niching algorithms. An adaptive mechanism implemented in [49] for the F and CR based on exponential weighting moving average performs better than some of its variants. Sarker et al. [50] developed a dynamic selection method for defining and choosing the most highly ranked combination of control parameter settings to be used in the subsequent generations. Recently, Corriveau et al. [51] developed a genetic adaptive method using a probabilistic model of the Bayesian network (BN) in which the BN uses a graphical model to represent the relationships among the GA parameters. Tanabe and Fukunaga [52] proposed an extension for self-adaptive mechanism (L-SHADE), in which the PS linearly decreases throughout the search process while the SHADE algorithm automatically adapts the F and CR based on their success histories. The experimental results showed that L-SHADE outperforms SHADE and other algorithms.

3 Hidden Markov Model

A Hidden Markov Model (HMM) is a powerful statistical data analysis method used to model a wide range of sequential data epitomized in a sequence of observations based on probabilistic measures [53]. Its basic theory was proposed in the 1960s by Baum and Petrie [54] based on Bayes theorem:

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}, \tag{5}$$

where A and B are events, $P(A)$ is the prior probability, $P(B)$ is the marginal probability for normalization purposes (i.e., evidence), $P(B \mid A)$ denotes the conditional probability of B given that A is true and $P(A \mid B)$ is the posterior of event A. An HMM is defined in [55] as a double stochastic process which is a discrete-time finite-state Markov chain that is not observable (i.e., in a hidden state). However, a sequence of observations can be formed with another set of stochastic processes. An HMM can be characterized by the parameters:

- T: the state transition probability matrix (Hidden states),
- N: the number of hidden states,
- E: the observed probability matrix (Emission matrix),
- M: the number of observations, and
- π: the initial state probabilities (at t=1).

In order to formulate an HMM, we use the notations by Rabiner and Juang [55]. We denote the set of latent states by $S = (s_1, s_2, ..., s_N)$, where each state generates one of the observation set $V = (v_1, v_2, ..., v_M)$. Since a system can change from one state to another state, by obtaining a set of states $Q = (q_1, q_2, ..., q_T)$ and a corresponding sequence of observations $O = (O_1, O_2, ..., O_T)$, such that $O_t \in V$, the transition probabilities from states i to j with a Markovian property representing a $N \times N$ transition probability matrix (T_{ij}) which satisfies:

$$T_{ij} = P(q_t = s_j \mid q_{t-1} = s_i) \tag{6}$$

Given a set of states and observations, we define a $N \times M$ observation (i.e., emission matrix E) with the probability of observation (V) being generated from state i, $E = (E_i(v_m))$, as:

$$E_i(v_m) = P(q_t = v_m \mid q_{t-1} = s_i) \tag{7}$$

A typical HMM contains three parameters, represented as $M = (T, E, \pi)$ which satisfy:

- $\sum_{j=1}^{N} T_{ij} = 1$ where $1 \leq i \leq N$
- $\sum_{j=1}^{M} E_{ij} = 1$ where $1 \leq i \leq N$
- $\sum_{i=1}^{N} \pi_i = 1$ where $\pi \geq 0$

4 Proposed DE-HMM algorithm

As previously discussed, different strategies and control parameter values for DE (i.e., the F and CR) need to be carefully adjusted for different optimization problems and, in the literature, are independently adapted; for example, the JADE algorithm [40] individually generates the F and CR and adapts them using Cauchy and Gaussian distributions, respectively. Mallipeddi et al. [38] stated that the performance of DE depends on the combination of F and CR but, although their interdependency has often been investigated, their actual dependencies have not been effectively demonstrated.

In this study, we propose a novel DE-HMM algorithm which aims to automatically adjust the CR and F of DE by coupling a HMM with the DE procedure. The speed of DE is not greatly impacted, particularly because all

the statistics required for the HMM are simply computed from the first-order mathematical measures, such as probability, mean, and standard deviation, at a linear computational cost.

A HMM can be precisely modeled by defining its states (T) and observations (E) from particular data streams. In multivariate time-series models (MTSM) [56], $x_0, x_1, ..., x_t$, where t is time, the values of these variables are observed over time. However, although we cannot obtain any information about the observed sequences of these values over time, namely the hidden states, they can be estimated using HMM, as schematically shown in Figure 1.

These states could be discrete or continuous depending on their data norms [57]. In order to identify the internal changes in the DE population over time, as a type of MTSM, we employ a HMM with two predefined discrete states measured from the change in DE's fitness probabilities.

We adapt the F and CR of the DE-HMM algorithm by computing the posterior and likelihood ratios of the HMM, respectively. A DE population changes naturally over time which is similar to the internal procedure of HMM (i.e., time series models [56]). We consider each change to be in one of two states: either negative when the fitness function does not increase and is denoted by 'low'; or positive when it increases and is denoted by 'high'. These states can be estimated using the HMM procedure and measured by computing their posterior probabilities considering values from 0 to 0.5 as 'low' and from 0.5 to 1 as 'high' which represent the HMM transition matrix. The advantage for using categories is to avoid over-fitting an HMM to a particular population since our objective is not reverse engineer the parameters as what Rees and Koehler [21] did, but to estimate the values that should be adopted. The HMM emission matrix is estimated by calculating the probability that a population is drawn from the DE stochastic process. A complete example demonstrating the working of DE-HMM is presented at the end of the supplementary materials.

In Figure 1, the HMM is represented as a directed weighted graph called a transition diagram in which the nodes correspond to the states of DE evolution changes and the edges of all possible transitions between the states. For each state, there is an emission probability matrix that explicitly indicates its observations as the HMM deals with both observed and hidden events.

The proposed DE-HMM estimates F and CR parameters, with each individual ($x_{i,G}$) in the population having its own $F_{i,G}$ and $CR_{i,G}$ which are used to produce the trial vectors shown in Figure 2. It executes a self-adaptation for these parameters by considering the inter-correlation between them as the chosen parameter values should generate the individuals most likely to survive. This interaction is modelled by first estimating CR then using the estimated CR in conjunction with the minimum posterior probability for the given population observation emission matrix ($EMseq$).

The full procedure for computing the DE control parameters (F and CR) is depicted in Figure 3. The relationship between DE and HMM is shown and the estimation of both F and CR depend on different HMM probabilities and the posterior and Likelihood ratio for F and CR, respectively. The DE population

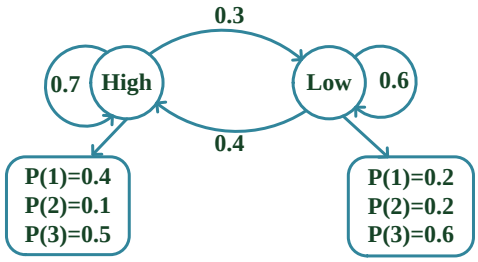

Fig. 1. Example of HMM transition diagram for DE. the low and high states represent the low and high change rates of DE, respectively, and the observation set ($O = \{1, 2, 3\}$) reflects the three dimensions of a population over time t.

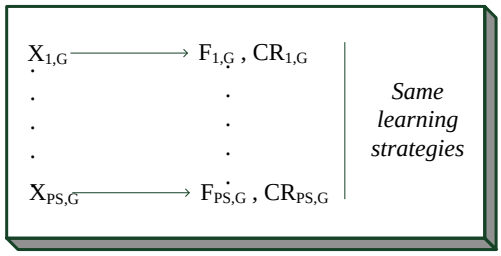

Fig. 2. Encoding of control parameters with individuals

is transformed into probabilities where the initial population is taken as prior probability; the parent vectors, denoting the population observations are taken as likelihood probability; and the mutant vectors, reflecting the change from the parent vectors are taken as the case of the posterior probability. HMM uses these probabilities to estimate CR and F.

The parent and mutant vectors of DE are transformed into probabilities using Algorithm 1, where it loads a population of size R rows and C columns and calculates the probability distribution for individuals in the population, pop_m.

The DE population values are used as input for computing the parameters of the HMM (π, T, E) and the HMM model is built concurrently as DE is running. Firstly, the initial states (π) are set at {0.5, 0.5} for {low, high} to begin building the model with equal probabilities. Secondly, the states of the HMM denote the population change rates over time (i.e., high or low) of the potential evaluation process which can be measured internally through changes in the observation probabilities [58, 59]. To compute CR, the likelihood ratio is calculated between the actual sequence of states ST_{seq} and the most likely state sequence $best_{seq}$ estimated by the Viterbi algorithm [60] (discussed shortly). The emission matrix (E) is measured from the population sample values to reflect the interde-

Input : Population pop_m as a matrix of size $R \times C$
Create a new matrix pop_r of size $R \times C$;
Create two vectors μ and σ of length C;
for *each column* j *in* pop_m **do**
  $\mu(j) \leftarrow mean(pop_m)$ over all rows for column j;
  $\sigma(j) \leftarrow stdev(pop_m)$ over all rows for column j;
end
for *each row* i *in* pop_m **do**
  for *each column* j *in* pop_m **do**
    $pop_r(i,j) \leftarrow prob(pop_m(i,j) \leq \zeta),\ \zeta \in N(\mu(j), \sigma(j))$
  end
end
Output: pop_r

Algorithm 1: Transformation of population values to probabilities.

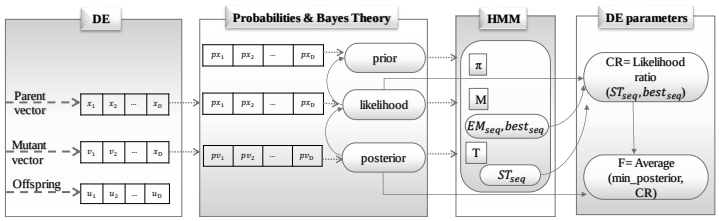

Fig. 3. Complete procedure for computing DE control parameters.

pendencies in the evolutionary process and reflects the likelihood probabilities. Similarly, the observation sequence EM_{seq} is estimated using the Viterbi algorithm to calculate the posteriors, then F is estimated by taking the average of the minimum posteriors and CR. The steps for estimating these parameters are provided in Algorithm 2 and details regarding building the HMM from DE are discussed in the following subsections.

Input : Population pop_m as a matrix of size $R \times C$
Convert pop_m to probability emission matrix B using Algorithm 1
Initialize A with low, high states
Sample random sequence of states ST_{seq} and emission symbols EM_{seq}
Apply Viterbi algorithm to find the best state sequence $best_{seq}$ that maximizes the likelihood of EM_{seq}
$CR \leftarrow Likelihoodratio\ (ST_{seq} \geq best_{seq})$
$F \leftarrow Average\ (min_{posterior}|EM_{seq}, CR)$
Output: F, CR

Algorithm 2: Steps for building the HMM from DE information.

4.1 Computing crossover rate (CR)

Setting the Crossover Rate CR is usually more sensitive to problem characteristics such as modality (unimodal/multimodal) [12]. CR decides the mixing probability of the current target vector and the donor vector resultant from the mutation operator to construct a trial vector (offspring).

In this study, we apply the binomial crossover operator as shown in Equation 3. The Viterbi algorithm [60] is employed to estimate CR by selecting the best state sequence that maximizes the probability of CR given an observation sequence. The algorithm uses dynamic programming and acts as a decoder in which the HMM parameters (T, E) and a sequence of observations $O = (O_1, O_2, ..., O_T)$ are given. It estimates the most probable state sequence from the maximum of all previous sequences $max_{q_0, q_1 .... q_{t-1}}$. Therefore, for a current state (q_j) at time (t), the Viterbi probability $v_t(j)$ is calculated as:

$$v_t(j) = \max_{t=1}^{N} v_{t-1}(i) * T_{ij} * E_j(O_t), \tag{8}$$

where, $v_{t-1}(i)$ denotes the preceding probability of the Viterbi path, T_{ij} is the probability of transition from states q_i to q_j, and $E_j(O_t)$ is the emitted observation probability given the current state j. CR is estimated by calculating the likelihood ratio of the actual sequence of states (ST_{seq}) generated being better than the best state sequence $(best_{seq})$ using the Viterbi algorithm, as:

$$CR = Likelihoodratio\ (ST_{seq} \geq best_{seq}) \tag{9}$$

We rely on the likelihood ratio (i.e., of being in a state $ST_{seq} \geq best_{seq}$) as a way of comparing a parent vector to its mutant vector for generating new offspring. This ratio becomes the likelihood of ST_{seq} (i.e., the actual state sequence) given $best_{seq}$ (i.e., the most likely state sequence), as shown in Figure 3.

Define fitness function f; population size PS; population dimensions D; the maximum number of function evaluation Max_{Eval};
Set $G = 0$; $Feval = PS$; $F = CR = 0.5$;
$pop_0 \leftarrow rand()$ using Eq 1;
Evaluate pop_0 with f;
while *(*$Feval \leq Max_{Eval}$*)* **do**
  for *each* $x_{i,G} \in pop$ **do**
    $v_{i,G} \leftarrow$ Mutation (Eq 2);
    $u_{i,G} \leftarrow$ Crossover (Eq 3);
    Evaluate $u_{i,G}$;
    Select($u_{i,G}$,$x_{i,G}$) (Eq 5);
  end
  Sort(pop_G,ascending on fitness);
  Update F,CR for the sorted, successful vectors using HMM in Algorithm 2;
end

Algorithm 3: An iteration of the DE-HMM algorithm.

4.2 Computing mutation factor (F)

The mutation factor F is automatically adapted using the HMM posteriors. It is an important factor for balancing exploration and exploitation and acts as a perturbation ratio for an offspring attempting to reach the desired region of the search space [61]. A dynamic F is computed from the minimum HMM posterior of a particular state, given the emission sequence, to indicate a change of state, either low or high, in the population sequence. The posterior distribution is used to generate a Bayesian inference towards the next search region to be explored which can be calculated as the conditional probability of ending up in such state given the observed population sequence, as shown in Figure 3.

In DE, one of the functionalities of a new mutant vector is to help exploring different regions in the search space [61]. The movement probability between these vectors (parent and mutant) can be estimated as the lowest probability that occurs in the mutant vector to explore the most possible sub-regions given all previous information in the original vectors, both the emission observations (EM_{seq}), and the changes in their CR values, as:

$$F = Average\ (min_{posterior}|EM_{seq}, CR) \tag{10}$$

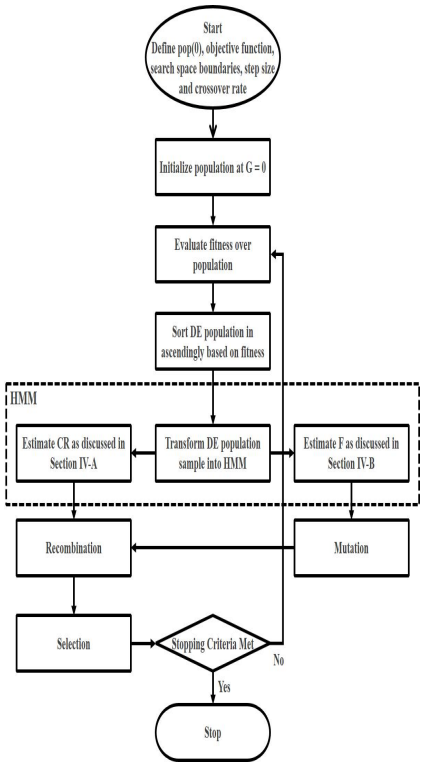

Fig. 4. General flowchart of the proposed DE-HMM algorithm.

4.3 Architecture of DE-HMM

The architecture of DE-HMM is presented in Algorithm 3, which demonstrates our way of implementing the HMM procedures in the DE evolution process for adjusting the intrinsic F and CR parameters, as introduced above. The main difference between the classical DE and DE-HMM is the phase for constructing the HMM from the DE's sorted population, as shown in Figure 4.

Firstly, a population of PS individuals is randomly initialized using Equation 1, with initially uses $F = 0.5$ and $CR = 0.5$. Subsequently, all the individuals are evaluated according to $f(x)$. Secondly, the mutation operation is employed for each generation via Equation 2 to generate the mutant individuals, followed by the crossover operation as we apply the basic binomial crossover scheme in Equation 3 to produce the trial individuals (offspring).

The procedure for updating the values of F and CR depends on how the DE population changes over time from generation to another. In DE-HMM, the DE population is sorted in an ascending order and gets transformed into probabilities as shown in Algorithm 1 in order to fit the HMM parameters. We then sample random sequences of ST_{seq} states and EM_{seq} from the HMM.

The CR value is estimated by calculating the likelihood ratio between the actual state sequence ST_{seq} that is better than the best state sequence $best_{seq}$, generated using the Viterbi algorithm, as in Equation 9. This is derived from DE where CR in crossover operation determines the improvement ratio of the new offspring from the original and mutant vectors. Also, the F value is estimated based on averaging the minimum posterior (i.e., reflects the change of DE state) and the estimated CR to recognize the dependency between F and CR, as in Equation 10. Finally, each original individual is compared with its corresponding offspring and the individual with less fitness value survives to the next generation.

5 Experimental Study

5.1 Benchmark functions and experimental setup

We use two benchmark test suites from IEEE CEC2005 [62] and IEEE CEC2014 [63] special sessions on real-parameter optimization to evaluate our proposed algorithm's performance. A total of 55 test functions with 10, 30, and 50 dimensions (D), which are summarized in the supplementary attachment (Tables A1 and A2). The 25 test functions from CEC2005 are denoted as C1 ~ C25 and those from CEC2014 are denoted as F1 ~ F30. The explanation of these benchmark problems can be found in [62] and [63]. We compare DE-HMM against the classical DE (i.e., fixed values of the F and CR), different variations of DE-HMM, and twelve competing algorithms of which eight are competitive DE-variants and four are non-DE EAs, described in Section 4.3.

DE-HMM was implemented in Matlab R2015a, and executed on a PC with a 3.40 GHz TM-Core i7 processor, 16 GB RAM running Windows 7 64 bits. The PS for DE-HMM is set to 60. The maximum number of function evaluations

Table 1. F and CR values used for DE-HMM variants.

Variant	F posterior	CR likelihood ratio
DE-HMM-minG	min	current $>$ best
DE-HMM-minL	min	current $<$ best
DE-HMM-minE	min	current $=$ best
DE-HMM-meanG	mean	current $>$ best
DE-HMM-meanL	mean	current $<$ best
DE-HMM-meanE	mean	current $=$ best
DE-HMM-maxG	max	current $>$ best
DE-HMM-maxL	max	current $<$ best
DE-HMM-maxE	max	current $=$ best

(FEs) are set to $10000 \times D$, and thirty independent runs are conducted for each problem. To measure deviations between the estimated and the optimal results, we use the mean and standard deviation (mean $\pm$ std) of the fitness error values $(f(x) - f(x^*))$, with those smaller than 1.0E-08 replaced by zero. Because of space limitation, the detailed results are provided in the online supplementary file. In the comparison tables, we highlight the best results in boldface type and the number of best results (No-Best) obtained by the corresponding algorithm for all test problems are reported at the end of the comparison tables.

We applied two non-parametric statistical hypothesis tests, namely, the Friedman and Wilcoxon tests [64, 65], executed using the SPSS statistical tool [66]. The former compares the algorithms' mean ranks while the latter, assess the significance of performance of DE-HMM and the other algorithms. Particularly, the null hypothesis for the tests assumed that there is no significant difference between the mean error values of two samples while the alternative hypothesis tries to determine if there is a significant difference between these samples, using a 5% significance level. The Wilcoxon statistical results are marked with "+" or "-" or "=" (in the supplementary statistical tables "significance" column) to show the significant difference in the average fitness values between DE-HMM and the peer algorithms, where "+" or "-" mean that DE-HMM is significantly better or worse than the competing algorithm, respectively while "=" means that no significant difference between the two algorithms.

5.2 Analysis of performances of DE-HMM and its own variants

In this experiment, we analyze the roles of different components in DE-HMM using: (i) the likelihood ratio for setting CR and (ii) the HMM posterior for setting the F. We use the 30D CEC2005 benchmark problems with eight different parameter choices. The best choice of DE-HMM is denoted in this section by DE-HMM-minG to differentiate it from other variants and to indicate it uses minimum posterior and maximum fitness. We compare DE-HMM-minG against a DE that randomly sets the parameters in each generation (i.e., Rand-gen). The comparison involves all variants and Rand-gen to allow for an appropriate test of significance to be used [67].

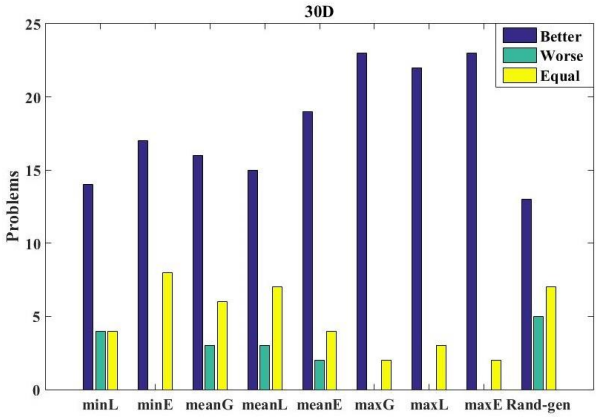

Fig. 5. Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and its own variants.

The eight DE-HMM variants are shown in Table 1 in which the second column refers to the posterior trend of the next region (i.e., minimum, maximum, or mean step size) and the third column indicates the number of individuals that can be improved ($>$ or $<$ or no improvement). We also compare DE-HMM against a DE with random control parameters that we call (Rand-gen). A summary of the performances of all the DE-HMM variants and Rand-gen DE for the 30D CEC2005 test problems based on the Wilcoxon's and Friedman tests are shown in Figure 5, and Table 2, respectively. The complete results (i.e., mean function error) and Wilcoxon results are presented in the supplementary document in Tables G1 and G2, respectively. In Figure 5, the proposed algorithm obtains better average fitness values (i.e., fewer errors) than the other algorithms as "Better" means that DE-HMM outperforms each of the compared variants while "Worse" indicates that the competitor variant has better results relative to DE-HMM. Also, Table 2 shows that the proposed DE-HMM can achieve the highest ranking.

When evaluating the range of values that F and CR take in different problems (Figures I1 and I2 in the supplementary file), we discover that these ranges are different for different problems. For C2 in Figure I1, the values of F with the minimum posterior is in the range of [0.3, 0.5], while the mean, and maximum posterior are within [0.6, 0.75] and [0.85, 1], respectively. Similarly, the CR for higher proportions of improvement has values within [0.7, 1] and those for a lower ratio or no further improvement are within [0.6, 0.9] or [0.5, 1]. DE-HMM adopts slightly different ranges for C12.

In summary, the DE-HMM variant we adopt in this paper (DE-HMM-minG) has better results than other variants and than random control of parameters.

Table 2. Statistical analysis of different variants of DE-HMM.

Method	30D
	Ranks
DE-HMM-minG "DE-HMM"	**2.42**
Rand-gen	3.2
DE-HMM-minL	3.62
DE-HMM-minE	4.18
DE-HMM-meanG	4.52
DE-HMM-meanL	5.16
DE-HMM-maxG	7.4
DE-HMM-meanE	7.94
DE-HMM-maxL	8.28
DE-HMM-maxE	8.28

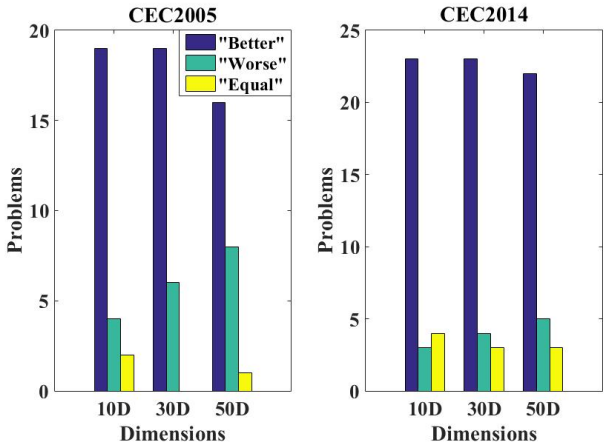

Fig. 6. Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and C-DE.

5.3 Performance analysis of DE-HMM

Table 3. Rankings obtained from Friedman's test of 10, 30, and 50 dimensional CEC2005 functions of DE-HMM, C-DE and its DE variants.

Method	Ranks		
	10D	30D	50D
DE-HMM	**2.56**	**2.72**	**2.80**
CoBiDE	2.60	2.74	2.90
JADE	3.32	3.12	3.16
jDE	3.52	4.04	3.52
SaDE	4.10	3.88	4.56
C-DE	4.90	4.50	4.06

We compare DE-HMM-minG against the classical DE, we use the DE/rand, called C-DE with its parameters set to $PS = 100$, $F = 0.5$ and $CR = 0.9$. A summary of the results obtained from the Wilcoxon statistical test of the two algorithms is shown in Figure 6 in which it is clear that DE-HMM-minG achieves better performances for the majority of the problems. Details of these results are provided in Tables B1 and B2 in the supplementary file.

We extend the comparison to vary the crossover operator and mutation strategy in DE-HMM. Four distict setups are shown in .

The four DE-HMM variants shown in Table 4 compare four variants of DE-HMM. The first column shows thebinomial crossover against exponential crossover shown in Equations 11 and 12, respectively. In Equation 11, $x_{best,G}$ is the best fitness vector chosen at generation G and F is the scaling factor, generated by DE-HMM method. l and$\prec l \succ_D$ in Equation 12 are the starting position, and the modulo function with modulus D, respectively. The second column shows the mutation strategies 'rand' and 'best'. The third column refers to the crossover scheme binomial and exponential, used for generating offspring solutions.

Table 4. Description of the four mutation and crossover setups used in DE-HMM analysis.

	Mutation strategy	Crossover scheme
DE-HMM "rand/bin"	rand	binomial
DE-HMM "rand/exp"	rand	exponential
DE-HMM "best/bin"	best	binomial
DE-HMM "best/exp"	best	exponential

$$v_{i,G} = x_{best,G} + F * (x_{r1,G} - x_{r1,G}) \tag{11}$$

$$u_{i,G} = \begin{cases} v_{i.G} & \forall j = \prec l \succ_D, \prec l+1 \succ_D, ..., \prec l+L-1 \succ_D \\ x_{i,G} & otherwise \end{cases} \quad (12)$$

We use the CEC2005 and CEC2014 benchmark functions at 30 dimensions to evaluate their performances in terms of the obtained solution quality, where the detailed results are reported in Table I1 and I2 in the supplementary file, respectively. The Wilcoxon statistical results for the four setup are shown in Tables 5 and 6 for both CEC2005 and CEC2014 test set, respectively. In Table 5, DE-HMM employing best strategy with binomial crossover achieves better results in three unimodal and one multimodal functions while using best and exponential operators get optimal results of two unimodal functions. Nevertheless, DE-HMM using rand strategy and binomial crossover is better for the majority of time. Moreover, for CEC2014 results in Table 6, DE-HMM with rand and binomial crossover operators is superior to the other variants.

Table 5. Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2005 test functions.

DE-HMM "rand/bin" vs.		"Better"	"Worse"	"Equal"	"p-value"	"Significance"
DE-HMM/rand/exp	30D	18	2	5	**0.005**	+
DE-HMM/best/bin		12	3	10	0.11	=
DE-HMM/best/exp		14	3	8	**0.0125**	+

Table 6. Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2014 test functions.

DE-HMM "rand/bin" vs.		"Better"	"Worse"	"Equal"	"p-value"	"Significance"
DE-HMM/rand/exp	30D	21	5	4	**0.001**	+
DE-HMM/best/bin		15	9	6	0.155	=
DE-HMM/best/exp		16	10	4	0.146	=

Comparison with the state-of-the-art variants Veček et.al [68] stated that comparisons in the literature should be made either against parameter settings that has been tuned using a parameter tuning algorithm, or against algorithms with abilities for parameter control. Given that our proposed method falls in the second category, we have compared against similar algorithms.

DE-HMM is compared against some state of the art parameter control algorithms. DE-HMM is compared with four DE variants (i.e., SaDE [30], JADE [40], jDE [13] and CoBiDE [45]) and four non-DE EAs (i.e. BNGA [51]), CLPSO [69], CMA-ES [46], and IPOP-CMA-ES [70]), methods which are frequently used for comparison in the literature. IPOP-CMA-ES was the winner of the CEC2005 competition.

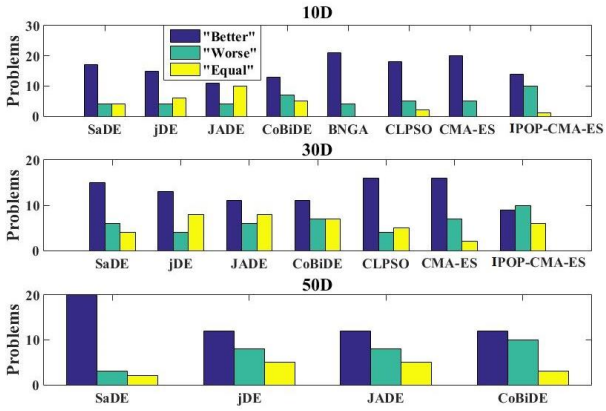

Fig. 7. Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and all the state-of-art variants.

DE-HMM, JADE, and CoBiDE perform better than SaDE and jDE and, in particular, DE-HMM offers significant improvements over the other methods for many test instances. DE-HMM shows promising results with different problem sizes, as shown in Table 3 which ranks the compared algorithms according to the Friedman test. For the 10D problems, DE-HMM is able to find the optimal solutions of all unimodal functions except for C3. The majority of multimodal and hybrid test functions can be solved efficiently by DE-HMM.

Similarly, for the 30D problems, DE-HMM exhibits superior results for most functions, especially the ones that have multimodality and complex hybrid composition. However, JADE can solve the unimodal functions better than all other algorithms and obtain the optimal results for C9, like jDE and CoBiDE. For the 50D problems, DE-HMM is still superior compared to the other algorithms, with similar rank to CoBiDE. Overall, the DE-HMM provides better or competitive performance to the other algorithms. The complete results of these experiments are provided in the supplementary document (Table C (I, II, and III)).

DE-HMM also compares well with the non-DE based algorithms, as detailed in the supplementary document (Table D (I and II)) and achieves better results for most test functions studied. CMA-ES obtains the best results on the first three instances, C5 and C7 with 10D. Comparing the general trends of these algorithms for the multimodal functions, IPOP-CMA-ES achieves good results for 30D problems. In the complex composition group of functions, the DE-HMM clearly shows better performance compared to the other algorithms.

The statistical results obtained from the Friedman test show the superiority of DE-HMM over the non-DE algorithms for both the 10D and 30D problems, as

Table 7. Rankings obtained from Friedman's test of 10, 30 dimensional CEC2005 functions of DE-HMM and its non-DE variants.

Method	Ranks	
	10D	30D
DE-HMM	**2.12**	**1.96**
IPOP-CMA-ES	2.18	2.16
CLPSO	3.00	2.90
BNGA	3.82	-
CMAES	3.88	2.98

presented in Table 7 in which DE-HMM has the highest ranking. Moreover, the statistical results obtained from the Wilcoxon test demonstrates that DE-HMM clearly obtains greater number of better results than the other algorithms, as depicted in Figure 7. More specifically, Table F1 in the supplementary file shows test of significance results and demonstrate that DE-HMM is significantly better than all competing algorithms (except JADE and CoBiDE), based on the average results obtained for all test problems for 10D and 30D problems. For JADE and CoBiDE, DE-HMM offers significantly better solutions for 10D test problems and equivalent performance for 30D and 50D.

To further demonstrate the performance of DE-HMM, convergence behavior of various algorithms is presented in Figure 8, where the x-axis indicates evolutions of generations (within the same function evaluations) and the y-axis is the median function error values.

Evaluation of DE-HMM against recent DE variants In this subsection, four most recent DE-variants (i.e., CPI-DE [42], TSDE [36], LSHADE [52], and UMOEA [39]) are compared with DE-HMM. The motivation for choosing these four algorithms for comparison is that all were either published on the proceedings of CEC2014 and/or were recently proposed DE variants that reflect the latest progress of DE. This experiment is conducted over 30 instances with 10, 30, and 50 decision variables for 30 independent runs to provide a more comprehensive comparison. All the computational results for the 10D, 30D, and 50D problems are documented in the supplementary file in Table E (I, II, III), respectively.

It is evident from Table E (I) that, for the 10D problems, performance is uniform across all algorithms for unimodal functions. DE-HMM reaches the optimal solution for two multimodal functions (F6 and F8) similar to LSHADE and UMOEA. DE-HMM is the best for six functions while LSHADE obtains the best for four functions and UMOEA is better than the others for three. Table E (II) summarizes the experimental comparison of 30D problems for which both UMOEA and LSHADE, and both TSDE and CPI-DE achieve the optimal solutions for three, and one unimodal functions, respectively, while DE-HMM's results are close to the optimal solution for F3. DE-HMM is able to obtain the

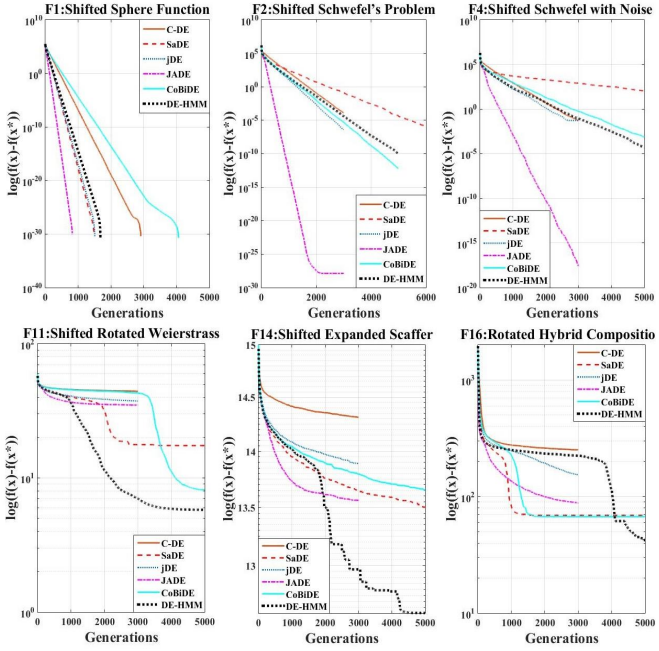

Fig. 8. Convergence plots of DE-HMM on F1, F2, F11, F14, and F16 with 30 dimensions where y-axis in log scale.

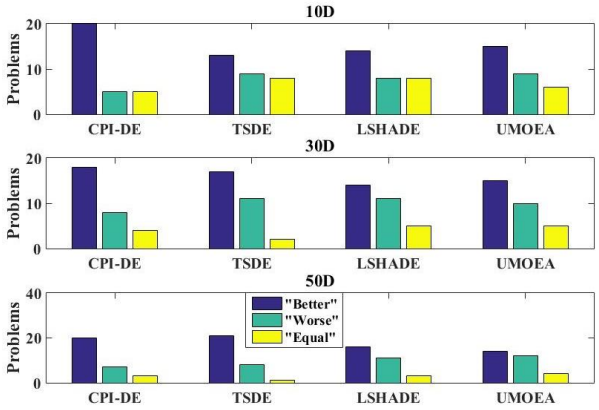

Fig. 9. Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and the up-to-date DE variants.

Table 8. Rankings obtained from Friedman's test of 10, 30 dimensional CEC2014 functions of DE-HMM and its up-to-date variants.

Method	Ranks		
	10D	30D	50D
DE-HMM	**2.67**	**2.78**	**2.67**
LSHADE	2.82	2.90	2.72
TSDE	2.83	3.68	4.22
UMOEA	3.43	3.08	2.78
C-DE	4.55	4.10	4.38
CPI-DE	4.70	4.45	4.23

optimal solution for the multimodal F7 and better results for three functions as well as competitive results for the other hybrid and composition functions. The results for 50D problems maintains the trend, where DE-HMM performs best over most multimodal, hybrid, and composition functions.

Finally, the results obtained from the Wilcoxon and Friedman tests are shown in Figure 9 and Table 8, respectively, for all dimensions in our experiment. To conclude this comprehensive comparison, it is worth mentioning that the proposed DE-HMM yields significantly better results than CPI-DE for 10D and 50D, while it is significantly better than TSDE, LSHADE, and UMOEA for 30D, 50D, and 10D, respectively as detailed in Table F2 in the supplementary file.

5.4 Time Complexity

This section describes the time complexity for comparing DE-HMM against other algorithms (either compared based on CEC2005 or CEC2014 benchmark datasets). The computational time is measured as defined in [62, 63] for CEC2005 and CEC2014, respectively. The comparisons are reported in Tables H2 and H3 for CEC2005 and CEC2014 benchmark datasets, accordingly. We employ the two standard functions used in the literature for this comparison.

Two plots are drawn to depict the performance of each algorithm in terms of the quality of the obtained solution and the time consumed for achieving it. Firstly, Figure 10 shows the trade off between the effectiveness and efficiency of DE-HMM and C-DE and DE variants, use the CEC2005 test set in the previous comparison. Function 3 is employed as defined in [62] at different dimensions (10, 30, and 50 dimensions). From this figure, it is clear that DE-HMM could consume more time than C-DE, JADE, and jDE but it achieves less error (better result) than them at 30D and 50D. In regard to the 10D results, although C-DE, jDE, and JADE are better in terms of time and quality for 10D, the difference when compared to the performance of DE-HMM is negligible.

Secondly, we test the time consumed for F18 in CEC2014 dataset as defined in [63] to evaluate the performance of the proposed DE-HMM algorithm and up-to-date DE variants. The comparisons are described in Figure 11, where DE-HMM performs better in terms of quality of solutions obtained, albeit with

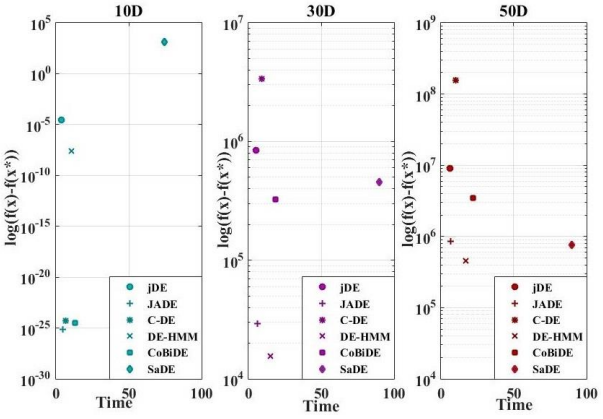

Fig. 10. Time complexity for C3 of CEC 2005 test problems at 10, 30, and 50 dimensions.

a slight increase in computational cost when compared to classical DE and LSHADE algorithms.

6 Conclusion

Dynamically controlling the parameters for Differential Evolution (DE) removes the burden from the user and reduces the time-consuming effort to find suitable parameter values manually. In this paper, we proposed Hidden Markov Models (HMMs) to automatically configure the two DE parameters, F and CR, without the need for any external information. We called the algorithm, DE-HMM. In our approach, the HMM posterior and likelihood ratios were estimated to assign the F and CR values during the evolutionary process. Two benchmark problem sets containing 55 test functions (i.e., IEEE CEC2005 and CEC2014) with 10, 30 and 50 dimensions were used to conduct comprehensive experiments to analyze the performance of DE-HMM. It was compared with classical DE, different combinations of mutation strategies and crossover operators, different variants of DE-HMM, and DE-based and non-DE state-of-the-art algorithms. The experimental results showed that DE-HMM achieve a competitive overall. In future work, this proposed self-adaptive method will be used to solve real-world optimization problems and be extended to handle constrained optimization problems.

References

1. Storn, R., Price, K.: Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Volume 3. ICSI Berkeley (1995)

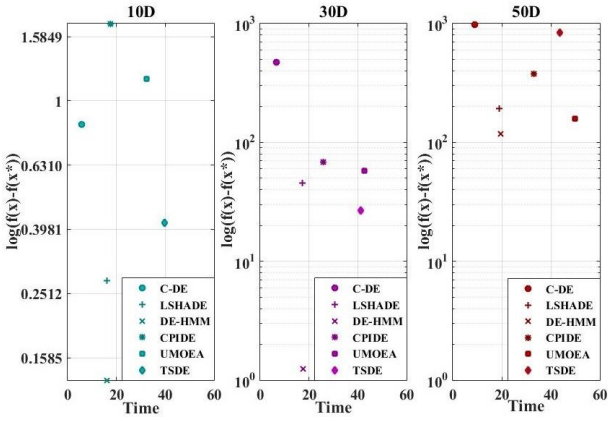

Fig. 11. Time complexity for F18 of CEC 2014 test problems at 10, 30, and 50 dimensions.

2. Abraham, A., Das, S., Konar, A.: Document clustering using differential evolution. In: 2006 IEEE International Conference on Evolutionary Computation, IEEE (2006) 1784–1791
3. Halder, U., Das, S., Maity, D.: A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments. IEEE transactions on cybernetics **43**(3) (2013) 881–897
4. Al-Dabbagh, R.D., Kinsheel, A., Mekhilef, S., Baba, M.S., Shamshirband, S.: System identification and control of robot manipulator based on fuzzy adaptive differential evolution algorithm. Advances in Engineering Software **78** (2014) 60–66
5. Hsu, C.H., Juang, C.F.: Evolutionary robot wall-following control using type-2 fuzzy controller with species-de-activated continuous aco. IEEE Transactions on Fuzzy Systems **21**(1) (Feb 2013) 100–112
6. Kamal, A., MOHD, M., Elshaikh, M., Badlishah, R.: Differential evolution (de) algorithm to optimize berkeley-mac protocol for wireless sensor network (wsn). Journal of Theoretical & Applied Information Technology **89**(2) (2016)
7. Regulwar, D., Choudhari, S., Raj, A.: Differential evolution algorithm with application to optimal operation of multipurpose reservoir. Journal of Water Resource and Protection (2010)
8. Onwubolu, G., Davendra, D.: Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research **171**(2) (2006) 674–692
9. Kheawhom, S.: Efficient constraint handling scheme for differential evolutionary algorithm in solving chemical engineering optimization problem. Journal of Industrial and Engineering Chemistry **16**(4) (2010) 620–628
10. Das, S., Mullick, S.S., Suganthan, P.: Recent advances in differential evolution–an updated survey. Swarm and Evolutionary Computation **27** (2016) 1–30
11. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE transactions on evolutionary computation **1**(1) (1997) 67–82
12. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Trans. Evolutionary Computation **19**(2) (2015) 167–187

13. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE transactions on evolutionary computation **10**(6) (2006) 646–657
14. Goldberg, D.E., Segrest, P.: Finite markov chain analysis of genetic algorithms. In: Proceedings of the second international conference on genetic algorithms. Volume 1. (1987) 1
15. Mahfoud, S.W.: Finite markov chain models of an alternative selection strategy for the genetic algorithm. Complex systems **7**(2) (1993) 155
16. Davis, T.E., Principe, J.C.: A markov chain framework for the simple genetic algorithm. Evolutionary computation **1**(3) (1993) 269–288
17. Rudolph, G.: Finite markov chain results in evolutionary computation: A tour d'horizon. Fundamenta informaticae **35**(1-4) (1998) 67–89
18. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence **127**(1) (2001) 57–85
19. He, J., Yao, X.: Average drift analysis and population scalability. IEEE Transactions on Evolutionary Computation **21**(3) (2017) 426–439
20. Braak, C.J.T.: A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces. Statistics and Computing **16**(3) (2006) 239–249
21. Rees, J., Koehler, G.J.: Learning genetic algorithm parameters using hidden markov models. European Journal of Operational Research **175**(2) (2006) 806–820
22. Hu, Z.b., Su, Q.h., Xiong, S.w., Hu, F.g.: Self-adaptive hybrid differential evolution with simulated annealing algorithm for numerical optimization. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE (2008) 1189–1194
23. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation **15**(1) (2011) 55–66
24. Storn, R.: On the usage of differential evolution for function optimization. In: Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American, IEEE (1996) 519–523
25. Gämperle, R., Müller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems, evolutionary computation **10** (2002) 293–298
26. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Computing **9**(6) (2005) 448–462
27. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: Proc. IEEE CEC. Volume 1. (2005) 506–513
28. Davis, L.: Adapting operator probabilities in genetic algorithms. In: proc. 3rd International conference on genetic algorithms. (1989) 61–69
29. Diao, R., Shen, Q.: Deterministic parameter control in harmony search. In: 2010 UK Workshop on Computational Intelligence (UKCI). (Sept 2010) 1–7
30. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation **13**(2) (2009) 398–417
31. Zaman, M.F., Elsayed, S.M., Ray, T., Sarker, R.A.: Evolutionary algorithms for dynamic economic dispatch problems. IEEE Transactions on Power Systems **31**(2) (2016) 1486–1495
32. Fan, Q., Yan, X.: Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. IEEE Transactions on Cybernetics **46**(1) (2016) 219–232

33. Das, S., Mandal, A., Mukherjee, R.: An adaptive differential evolution algorithm for global optimization in dynamic environments. IEEE transactions on cybernetics **44**(6) (2014) 966–978
34. Islam, S.M., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **42**(2) (2012) 482–500
35. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation **15**(1) (2011) 55–66
36. Liu, Z.Z., Wang, Y., Yang, S., Cai, Z.: Differential evolution with a two-stage optimization mechanism for numerical optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE (2016)
37. Tang, L., Dong, Y., Liu, J.: Differential evolution with an individual-dependent mechanism. IEEE Transactions on Evolutionary Computation **19**(4) (2015) 560–574
38. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing **11**(2) (2011) 1679–1696
39. Elsayed, S.M., Sarker, R.A., Essam, D.L., Hamza, N.M.: Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE (2014) 1650–1657
40. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE transactions on evolutionary computation **13**(5) (2009) 945–958
41. Qiu, X., Xu, J.X., Tan, K.C., Abbass, H.A.: Adaptive cross-generation differential evolution operators for multiobjective optimization. IEEE Transactions on Evolutionary Computation **20**(2) (2016) 232–244
42. Wang, Y., Liu, Z.Z., Li, J., Li, H.X., Yen, G.G.: Utilizing cumulative population distribution information in differential evolution. Applied Soft Computing **48** (2016) 329–346
43. Yu, W.J., Shen, M., Chen, W.N., Zhan, Z.H., Gong, Y.J., Lin, Y., Liu, O., Zhang, J.: Differential evolution with two-level parameter adaptation. IEEE T. Cybernetics **44**(7) (2014) 1080–1099
44. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary computation **11**(1) (2003) 1–18
45. Wang, Y., Li, H.X., Huang, T., Li, L.: Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. Applied Soft Computing **18** (2014) 232–247
46. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2) (2001) 159–195
47. Abbass, H.A.: The self-adaptive pareto differential evolution algorithm. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. Volume 1., IEEE (2002) 831–836
48. Basak, A., Das, S., Tan, K.C.: Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection. IEEE Transactions on Evolutionary Computation **17**(5) (2013) 666–685

49. Aalto, J., Lampinen, J.: A mutation and crossover adaptation mechanism for differential evolution algorithm. In: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE (2014) 451–458
50. Sarker, R.A., Elsayed, S.M., Ray, T.: Differential evolution with dynamic parameters selection for optimization problems. IEEE Transactions on Evolutionary Computation **18**(5) (2014) 689–707
51. Corriveau, G., Guilbault, R., Tahan, A., Sabourin, R.: Bayesian network as an adaptive parameter setting approach for genetic algorithms. Complex & Intelligent Systems (2016) 1–22
52. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE (2014) 1658–1665
53. Mohamed, M.A., Gader, P.: Generalized hidden markov models. i. theoretical frameworks. IEEE Transactions on fuzzy systems **8**(1) (2000) 67–81
54. Baum, L.E., Petrie, T.: Statistical inference for probabilistic functions of finite state markov chains. The annals of mathematical statistics **37**(6) (1966) 1554–1563
55. Rabiner, L., Juang, B.: An introduction to hidden markov models. ieee assp magazine **3**(1) (1986) 4–16
56. Harl, F., Chatelain, F., Gouy-Pailler, C., Achard, S.: Bayesian model for multiple change-points detection in multivariate time series. IEEE Transactions on Signal Processing **64**(16) (2016) 4351–4362
57. Ku, M.L., Chen, Y., Liu, K.J.R.: Data-driven stochastic models and policies for energy harvesting sensor communications. IEEE Journal on Selected Areas in Communications **33**(8) (Aug 2015) 1505–1520
58. Morimoto, H.: Hidden markov models and self-organizing maps applied to stroke incidence. Open Journal of Applied Sciences **6**(03) (2016) 158
59. Cao, Y., Li, Y., Coleman, S., Belatreche, A., McGinnity, T.M.: Adaptive hidden markov model with anomaly states for price manipulation detection. IEEE Transactions on Neural Networks and Learning Systems **26**(2) (Feb 2015) 318–330
60. Lou, H.L.: Implementing the viterbi algorithm. IEEE Signal processing magazine **12**(5) (1995) 42–52
61. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM Computing Surveys (CSUR) **45**(3) (2013) 35
62. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report (2005)
63. Liang, J., Qu, B., Suganthan, P.: Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013)
64. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour a case study on the cec2005 special session on real parameter optimization. Journal of Heuristics **15**(6) (2009) 617–644
65. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1) (2011) 3–18
66. : The spss statistical tool. (2016)

67. Veček, N., Črepinšek, M., Mernik, M.: On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms. Applied Soft Computing (2017)
68. Veček, N., Mernik, M., Črepinšek, M.: A chess rating system for evolutionary algorithms: a new method for the comparison and ranking of evolutionary algorithms. Information Sciences **277** (2014) 656–679
69. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE transactions on evolutionary computation **10**(3) (2006) 281–295
70. Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: 2005 IEEE congress on evolutionary computation. Volume 2., IEEE (2005) 1769–1776