Expert Systems With Applications 115 (2019) 189–199

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Quantum firefly swarms for multimodal dynamic optimization problems

Fehmi Burcin Ozsoydan, Adil Baykasoğlu*

Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, Izmir, Turkey

A R T I C L E   I N F O

Article history:
Received 22 May 2018
Revised 4 August 2018
Accepted 5 August 2018
Available online 6 August 2018

Keywords:
Dynamic optimization
Moving peaks benchmark
Multi-modal optimization
Firefly algorithm
Quantum particles

A B S T R A C T

Optimization problems have attracted attention of researchers for decades. Commonly, problem related data and problem domain are assumed to be exactly known beforehand and to remain stationary. However, numerous real life optimization problems are dynamic. In practice, unpredictable events like due date changes, arrivals of new jobs or cancellations yield to changes in parameters, constraints or variables. In addition to the challenges of traditional stationary optimization problems, diverse parts of the problem space should also be monitored to keep track of moving optima in dynamic problems. Therefore, dividing a population (swarm) into smaller sized sub-swarms is a promising strategy particularly for multi-modal problems. In this context, the present work extends Firefly Algorithm (FA) as a multi-population based algorithm to solve multi-modal dynamic optimization problems due to its popularity and demonstrated competitive performance. Quantum particles are employed to monitor the neighborhoods of the best solutions of each sub-swarm in order to overcome the loss of diversity problem. Quantum strategy is also used to respond to dynamic events. Moreover, economical FA along with a simpler move function is introduced in order to consume fitness evaluations more efficiently. Most of the previous approaches ignore *prioritizing* sub-swarms which can be advantageous. For example, sub-swarms can either be evolved sequentially, randomly or they can be prioritized via some learning-based techniques. Thus, more promising regions might be discovered at earlier evaluations. In this context, the proposed FA extension is further enhanced with such *prioritizing* strategies, which are based on the feedback from sub-swarms. The experiments are conducted on the well-known Moving Peaks Benchmark along with comparisons with well-known methods. The proposed FA is found as promising and competitive according to the outcomes of the comprehensive experimental study.

1. Introduction

It is commonly assumed that the problem related data remains stationary throughout optimization process or while implementing a solution. However, majority of real-life problems are dynamically changing over-time. For instance, problem domain or related parameters might change due to unpredictable events like new job arrivals, cancellations, due date modifications, changes in production constraints or breakdowns in workshops. Such problems are referred to as dynamic optimization problems (DOPs) in the related literature (Branke, 1999; Branke, 2001; Morrison, 2004). In DOPs, rather than finding the global optimum of the problem, the motivation is to keep track of the changing optima.

If a dependency between consecutive problem environments exists, using the gathered information from previous environments might be a more efficient approach in comparison to reinitializing the used algorithm from scratch. Additionally, monitoring diverse parts of the problem space is vital in DOPs. For this reason, the common approach in DOPs community is to divide a population into smaller sized sub-swarms, particularly if the problem is multi-modal. Accordingly, in such strategies, one also needs to decide which sub-swarm should be evolved first, because, there is an opportunity for achieving better results if more promising sub-swarms are encouraged. This is one of the main motivations of the present study. Surprisingly this issue is not paid attention in the related literature. In this regard, first, FA, which is shown to be a promising algorithm in numerous studies (Yang, 2009), is turned into a multi-swarm algorithm in the present work. Next, quantum particles are employed in order to overcome the loss of diversity problem. In this technique, each sub-swarm is comprised of two types of individuals, namely, neutral and quantum fireflies that exhibit different move procedures. While neutral individuals are allowed to follow other fireflies as in the standard FA, quantum particles are scattered within quantum clouds, which are located

* Corresponding author.
E-mail addresses: burcin.ozsoydan@deu.edu.tr (F.B. Ozsoydan), adil.baykasoglu@deu.edu.tr (A. Baykasoğlu).

https://doi.org/10.1016/j.eswa.2018.08.007
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

around the promising regions. Thus, while an implicit local search is performed within a quantum cloud, also diversity is maintained by these fireflies. Additionally, a further modification of this quantum strategy is employed to respond to dynamic events.

Subsequently, a simpler move function is employed along with a strategy that allows reducing the computational expense of the standard FA. This technique is referred to as economical FA (eFA) in the present study. Finally, different sub-swarm prioritizing strategies, including roulette-wheel based selection, are adopted in the developed approaches.

Experiments are conducted on the well-known Moving Peaks Benchmark and the proposed approach is compared with the well-known methods. In regard to the outcomes of the comprehensive experimental study, the proposed strategy is found as promising and competitive.

The rest of the paper is organized as follows: literature review is given in Section 2. The details of the proposed approaches and experimental results are presented in Section 3 and Section 4, respectively. Finally, concluding remarks and future work are given in Section 5.

2. Related work

Solution approaches for DOPs can be classified into four main categories (Baykasoğlu & Ozsoydan, 2017; Cruz, González, & Pelta, 2011; Mavrovouniotis, Li, & Yang, 2017, Ma et al. in press) as presented in the following.

2.1. Introducing diversity methods & adaptation of parameters

In this technique, diversity of a population is increased, whenever it falls below a threshold. One of the easiest methods in this category is to re-initialize the used algorithm with its initial configuration and to solve the changing environments as separate stationary problems (Krishnakumar, 1990). However, complete re-initialization results in loss of history and information gathered throughout generations. In some studies, (Baykasoğlu & Ozsoydan, 2014; Yu and Suganthan, 2009; Hu & Eberhart, 2002; Woldesenbet & Yen, 2009), a partial re-initialization was proposed to overcome this issue.

García-Villoria and Pastor (2009) proposed another diversity introducing method which was used in a Particle Swarm Optimization (PSO). One of the well-known techniques here is hyper-mutation (Cobb, 1990). In this technique, mutation rate is increased as a response to dynamic events. In parallel, Huang, Qin, and Suganthan (2006) and Qin, Huang, and Suganthan (2009) used algorithm related parameters such as scaling factors and crossover rates as self-adaptive parameters. In another study, Lepagnot, Nakib, Oulhadj, and Siarry (2013) presented several local search variants for continuous DOPs. Other adaptation of parameters based techniques of the same category are presented in Cadenas, Garrido, and Munoz (2011) and Karimi, Nobahari, and Pourtakdoust (2012). Wang, Wang, and Yang (2009), Yang and Yao (2005) employed an algorithm selection mechanism that is based on a hyper heuristic framework. Similar strategies were adopted by Topcuoglu, Ucar, and Altin (2014) and Calderín, Masegosa, and Pelta (2015). Chen et al. (2017) introduced a bio-inspired adaptive metaheuristic algorithm that shares algorithm related parameters with individuals of the same population. In one of the recent studies, Altin and Topcuoglu (2018) analyzed the effects of adaptive algorithm parameters on the solution quality.

2.2. Maintaining diversity methods

Maintaining diversity techniques attempt to keep diversity level above a threshold throughout the search. Random immigrants method (Grefenstette, 1992; Tinós & Yang, 2007; Yang, 2008) is a typical example of this category. In this strategy, some random solutions are occasionally inserted into population. A similar technique was implemented by Blackwell and Branke (2004) by employing quantum particles. Inspired by repulsion theory, charged PSO was proposed by Blackwell (2003). Mendes and Mohais (2005), du Plessis and Engelbrecht (2012; 2013) reported similar strategies with Brownian particles to maintain diversity around the best-found solution. Li (2004) introduced an extension of PSO, where the global best particle replaces the neighborhood best in order to avoid premature convergence.

2.3. Memory based methods

Another common approach in DOPs is to generate an archive of the previously found good solutions and to use them in the current environment (Branke, 1999; Branke, 2001; Ma et al., in press; Mavrovouniotis et al., 2017). The use of memory can be classified into sub-categories. In one of them, memory is utilized implicitly via associative procedures, whereas in the second sub-category the past information is explicitly used.

2.3.1. The use of implicit memory

To the best of our knowledge, the use of implicit memory was introduced by Goldberg and Smith (1987). In this method, redundant solution representation with diploid genomes is utilized. Thus, memory can implicitly be integrated into any metaheuristic algorithm via solution representations. A decade later, Hadad and Eick (1997) proposed an extension of the same strategy. Implementations of this category in DOPs were presented in Uyar and Harmanci (2005) and S. Yang (2006). In a recent study, Cao, Xu, and Goodman (2018) analyzed the performance of PSO enhanced by adopting a neighbor-based learning method with short-term and long-term memory.

2.3.2. The use of explicit memory

In this method, previously found good solutions are occasionally used either to introduce diversity or to forecast the promising regions of the upcoming problem environment. In other words, when a dynamic event is detected, the recorded memory is utilized while generating new individuals or relocating the existing ones.

Karaman, Uyar, and Eryiğit (2005) reported a memory indexing evolutionary algorithm, which was inspired by hyper-mutation and distribution estimation. S. Yang (2006) and Yang, Cheng, and Wang (2010) proposed generating new individuals to increase diversity by making use of memory. In some other studies, (Pelta, Cruz, & Verdegay, 2009; Richter, 2010; Richter & Yang, 2008; Richter & Yang, 2009) probabilities of dynamic events, fitness landscapes of promising and feasible regions were derived from memory records. Turky and Abdullah (2014b) employed an external archive with a harmony search algorithm. Mukherjee, Patra, Kundu, and Das (2014) used a similar technique supported by a clustering strategy. In another study, Nasiri, Meybodi, and Ebadzadeh (2016) employed a history-driven PSO, where a binary space partitioning tree structure was used to record promising solutions.

2.4. Multi-population based methods

The strategy of using multiple populations was introduced by Goldberg and Richardson (1987). A decade later, Petrowski (1996) extended this technique by utilizing sharing concepts within the same niches. This approach was later adopted in Bird and Li (2006). In another work, Ursem (1999) proposeed making use of multiple nations approach, where a nation is comprised of government, population and a policy. Inspired by forking GA of Branke, Kaußler, Smidt, and Schmeck (2000), Tsutsui et al. (1997) introduced self-organizing scouts (SOS). In SOS, search is initialized as in traditional GAs; however, after reaching a previously defined level of convergence, a part of the population is separated from whole population in order to form up a new sub-population. The notion of using core and sub-populations as parent and child swarms was adopted also by Kamosi, Hashemi, and Meybodi (2010), Kordestani, Rezvanian, and Meybodi (2014) and Li and Yang (2008). Li, Branke, and Blackwell (2006) and Parrott and Li (2006) further improved another novel study known as Speciation Based PSO (SPSO). In this approach, a sub-population is comprised of particles within a distance from the best particle (seed) of the corresponding species. One year later, in Bird and Li (2007), a new approach (rSPSO), utilizing both evolutionary and statistical methods such as regression analysis, was proposed. An optimization algorithm that is inspired by charged atomic particles, was reported by Blackwell and Bentley (2002). Similar idea along with several enhancements was adopted by Blackwell and Branke (2006). In that paper, two distinct algorithms, namely, mQSO and mCPSO were proposed. Janson and Middendorf (2004) analyzed the effects of hyerarchial sub-swarms. Du and Li (2008) introduced an algorithm, which they call multi-strategy ensemble particle swarm optimization (MEPSO). The main idea here is to split the whole population into two sub-swarms, referred to as part I and part II. Li and Yang (2009) and Yang and Li (2010) proposed making use of clustering techniques (CPSO). Blackwell, Branke, and Li (2008) utilized quantum particles as well as other procedures like anti-convergence and exclusion. A hibernating (sleeping-awaking) method, which temporarily terminates search of some sub-swarms, was utilized in some studies (Kamosi, Hashemi, & Meybodi, 2010;Yazdani, Nasiri, Sepas-Moghaddam, & Meybodi, 2013). Li, Yang, and Yang (2014) presented a self-adaptive approach for DOPs. The study of Mendes and Mohais (2005) was further improved in du Plessis and Engelbrecht (2013), where a multi population based Differential Evolution Algorithm with several extensions was proposed. Using cellular based approach by partitioning search space into smaller scaled sub-regions is another approach (Hashemi & Meybodi, 2009; Noroozi, Hashemi, & Meybodi, 2011) of this category. Nasiri et al. (2016) presented a recent multi-population PSO extension, which adopts a peak finding strategy similar to that of the finder tracker PSO's (Yazdani et al., 2013). Turky and Abdullah (2014a) employed a multi-population based Electromagnetic Algorithm with several modifications. Next, Turky and Abdullah (2014b) and Turky, Abdullah, and Dawod (2018) extended a harmony search algorithm as a multi-population based approach. Nseef, Abdullah, Turky, and Kendall (2016) proposed an artificial bee colony algorithm, where the number of sub-populations were tuned adaptively. Boulesnane and Meshoul (2017) introduced a new optimizer for DOPs that draws inspiration from partitioning a population into smaller-scaled sub-swarms. Recently, Kordestani, Firouzjaee, and Meybodi (2018) presented an adaptive bi-flight cuckoo search algorithm that is further enhanced by adopting a local search method.

3. The proposed solution approach

FA is one of the popular bio-inspired optimization algorithms. It draws inspiration from flashing and movement mechanisms of fireflies (Yang X-S, 2009). Although it was tested in numerous applications particularly in engineering optimization problems, FA is rarely used in DOPs. As one can see from the latest surveys (Fister, Fister Jr, Yang, & Brest, 2013; Mavrovouniotis et al., 2017), FA is less frequently employed in solving DOPs in comparison to other swarm intelligence based algorithms. In one of these studies, Farahani, Nasiri, and Meybodi (2011) proposed a variant of multiple population based approach for FA. Similar works were reported by Nasiri and Meybodi (2012) and Ozsoydan and Baykasoglu (2015). An interesting application, where FA is used for determining the optimal location and size of distributed generation, was presented by Sulaiman, Mustafa, Azmi, Aliman, and Rahim (2012). Baykasoğlu and Ozsoydan (2014) proposed a discrete optimization application of FA for DOPs.

It is apparent that PSO and its various extensions achieved excellent results in dynamic optimization benchmarking problems (Fister et al., 2013; Mavrovouniotis et al., 2017). However, it should also be noted that, after detecting dynamic events, one needs to re-evaluate the best solution vectors of PSO based algorithms. In other words, unless best positions are simply equalized to the new positions of existing solutions that are re-evaluated subsequent to dynamic events, more fitness function evaluations are required while adapting to the new environment. For this reason, researchers are interested in exploring the use of fitness function evaluations more efficiently.

The standard FA suffers from the complexity of $O(n^2)$ in movement procedure of fireflies. As a result, in comparison to a standard population-based algorithm with $O(n)$ complexity, FA can consume more function evaluations in a single iteration. The present study introduces a modification on the movement procedures of FA in order to use fitness evaluations more economically. For a better understanding, the standard FA is presented next.

3.1. The standard FA

Brightness of a firefly represents the quality of a solution vector in FA. A brighter firefly means a better solution and it attracts other fireflies. As the distance between fireflies increases, the attractiveness decreases due to light absorption of medium. FA has three basic assumptions:

i All fireflies are unisexual and every firefly attracts/gets attracted to every other firefly.
ii The attractiveness of a firefly is directly proportional to the brightness of that firefly (The brightness decreases as the spatial distance between fireflies increases.).
iii Fireflies move randomly if they do not find a more attractive firefly in the population.

The Euclidean distance between two fireflies (i and j) is evaluated by Eq. 1, where D is the dimension, x_{id} is the dth dimension of the ith firefly.

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{d=1}^{D} \left(x_{id} - x_{jd}\right)^2} \tag{1}$$

The attractiveness of a firefly at a distance r is formulated by Eq. 2, where γ is the light absorption coefficient, m is a number modifying the distance metric and $\beta_{(0)}$ is the attractiveness at 0 distance.

$$\beta_{(r)} = \beta_{(0)} e^{-\gamma r^m} \quad m \geq 1 \tag{2}$$

Finally, if a firefly i gets attracted to another more attractive firefly j, then it is moved in the direction of the jth firefly according to Eq. 3, where α is a user supplied randomness scaling parameter and $\psi \in [0, 1]$ is a uniform random number. A pseudo code for the standard FA is given in Algorithm 1.

$$x_{id} = x_{id} + \beta_{(0)} e^{-\gamma r^m} \left(x_{jd} - x_{id}\right) + \alpha(\psi - 0.5) \tag{3}$$

3.2. The proposed eFA and its subroutines

3.2.1. Movement in eFA

It is clear from Eq. 3 that as the distance between fireflies increases, $\beta_{(0)} e^{-\gamma r^m}$ term approaches to 0 and fireflies can get stuck

Algorithm 1. Pseudo code for the standard FA.

```
begin
  generate initial population $X_i$, $i = 1 \ldots popSize$;
  evaluate fitness values $f(X_i)$, $i = 1 \ldots popSize$;
  while evalMax not reached do
    for ($i = 1$; $i \le popSize$; $i{+}{+}$) do
      for ($j = 1$; $j \le popSize$; $j{+}{+}$) do
        if $f(X_j) > f(X_i)$ then
          move $X_i$ towards $X_j$ in all directions;
        end
        attractiveness varies with dist. r via $e^{-\gamma r^2}$
        evaluate new solutions and update light intensity
      end
    end
    rank fireflies and find the current best
  end
  results and visualization
end
**evalMax: maximum number of function evaluations
**popSize: population size
```

in their current positions throughout generations. In order to avoid this, γ can precisely be set to appropriate values, however, obtaining an appropriate level for γ is not an easy task in DOPs. Therefore, a simpler move function which was proposed by Baykasoğlu and Ozsoydan (2014, 2018) as formulated by Eq. 4, is also adopted here. Ω is a parameter to avoid division by zero and it is fixed to unity. Finally, the best firefly, which cannot get attracted by any other fireflies, is moved in regard to the formulation given by Eq. 5. If the trial vector improves it, the best firefly replaces this new solution. Otherwise, it is rejected.

$$x_{id} = x_{id} + \beta_{(0)}\left({}^{1}\!/\!_{(\Omega + r)}\right)\left(x_{jd} - x_{id}\right) + \alpha(\psi - 0.5) \tag{4}$$

$$x_{trial,d} = x_{best,d} + \alpha(\psi - 0.5) \tag{5}$$

As stated before, the main motivation in the proposed eFA is using function evaluations more economically. In this regard, fitness values of fireflies are evaluated subsequent to ending of all movements. In other words, new fitness value of a firefly is evaluated only after it is moved in the direction of all other brighter fireflies in the swarm. Furthermore, because finding promising solutions as fast as possible is crucial in DOPs (due to the performance metrics, which will be clarified later), better solutions are encouraged to move first. Therefore, first, fireflies are ranked in decreasing order according to their fitness values, where the first rank firefly represents the best solution vector of the corresponding swarm. Thus, if a dynamic event is triggered throughout the evolution of any swarm, remaining evaluations just before that dynamic event are devoted to brighter (more promising) fireflies.

In eFA, a naïve firefly hierarchically interacts with all other brighter fireflies but its fitness is evaluated only after all interactions. For instance, let's assume that 4 fireflies exist in a swarm. The first step is to rank the fireflies in regard to their decreasing fitness values. The first move is devoted to the brightest firefly. It flies randomly via Eq. 5. Next, the *2*nd rank firefly is moved towards the *1*st rank firefly and then its fitness is evaluated. After the movement of the *2*nd rank firefly, the *3*rd rank firefly is moved towards the *2*nd and the *1*st rank fireflies in sequence and then its new fitness is evaluated. Finally, the *4*th rank firefly is moved towards the 3rd, the *2*nd and the *1*st rank fireflies in sequence and then its fitness is evaluated. Thus, it is guaranteed that the total number of fitness function evaluations is equal to the number of population size in a single iteration.

3.2.2. Employing quantum particles in eFA

Based on the repulsion theorem, Blackwell and Bentley (2002) proposed making use of charged particles along with neutral particles in order to main diversity. Later, Blackwell and Branke (2004) further extended this atomic swarm by introducing a quantum model that is inspired by movements of charged particles. In a classical atomic structure, electrons orbit around the nucleus at some distance. However, rather than following a deterministic path as in classical structure, in a quantum model, electrons are randomly distributed within a sphere (Blackwell et al., 2008). This model is adopted here.

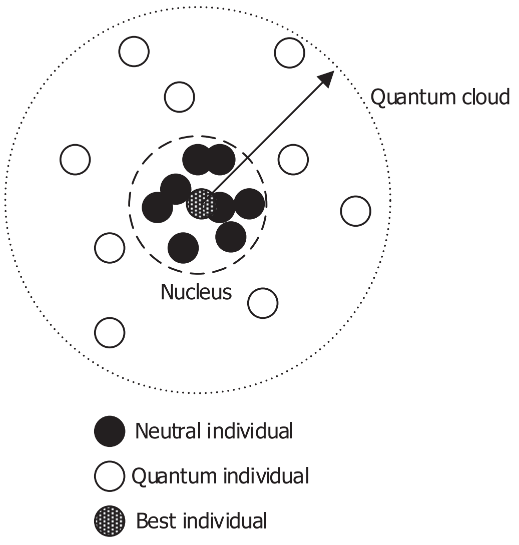

Fig. 1. A snapshot of a quantum sub-swarm.

Algorithm 2. A pseudo code for generating quantum fireflies.

```
begin
  proceed = 1;
  while proceed == 1 do
    generate a random Gaussian vector $v_j$ from $N(0,1)$ for $1 \le j \le D$
    calculate the spatial distance of V from the origin, $dist = \sqrt{\sum_{j=1}^{D} v_j^2}$
    generate a uniform random number $u \in [0, 1]$
    for ($j = 1$; $j \le D$; $j{+}{+}$) do
      $q_j = x_{best,j} + v_j \times r_{cloud} \times \sqrt[D]{u}/dist$
    end
    if feasbiliy of quantum firefly is maintained
      proceed = 0
    end
  end
end
**D:dimension of the Gaussian vector
**$q_j$: jth dimension of the quantum firefly
```

The swarms employed in eFA are comprised of two types of solution vectors, namely neutral fireflies, and quantum fireflies. Neutral fireflies are placed in a nucleus and they move according to the formulations given in Eqs. 4-5. The rest of a swarm is comprised of quantum particles, which are randomly distributed at a distance from the center of a sub-swarm. Center of a sub-swarm is chosen as the best solution vector found by the related sub-swarm and the mentioned distance is defined by a user supplied parameter denoted by r_{cloud}. This parameter is used for generating a space (quantum cloud), where the quantum particles can be scattered. Such a swarm, comprised of both quantum and neutral fireflies is referred to as quantum swarm in the present study and it is depicted in Fig. 1. A pseudo code for generating quantum particles is presented in Algorithm 2.

As one can see from Fig. 1, the fireflies in nucleus come closer to each other throughout generations and as a result, they are tightly clustered around the best particle, while quantum particles maintain diversity. It should be stressed that

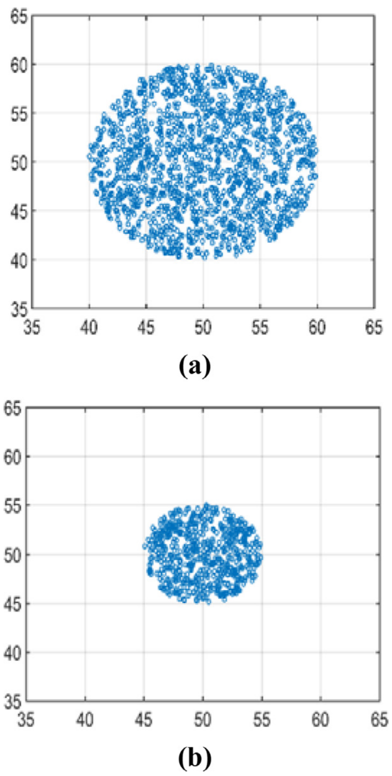

Fig. 2. An example of a uniformly distributed quantum cloud located on the coordinates of (50,50) **a)** with $r_{cloud} = 10$ and **b)** with $r_{cloud} = 5$.

Algorithm 3 A pseudo code for the eFA.

```
begin
  generate initial population $X_i$, $i$=1...$popSize$;
  evaluate fitness values $f(X_i)$, $i$=1...$popSize$;
  Rank the fireflies in regard to the decreasing fitness values
  while evalMax not reached do
    for ( $j$=1; $i \le popSize$; $i$++) do
      if ($j$==1)
        move the jth (the best) firefly randomly via Eq. 5
        attractiveness varies with dist. r via 1/r
        evaluate new $X_j$ and update it if necessary
      end
      if ($j$>1 and $j \le popSize-quantumSize$)
        for ($i$=1; $i \le j$-1; $i$++) do
          move $X_j$ towards $X_{j-i}$ in all directions via Eq. 4
          evaluate new fitness of $X_j$
        end
      end
      if ( $j$ >$popSize-quantumSize$)
        perform a quantum move for $X_j$
        evaluate new fitness of quantum firefly Q
        if $f(Q) > f(X_j)$
          $X_j = Q$
          $f(X_j) = f(Q)$
        end
      end
    end
  end
end
** evalMax : maximum number of function evaluations
**quantumSize : number of quantum fireflies in the sub-swarm
**popSize : total number of fireflies in the sub-swarm
```

Blackwell et al. (2008) reported several types of probability distributions that can be employed while generating a quantum cloud and distributing particles. However, because it generates a homogenous cloud, uniform volume distribution is used in the present study. Examples of uniformly scattered fireflies with radii $r_{cloud} = 10$ and $r_{cloud} = 5$, are presented in Fig. 2.a and Fig. 2.b, respectively. The best solution in this figure is located on the coordinates of (50, 50).

Finally, a pseudo code for eFA is presented in Algorithm 3. As one can see from this algorithm, some fireflies (*quantumSize*) are labeled as quantum fireflies, while the rest of the population (*popSize-quantumSize* = *neutralSize*) is compired of neutral fireflies. If the new fitness of a quantum particle is better than its current value, it replaces that new solution vector, otherwise it is rejected (Algorithm 3). It should be noted that this pseudo code is modified for a single sub-swarm. The next section is devoted to extending eFA as a multi-population based approach.

3.3. Extending quantum eFA as a multi-population based approach

In the present study, it is assumed that all sub-swarms have identical number of neutral and quantum fireflies. Thus, population size of each sub-swarm is assumed to be equal. Furthermore, for simplicity, a fixed number of sub-swarms are allowed to exist throughout evaluations.

3.3.1. Prioritizing sub-swarms in multi-population eFA

After initializing all individuals, a sub-swarm among all existing sub-swarms is selected to evolve. Sub-swarm selection mechanism can be considered as a crucial decision that clearly affects the success of the used algorithm. Its importance can be explained in a few words as in the following.

If always the best sub-swarm of the current environment is encouraged to evolve, the used algorithm might perform better for a short time. However, subsequent to a dynamic event, if the current optimum moves to another position, performance of the algorithm might possibly suffer dramatically, because, other sub-swarms, which are scattered around diverse locations of the problem space, have not been allowed to evolve yet. Therefore, a balance should be established while selecting a sub-swarm to evolve. du Plessis and Engelbrecht (2012; 2013) presented a competitiveness based approach while selecting a sub-swarm to evolve.

In the present study, as sequential selection and a roulette wheel selection, two different sub-swarm selecting strategies are employed. In sequential selection, the next sub-swarm to evolve is selected sequentially. Therefore, it can be put forward that each sub-swarm has equal chance to evolve. Roulette wheel selection strategy performs a roulette wheel selection, which is based on the current performances of the best fireflies in sub-swarms. Thus, more promising sub-swarms are encouraged.

In this strategy, first, current error values of best fireflies of sub-swarms are evaluated as formulated in Eq. 6, where $f_k(t)$, $e_k(t)$, $opt(t)$ represent the fitness of the best firefly of the kth sub-swarm, best-error of the kth sub-swarm, and the optimum at the tth evaluation. Since the aim here is to minimize the error, $e_k(t)$ values obtained via Eq. 6 cannot be used. Therefore, the values of $e_k(t)$ are modified according to the formulation given in Eq. 7, where $em_k(t)$ and K represent the modified error of the kth sub-swarm at the tth evaluation and the total number of existing sub-swarms, respectively. Next, $em_k(t)$ values of all sub-swarms are increased by the modified error of the 2nd worst sub-swarm as given in Eq. 8, where $ei_k(t)$ represents the increased error of the kth sub-swarm at the tth evaluation. Finally, roulette wheel is applied based on the $ei_k(t)$ values, where a greater $ei_k(t)$ represents a greater chance to be selected. In the rest of the paper, the multi-population algorithm adopting a roulette wheel-based selection strategy is referred to as eFA-rw, while the algorithm using a sequential-based

Algorithm 4. A pseudo code for the proposed multi-population eFA.

```
begin
  set the initial values of parameters   generate all sub-swarms
  while maxEval not reached do
    check for dynamic events
    if a dynamic event is detected
      convert all fireflies to quantum fireflies and reposition them
      re-evaluate all existing individuals
    end
    select a sub-swarm to evolve
    evolve the selected sub-swarm
    perform exclusion
  end
  results and visualization
end
** maxEval: maximumum number of allowed fintess function evaluations
```

selection referred to as eFA-seq.

$$e_k(t) = (opt(t) - f_k(t)) \tag{6}$$

$$em_k(t) = \max_{q=1\ldots K}\{e_q(t)\} - e_k(t) \tag{7}$$

$$ei_k(t) = em_k(t) + \min_{q=1\ldots K}\{em_q(t)\,|em_q(t) \neq 0\} \tag{8}$$

3.3.2. Exclusion of sub-swarms

In a multi-population based approach, sub-swarms might occasionally converge to the same promising regions. As a result, some of the peaks become overcrowded. In such a case, in order not to waste function evaluations on the same peak, sub-swarms with worse performances are excluded. This procedure is referred to as exclusion operation.

There are several exclusion techniques in the related literature. One of the most widely-known strategies (Blackwell & Branke, 2004; Blackwell et al. 2008) is adopted here. In this technique, if the spatial distance between the best fireflies of any two sub-swarms is less than the exclusion radius (r_{excl}), it is concluded that these two sub-swarms overlap. As a result, only the outperforming sub-swarm is allowed to survive while the other sub-swarm is reinitialized. Exclusion radius is evaluated based on Eq. 9, where R, M and D represent the range of the variables, the number of the peaks and the number of dimensions used in the problem, respectively.

$$r_{excl} = \frac{R}{2M^{\frac{1}{D}}} \tag{9}$$

3.3.3. Detecting dynamic events and responses

In order to detect dynamic events, sentry point method is employed. In this technique, sentry point is selected as the best firefly among all existing sub-swarms. In parallel with some reported studies (du Plessis & Engelbrecht, 2012; du Plessis & Engelbrecht, 2013; Turky & Abdullahi, 2014a; Turky & Abdullahi, 2014b), the best firefly is revaluated subsequent to evolution of a selected sub-swarm. A possible change in the fitness of the best firefly points out a dynamic event.

As a response to a dynamic event, first, all fireflies in all existing sub-swarms are temporarily converted to quantum fireflies, and they are repositioned within a quantum cloud, centered on the coordinates of the best fireflies of the related sub-swarms just before the dynamic event. Next, new fitness values of all fireflies are evaluated. Finally, putting things together, the flow of the proposed approach is summarized in Algorithm 4.

Table 1
MPB parameter settings.

parameter	*values*	*additional test cases*
number of peaks, M	10	1,2,5,20,30,50,100,200
frequency	5000	
heigth_severity	7.0	
width_severity	1.0	
peaks shape	cone	
basic function	No	
shift length (s)	1.0	2.0, 3.0, 4.0, 5.0
number of dimensions (D)	5	
correlation parameter (λ)	0.0	
peak coordinates range	[0,100]	
peaks height range	[30,70]	
peaks width range	[1,12]	

4. Experimental study

Due to its popularity in DOPs, Moving Peaks Benchmark (MPB) (Branke, 1999) is used here to test the proposed approaches. In MPB, there exist a number of peaks located on a D-dimensional problem domain with heights, widths and coordinates change after reaching a predefined number of fitness function evaluations, referred to as *frequency*. The aim here is to locate and to track the moving optima, which is indeed the highest peak.

The heights, widths and coordinates of the peaks are shifted in regard to Eqs. 10-13, where $H_i(t)$, $W_i(t)$ and $\vec{X}_i(t)$ represent the height, width and coordinates of the ith peak at the tth environment, respectively. Locations of the peaks are shifted by the vector $\vec{v}_i$, which is a function of a random vector $\vec{r}$, shift severity s and correlation parameter λ. Since it is one of the best-known benchmarks in DOPs, the MPB Scenerio-2 is adopted here. Additional tests along with canonical data are presented in Table 1.

$$H_i(t) = H_i(t-1) + heigth\_severity \times \rho_h,\ \ \rho_h \sim N(0,1) \tag{10}$$

$$W_i(t) = W_i(t-1) + width\_severity \times \rho_w,\ \ \rho_w \sim N(0,1) \tag{11}$$

$$\vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v}_i(t) \tag{12}$$

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1))\ ,\quad 0 \le \lambda \le 1 \tag{13}$$

In order to analyze the efficiency of the developed algorithms, *offline_error* (Branke & Schmeck, 2003), which is one of the most commonly used performance metrics, is adopted here. It can be formulated as given in Eq. 14, where *maxEval* and e_t^* represent the maximum number of allowed function evaluations for a run and the best error found by the algorithm since the last change at the tth function evaluation, respectively. It is simply the average of all current errors until the tth evaluation.

$$offline\_error = \frac{1}{maxEval}\sum_{t=1}^{maxEval} e_t^* \tag{14}$$

For each run, 100 dynamic environments are allowed. For example, for *frequency* = 5000, it results in 100 × 5000 = 500,000 fitness function evaluations to terminate a run. All reported results are evaluated over 50 independent runs with different random seeds.

4.1. Fine tuning of parameters

Whiletuning the parameters of the proposed approach, default values of MPB (Table 1) are used. Additional tests are performed later in the experimental analysis. Since the number of peaks is fixed to 10 in this scenario, the number of existing sub-swarms is

Table 2
Tuning of $\beta_{(0)}$.

offline_error± *std_error*	$\beta_{(0)}$
2.7806 ± 0.0674	4.0
2.6019 ± 0.0451	3.6
2.5153 ± 0.0498	3.2
2.4994 ± 0.0665	2.8
2.4969 ± 0.0642	2.3
2.4546 ± 0.0465	2.2
2.4351 ± 0.0608	2.1
2.4564 ± 0.0520	2.0
2.3451 ± **0.0568**	**1.9**
2.5008 ± 0.0769	1.8
2.4925 ± 0.0682	1.7
2.4278 ± 0.0575	1.6
2.4001 ± 0.0463	1.2
2.5133 ± 0.0506	1.0
2.7143 ± 0.0524	0.6
3.2191 ± 0. 0742	0.2

Table 3
The effects of r_{cloud}.

offline_error± *std_error*	r_{cloud}
1.8737 ± 0.0511	1.40
1.7965 ± 0.0630	1.20
1.7474 ± **0.0527**	**1.00**
1.7760 ± 0.0620	0.90
1.7525 ± 0.0736	0.80
1.7609 ± 0.0641	0.60
1.7792 ± 0.0626	0.40
1.7988 ± 0.0734	0.20
1.7814 ± 0.0674	0.10
1.8782 ± 0.0765	0.08
1.8900 ± 0.0801	0.06
2.1937 ± 0.0797	0.02

Table 4
Tuning of α.

offline_error± *std_error*	α
2.9798 ± 0.0585	0.01
2.2086 ± 0.0570	0.04
1.7409 ± 0.0633	0.10
1.7067 ± 0.0635	0.20
1.6782 ± 0.0728	0.25
1.5842 ± **0.0525**	**0.30**
1.7077 ± 0.0454	0.35
1.7963 ± 0.0566	0.40
1.8380 ± 0.0571a	0.50
1.9333 ± 0.0486	0.60
2.0115 ± 0.0499	0.70
2.2596 ± 0.0449	0.90
2.5630 ± 0.0716	1.10
3.0799 ± 0.0636	1.60
3.2182 ± 0.0544	2.00

Table 6
Allowed number of sub-swarms and fireflies.

number of peaks	*number of sub-swarms used*	$N+Q$
1	1	8 + 2
2	2	8 + 2
5	5	8 + 2
10	10	8 + 2
20	20	4 + 1
30	20	4 + 1
50	30	4 + 1
100	50	4 + 1
200	50	4 + 1

also fixed to 10. All printed results in parameter tuning tests are the averages of 50 runs, where 100 dynamic environments are allowed. Parameter tuning tests are performed by using eFA-seq.

First, the step size of the difference vector $\beta_{(0)}$ is tuned. While tuning this parameter, the number of the neutral fireflies (*neutralSize*), quantum fireflies (*quantumSize*), r_{cloud} and α are fixed to 8, 2, 1.0 and 1.0, respectively. Obtained results for varying $\beta_{(0)}$ are presented in Table 2. According to this analysis, $\beta_{(0)}$ is fixed to 1.9 throughout the rest of tests.

The next parameter to tune is selected as r_{cloud}. In order to see the effects of this parameter more clearly, both *quantumSize* and *neutralSize* are fixed to 5, while $\beta_{(0)}$ and α are fixed to 1.9 and 0.3, respectively. The obtained results for different levels of r_{cloud} are printed in Table 3. According to this anlaysis, it is concluded that the best results can be achieved when r_{cloud} is fixed to unity.

Next, the randomness parameter α is tuned. In this analysis the parameters $\beta_{(0)}$, *neutralSize, quantumSize* and r_{cloud} are fixed to 1.9, 8, 2 and 1.0, respectively. Other test conditions are identical to the previous tests. The obtained results are presented in Table 4. According to these results, the best *offline_error* value is obtained when the parameter α is fixed to 0.30.

Another test is conducted to determine the proportion of quantum and neutral particles in a sub-swarm. In this respect, this analysis is performed under three different values of *popSize* (10, 8 and 6). All possible combinations are tested and the results are presented in Table 5, where $N+Q$ columns denote the number of neutral and the number of quantum particles in a sub-swarm. The paremeters r_{cloud}, $\beta_{(0)}$ and α are fixed to 1.0, 1.9 and 0.3, respectively.

Finally, it is observed that increasing the number of existing swarms due to peak increase, yields to better results. However, additional sub-swarms cost additional function evaluations. In order to keep a balance, while the number of existing sub-swarms is increased, allowed number of quantum and neutral fireflies are decreased. The related configuration is presented in Table 6.

Table 5
The effects of the number of neutral and quantum particles.

offline_error ± *std_error*	$N+Q=10$	*offline_error* ± *std_error*	$N+Q=8$	*offline_error* ± *std_error*	$N+Q=6$
2.7498 ± 0.0612	0 + 10	2.6312 ± 0.0563	0 + 8	2.8466 ± 0.0555	0 + 6
2.7492 ± 0.0641	1 + 9	2.8354 ± 0.0785	1 + 7	2.7961 ± 0.0601	1 + 5
2.1859 ± 0.0610	2 + 8	1.9936 ± 0.0481	2 + 6	1.9647 ± 0.0622	2 + 3
1.9471 ± 0.0618	3 + 7	1.8322 ± 0.0546	3 + 5	1.8364 ± 0.0654	3 + 3
1.7321 ± 0.0504	4 + 6	1.7717 ± 0.0589	4 + 4	**1.6724** ± **0.0622**	**4 + 2**
1.7474 ± 0.0527	5 + 5	1.8249 ± 0.0663	5 + 3	1.6827 ± 0.0588	5 + 1
1.6950 ± 0.0572	6 + 4	1.6350 ± 0.0612	6 + 2	1.9804 ± 0.0726	6 + 0
1.6473 ± 0.0658	7 + 3	**1.6257** ± **0.0530**	7 + 1		
1.5842 ± **0.0525**	**8 + 2**	2.0641 ± 0.0710	8 + 0		
1.6754 ± 0.0497	9 + 1				
2.2485 ± 0.0777	10 + 0				

Table 7
Performances of the algorithms (*offline_error*± *std_error*).

		number of peaks								
ref.	*algorithm*	1	2	5	10	20	30	50	100	200
Blackwell and Branke (2006)	mCPSO-with anticonvergence	4.93 ± 0.17	3.36 ± 0.26	2.07 ± 0.08	2.08 ± 0.07	2.64 ± 0.07	2.63 ± 0.08	2.65 ± 0.06	2.49 ± 0.04	2.44 ± 0.04
Blackwell and Branke (2006)	mCPSO-without anticonvergence	4.93 ± 0.17	3.36 ± 0.26	2.07 ± 0.08	2.05 ± 0.07	2.95 ± 0.08	3.38 ± 0.11	3.68 ± 0.11	4.07 ± 0.09	3.97 ± 0.08
Blackwell and Branke (2006)	mQSO-with anticonvergence	5.07 ± 0.17	3.47 ± 0.23	1.81 ± 0.07	1.80 ± 0.06	2.42 ± 0.07	2.48 ± 0.07	2.50 ± 0.06	2.36 ± 0.04	2.26 ± 0.03
Blackwell and Branke (2006)	mQSO-without anticonvergence	5.07 ± 0.17	3.47 ± 0.23	1.81 ± 0.07	1.75 ± 0.06	2.74 ± 0.07	3.27 ± 0.11	3.65 ± 0.11	3.93 ± 0.08	3.86 ± 0.07
Bird and Li (2007)	SPSO	2.64 ± 0.10	2.31 ± 0.11	2.15 ± 0.07	2.51 ± 0.09	3.21 ± 0.07	3.64 ± 0.07	3.86 ± 0.08	4.01 ± 0.07	3.82 ± 0.05
Bird and Li (2007)	rSPSO	1.42 ± 0.06	1.10 ± 0.03	1.04 ± 0.03	1.50 ± 0.08	2.20 ± 0.07	2.62 ± 0.07	2.72 ± 0.08	2.93 ± 0.06	2.79 ± 0.05
Hashemi and Meybodi (2009)	CLPSO*	2.55 ± 0.12	–	1.68 ± 0.11	1.78 ± 0.05	2.61 ± 0.07	2.93 ± 0.08	3.26 ± 0.08	3.41 ± 0.07	3.40 ± 0.06
Li and Yang (2008)	FMSO*	3.44 ± 0.11	–	2.94 ± 0.07	3.11 ± 0.06	3.36 ± 0.06	3.28 ± 0.05	3.22 ± 0.05	3.06 ± 0.04	2.84 ± 0.03
Hu Eberhart (2002)	RPSO*	0.56 ± 0.04	–	12.22 ± 0.76	12.98 ± 0.48	12.79 ± 0.54	12.35 ± 0.62	11.34 ± 0.29	9.73 ± 0.28	8.90 ± 0.19
Li et al. (2006)	BSPSO	–	–	3.20 ± 0.17	3.04 ± 0.13	–	–	5.06 ± 0.12	5.38 ± 0.12	5.24 ± 0.13
Li et al. (2006)	RWS	–	–	3.50 ± 0.21	3.04 ± 0.09	–	–	3.97 ± 0.08	4.07 ± 0.09	3.95 ± 0.06
Li et al. (2006)	SPSO-PD	–	–	1.98 ± 0.05	1.98 ± 0.05	–	–	3.47 ± 0.06	3.60 ± 0.07	3.47 ± 0.07
Li et al. (2006)	SPSO-PD**	–	–	2.10 ± 0.07	2.10 ± 0.06	–	–	3.81 ± 0.10	3.68 ± 0.07	3.43 ± 0.05
eFA-seq		0.64 ± 0.03	0.76 ± 0.08	1.06 ± 0.05	1.58 ± 0.05	2.22 ± 0.04	2.64 ± 0.03	2.91 ± 0.03	3.27 ± 0.04	3.44 ± 0.03
eFA-rw		0.64 ± 0.03	0.66 ± 0.04	1.21 ± 0.09	1.63 ± 0.05	2.16 ± 0.05	2.50 ± 0.04	2.70 ± 0.03	3.03 ± 0.03	3.14 ± 0.03

*Adopted from Yazdani et al. (2013)
**no conversion from quantum particles is allowed

4.2. Numerical results

The results obtained by both eFA-seq and eFA-rw are given in Table 7. They are compared to the well-known algorithms, which are assumed to be the fundamental benchmarking algorithms in the DOPs domain.

Comparing eFA-seq and eFA-rw with other algorithms, one can see that several results are outperformed by the proposed approaches. It is also clear from these results that, although eFA-seq and eFA-rw do not outperform all algorithms in all benchmarks, these approaches can be considered as competitive and promising algorithms.

For the single peak environment, both eFA-seq and eFA-rw can be considered as equivalent approaches. However, for the rest of the problems, they differ in terms of both *offline_error* and *std_error*. It can be seen from Table 7 that as the number of existing peaks in the problem environment increases, the efficiency of eFA-rw becomes more apparent. This is one of the expected results in this study. Since, giving priority to more promising sub-swarms allocates more function evaluations around promising regions. On the other hand, all sub-swarms have equal chances to evolve in eFA-seq. Nevertheless, in the instances with 5 and 10 peaks, eFA-seq seems to be performing better than eFA-rw. Finally, the analysis, which is based on the varying number of peaks, is depicted in Fig. 3.

In the above analysis, eFA-rw seems to be performing better than eFA-seq, particularly in the instances with greater number of peaks. However, further statistical analysis is required here to demonstrate whether a significant difference between the performances of eFA-rw and eFA-seq exists. This analysis might include both parametric and non-parametric tests. Particularly, in stochastic conditions, where safe use of parametric tests mostly cannot be fulfilled, the use of non-parametric tests is proposed by Derrac, García, Molina, and Herrera (2011) and García, Molina, Lozano, and Herrera (2009). In this context, Mann-Whitney U Test is considered as an appropriate test and it is employed here to detect possible significant differences. The results for significance level $\alpha = 0.05$ are presented in Table 8, where the signs "∼" and "+" denote non-significant and significant differences, respectively. According to the results presented in Table 8, it can be concluded that eFA-rw is significantly better than eFA-seq in the instances with 30, 50, 100 and 200 peaks.

Another analysis is conducted to see the effects of the varying *shift length*. While a greater *shift length* yields to greater changes, less severe *shift length* values result in minor effects in peaks coordinates. It is clear that tracking the peaks that are exposed to more severe changes is more challenging.

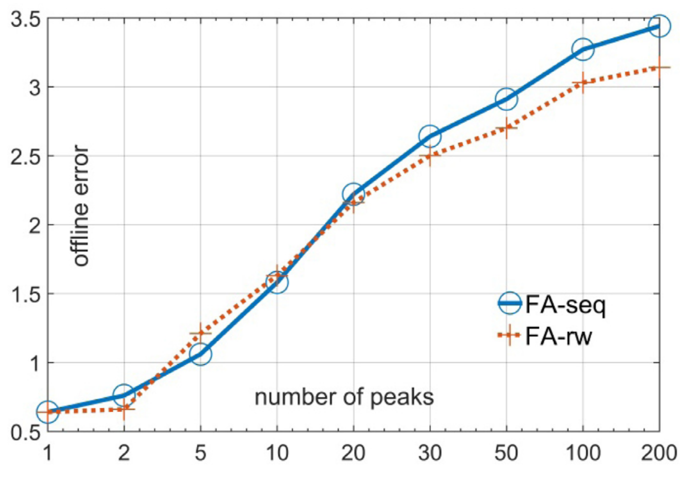

Fig. 3. The effects of varying number of peaks.

Table 8
Significant differences between eFA-rw and eFA-seq.

	number of peaks								
	1	2	5	10	20	30	50	100	200
eFA-rw vs. eFA-seq	∼	∼	∼	∼	∼	+	+	+	+

Table 9
The effects of varying values for shift length.

		shift length				
ref.	*algorithm*	1	2	3	4	5
Bird and Li (2007)	SPSO	2.51 ± 0.09	3.78 ± 0.09	4.96 ± 0.12	5.56 ± 0.13	6.76 ± 0.15
Bird Li (2007)	rSPSO	1.50 ± 0.08	1.87 ± 0.05	2.40 ± 0.08	2.90 ± 0.08	3.25 ± 0.09
Blackwell and Branke (2006)	10*1(10 + 0)	9.89 ± 0.73	10.83 ± 0.69	11.63 ± 0.64	11.73 ± 0.57	11.83 ± 0.62
Li et al. (2006)	$10(5+5^{+})$	2.05 ± 0.07	2.80 ± 0.07	3.57 ± 0.08	4.18 ± 0.09	4.89 ± 0.11
Li et al. (2006)	$10(5+5^{q})$	1.75 ± 0.06	2.40 ± 0.06	3.00 ± 0.06	3.59 ± 0.10	4.24 ± 0.10
Li et al. (2006)	10(10 + 0)	2.32 ± 0.06	3.37 ± 0.20	4.24 ± 0.09	5.08 ± 0.11	5.90 ± 0.12
Nasiri Meybodi (2012)	SFA	1.05 ± 0.08	1.44 ± 0.06	2.06 ± 0.07	–	2.89 ± 0.13
eFA-seq		1.58 ± 0.05	1.91 ± 0.03	2.27 ± 0.06	2.50 ± 0.06	2.57 ± 0.05
eFA-rw		1.63 ± 0.05	2.00 ± 0.06	2.30 ± 0.06	2.42 ± 0.07	2.60 ± 0.06

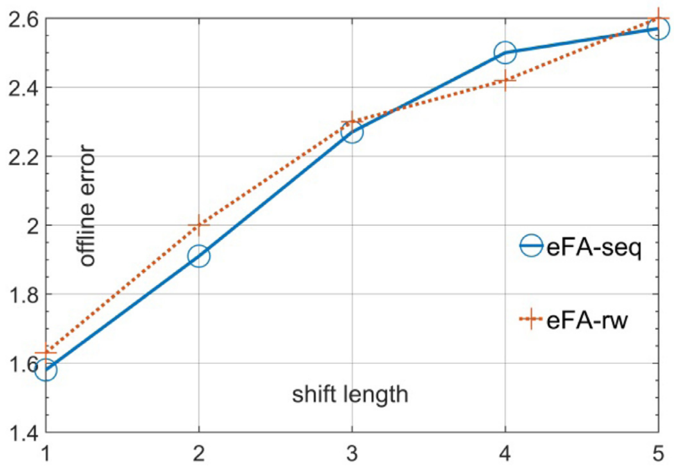

Fig. 4. The effects of varying values for shift length.

According to the results presented in Table 9 and illustrated in Fig. 4, where *frequency* and *number of peaks* (*M*) are fixed to 5000 and 10, respectively, the eFA-seq and eFA-rw achieve competitive results. However, it should be noted that in the previous analysis, eFA-seq and eFA-rw obtains similar results in the instance with 10 peaks. In parallel, a similar pattern is also apparent here. Finally, in comparison to the results of other well-known algorithms', the results obtained here can be considered as competitive.

It is clear that there are numerous sophisticated algorithms in the related literature. However, most of them ignore an important issue, which can be summarized as prioritizing the sub-swarms and analyzing these effects. The present study attempts to stress this approach by proposing several new modifications for FA, which is shown to be a promising optimizer in numerous types of optimization problems. Finally, according to the statistically verified results, it can be concluded that the proposed approaches here can be considered as promising. Obtained results point out that particularly eFA-rw has potential for more challenging multi-modal optimization problems. Although statistically verified results show the efficiency of the proposed approach, it is still open to possible improvements to compete with more sophisticated algorithms.

5. Conclusion

In the present work, a multi-population extension of Firefly Algorithm (FA) is proposed to solve multi-modal dynamic optimization problems. The proposed FA modification uses fitness function evaluations more efficiently. In order to maintain diversity and increasing solution quality, quantum particles, which implicitly perform a local search within quantum clouds, are employed. Moreover, two different sub-swarm selection strategies including sequential and roulette wheel-based selection are developed. The proposed algorithms are compared to the well-known methods reported in the related literature.

Comprehensive experimental study and statistical tests verify that roulette-based selection strategy performs better than sequence-based selection, particularly in the instances with greater number of peaks. Additionally, it's worth stressing that although the results are not the best published results thus far, the proposed approaches seem to be competitive in comparison to those well-known methods.

As demonstrated in the present contribution, prioritizing methodology of sub-swarms is an important factor that might significantly affect the performance of the used multi-population based algorithm. Although there are numerous sophisticated multi-population based techniques, unfortunately, sub-swarm selection mechanism is poorly discussed in the related literature. In this respect, along with several new modifications proposed for FA, it is expected that the present work contributes to the existing literature. Finally, performing further improvements on the algorithms reported here by using memory-based approaches is scheduled as a future work.

Authos contributions

Each author contributed equally to this paper and they do not have conflict of interest.

Acknowledgements

The present work is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant number: 214M240.

References

Altin, L., & Topcuoglu, H. R. (2018). Impact of sensor-based change detection schemes on the performance of evolutionary dynamic optimization techniques. *Soft Computing, 22*(14), 4741–4762.
Baykasoğlu, A., & Ozsoydan, F. B. (2014). An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Systems with Applications, 41*(8), 3712–3725.
Baykasoğlu, A., & Ozsoydan, F. B. (2017). Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization. *Information Sciences, 420*, 159–183.
Baykasoğlu, A., & Ozsoydan, F. B. (2018). Dynamic optimization in binary search spaces via weighted superposition attraction algorithm. *Expert Systems with Applications, 96*, 157–174.
Bird, S., & Li, X. (2006). Adaptively choosing niching parameters in a PSO. In *proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 1–3). doi:10.1145/1143997.1143999.
Bird, S., & Li, X. (2007). Using regression to improve local convergence. In *proceedings of IEEE Congress on Evolutionary Computation* (pp. 592–599). doi:10.1109/CEC.2007.4424524.
Blackwell, T., & Branke, J. (2004). Multi-swarm optimization in dynamic environments. In G. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, & Y. Jin, et al. (Eds.), *Applications of evolutionary computing* (pp. 489–500). Berlin: Springer Heidelberg.

Blackwell, T., & Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE transactions on evolutionary computation, 10*(4), 459–472.

Blackwell, T., Branke, J., & Li, X. (2008). Particle swarms for dynamic optimization problems. In C. Blum, & D. Merkle (Eds.), *Swarm intelligence* (pp. 193–217). Berlin: Springer Heidelberg.

Blackwell, T. M., & Bentley, P. J. (2002). Dynamic search with charged swarms. *In proceedings of the Genetic and Evolutionary Computation Conference, 2*, 19–26.

Blackwell, T. M. (2003). Swarms in dynamic environments. In E. Cantú-Paz, J. Foster, K. Deb, L. Davis, R. Roy, & U-M. O'Reilly, et al. (Eds.). In *Genetic and evolutionary computation: 2723* (pp. 1–12). Berlin: Springer Heidelberg. doi:10.1007/3-540-45105-6_1.

Boulesnane, A., & Meshoul, S. (2017). WD2O: A novel wind driven dynamic optimization approach with effective change detection. *Applied Intelligence, 47*(2), 488–504.

Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. In *proceedings of Evolutionary Computation* (pp. 1875–1882). doi:10.1109/CEC.1999.785502.

Branke, J. (2001). *Evolutionary optimization in dynamic environments*. Dordrecht: Kluwer Academic Publishers.

Branke, J., Kaußler, T., Smidt, C., & Schmeck, H. (2000). A multi-population approach to dynamic optimization problems. In I. C. Parmee IC (Ed.), *Evolutionary design and manufacture* (pp. 299–307). London: Springer.

Branke, J., & Schmeck, H. (2003). Designing evolutionary algorithms for dynamic optimization problems. In A. Ghosh, & S. Tsutsui (Eds.), *Advances in evolutionary computing* (pp. 239–262). Berlin: Springer, Heidelberg.

Cadenas, J. M., Garrido, M. C., & Munoz, E. (2011). Facing dynamic optimization using a cooperative metaheuristic configured via fuzzy logic and SVMs. *Applied Soft Computing, 11*(8), 5639–5651.

Calderín, J. F., Masegosa, A. D., & Pelta, D. A. (2015). Algorithm portfolio based scheme for dynamic optimization problems. *International Journal of Computational Intelligence Systems, 8*(4), 667–689.

Cao, L., Xu, L., & Goodman, E. D. (2018). A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems. *Information Sciences, 453*, 463–485.

Chen, H., Ma, L., He, M., Wang, X., Liang, X., & Sun, L. (2017). Artificial bee colony optimizer based on bee life-cycle for stationary and dynamic optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(2), 327–346.

Cobb, H. G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. *In NRL Memorandum Report, 6760*, 523–529.

Cruz, C., González, J. R., & Pelta, D. A. (2011). Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing, 15*(7), 1427–1448.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). 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), 3–18.

Du Plessis, M. C., & Engelbrecht, A. P. (2012). Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research, 218*(1), 7–20.

Du Plessis, M. C., & Engelbrecht, A. P. (2013). Differential evolution for dynamic environments with unknown numbers of optima. *Journal of Global Optimization, 55*(1), 73–99.

Du, W., & Li, B. (2008). Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences, 178*(15), 3096–3109.

Farahani, S., Nasiri, B., & Meybodi, M. (2011). A multiswarm based firefly algorithm in dynamic environments. In *Proceedings of the Third International Conference on Signal Processing Systems* (pp. 68–72).

Fister, I., Fister Jr, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation, 13*, 34–46.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non–parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics, 15*(6), 617.

García-Villoria, A., & Pastor, R. (2009). Introducing dynamic diversity into a discrete particle swarm optimization. *Computers & Operations Research, 36*(3), 951–966.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms and their applications* (pp. 41–49).

Goldberg, D. E., & Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *proceedings of International Conference on Genetic Algorithms* (pp. 59–68).

Grefenstette, J. J. (1992). Genetic algorithms for changing environments. In *proceedings of Parallel Problem Solving from Nature* (pp. 137–144).

Hadad, B., & Eick, C. (1997). Supporting polyploidy in genetic algorithms using dominance vectors. In P. Angeline, R. Reynolds, J. McDonnell, & R. Eberhart (Eds.), *Evolutionary programming VI* (pp. 223–234). Berlin: Springer Heidelberg.

Hashemi, A. B., & Meybodi, M. R. (2009). Cellular PSO: A PSO for dynamic environments. In Z. Cai, Z. Li, Z. Kang, & Y. Liu (Eds.), *Advances in computation and intelligence* (pp. 422–433). Berlin: Springer Heidelberg, Berlin.

Hu, X., & Eberhart, R. C. (2002). Adaptive particle swarm optimization: Detection and response to dynamic systems. In *proceedings of congress on the Evolutionary Computation* (pp. 1666–1670). doi:10.1109/CEC.2002.1004492.

Huang, V. L., Qin, A. K., & Suganthan, P. N. (2006). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *proceedings of IEEE Congress on Evolutionary Computation* (pp. 215–222). doi:10.1109/CEC.2006.1688285.

Janson, S., & Middendorf, M. (2004). A hierarchical particle swarm optimizer for dynamic optimization problems. In G. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, & Y. Jin, et al. (Eds.), *Applications of evolutionary computing* (pp. 513–524). Berlin: Springer Heidelberg.

Kamosi, M., Hashemi, A. B., & Meybodi, M. R. (2010a). A new particle swarm optimization algorithm for dynamic environments. In B. Panigrahi, S. Das, P. Suganthan, & S. Dash (Eds.), *Swarm, evolutionary, and memetic computing* (pp. 129–138). Berlib: Springer Heidelberg.

Karaman, A., Uyar, Ş., & Eryiğit, G. (2005). The memory indexing evolutionary algorithm for dynamic environments. In F. Rothlauf, J. Branke, S. Cagnoni, D. Corne, R. Drechsler, & Y. Jin, et al. (Eds.), *Applications of evolutionary computing* (pp. 563–573). Berlin: Springer Heidelberg.

Kamosi, M., Hashemi, A. B., & Meybodi, M. R. (2010b). A hibernating multi-swarm optimization algorithm for dynamic environments. In *proceedings of Second World Congress on Nature and Biologically Inspired Computing* (pp. 363–369). doi:10.1109/NABIC.2010.5716372.

Karimi, J., Nobahari, H., & Pourtakdoust, S. H. (2012). A new hybrid approach for dynamic continuous optimization problems. *Applied Soft Computing, 12*(3), 1158–1167.

Kordestani, J. K., Firouzjaee, H. A., & Meybodi, M. R. (2018). An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems. *Applied Intelligenc, 48*(1), 97–117.

Kordestani, J. K., Rezvanian, A., & Meybodi, M. R. (2014). CDEPSO: A bi-population hybrid approach for dynamic optimization problems. *Applied intelligence, 40*(4), 682–694.

Krishnakumar, K. (1990). Micro-genetic algorithms for stationary and non-stationary function optimization. In *proceedings of SPIE 1196 International Conference on Intelligent Control and Adaptive Systems* (pp. 289–296). doi:10.1117/12.969927.

Lepagnot, J., Nakib, A., Oulhadj, H., & Siarry, P. (2013). A multiple local search algorithm for continuous dynamic optimization. *Journal of Heuristics, 19*(1), 35–76.

Li, C., & Yang, S. (2008). Fast multi-swarm optimization for dynamic optimization problems. In *proceedings of the fourth International Conference on Natural Computation* (pp. 624–628). doi:10.1109/ICNC.2008.313.

Li, C., & Yang, S. (2009). A clustering particle swarm optimizer for dynamic optimization. In *proceedings of the IEEE Congress on Evolutionary Computation* (pp. 439–446). doi:10.1109/CEC.2009.4982979.

Li, C., Yang, S., & Yang, M. (2014). An adaptive multi-swarm optimizer for dynamic optimization problems. *Evolutionary computation, 22*(4), 559–594.

Li, X. (2004). Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In K. Deb (Ed.), *Genetic and evolutionary computation* (pp. 105–116). Berlin: Springer Heidelberg, Berlin.

Li, X., Branke, J., & Blackwell, T. (2006). Particle swarm with speciation and adaptation in a dynamic environment. In *proceedings of the 8th annual Conference on Genetic and Evolutionary Computation* (pp. 51–58). doi:10.1145/1143997.1144005.

Ma, H., Shen, S., Yu, M., Yang, Z., Fei, M., Zhou, H., Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey, *Swarm and Evolutionary Computation*, in press.

Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation, 33*, 1–17.

Mendes, R., & Mohais, A. (2005). DynDE: A differential evolution for dynamic optimization problems. In *proceedings of IEEE Congress on Evolutionary Computation* (pp. 2808–2815). doi:10.1109/CEC.2005.1555047.

Morrison, R. W. (2004). *Designing evolutionary algorithms for dynamic environments*. Berlin: Springer-Verlag.

Mukherjee, R., Patra, G. R., Kundu, R., & Das, S. (2014). Cluster-based differential evolution with crowding archive for niching in dynamic environments. *Information Sciences, 267*, 58–82.

Nasiri, B., & Meybodi, M. (2012). Speciation based firefly algorithm for optimization in dynamic environments. *International Journal of Artificial Intelligence, 8*(12), 118–132.

Nasiri, B., Meybodi, M., & Ebadzadeh, M. (2016). History-Driven Particle Swarm Optimization in dynamic and uncertain environments. *Neurocomputing, 172*, 356–370.

Noroozi, V., Hashemi, A., & Meybodi, M. (2011). CellularDE: A cellular based differential evolution for dynamic optimization problems. In A. Dobnikar, U. Lotrič, & B. Šter (Eds.), *Adaptive and natural computing algorithms* (pp. 340–349). Berlin: Springer Heidelberg.

Nseef, S. K., Abdullah, S., Turky, A., & Kendall, G. (2016). An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems. *Knowledge-Based Systems, 104*, 14–23.

Ozsoydan, F. B., & Baykasoglu, A. (2015). A multi-population firefly algorithm for dynamic optimization problems. In *proceedings of IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)* (pp. 27–33).

Parrott, D., & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation, 10*(4), 440–458.

Pelta, D., Cruz, C., & Verdegay, J. L. (2009). Simple control rules in a cooperative system for dynamic optimisation problems. *International Journal of General Systems, 38*(7), 701–717.

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *proceedings of IEEE International Conference on Evolutionary Computation* (pp. 798–803). doi:10.1109/ICEC.1996.542703.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation, 13*(2), 398–417.

Richter, H. (2010). Memory design for constrained dynamic optimization problems. In C. Di Chio, S. Cagnon, C. Cotta, M. Ebner, A. Ekárt, & A. Esparcia-Alcazar, et al. (Eds.), *Applications of evolutionary computation* (pp. 552–561). Berlin: Springer Heidelberg.

Richter, H., & Yang, S. (2008). Memory based on abstraction for dynamic fitness functions. In M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, R. Drechsler, & A. Ekárt, et al. (Eds.), *Applications of evolutionary computing* (pp. 596–605). Berlin: Springer Heidelberg.

Richter, H., & Yang, S. (2009). Learning behavior in abstract memory schemes for dynamic optimization problems. *Soft Computing, 13*(12), 1163–1173.

Sulaiman, M. H., Mustafa, M. W., Azmi, A., Aliman, O., & Rahim, S. A. (2012). Optimal allocation and sizing of distributed generation in distribution system via firefly algorithm. In *proceedings of the IEEE International Conference on Power Engineering and Optimization Conference (PEDCO)* (pp. 84–89).

Topcuoglu, H. R., Ucar, A., & Altin, L. (2014). A hyper-heuristic based framework for dynamic optimization problems. *Applied Soft Computing, 19*, 236–251.

Tinós, R., & Yang, S. (2007). A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines, 8*(3), 255–286.

Tsutsui, S, Fujimoto, Y, & Ghosh, A (1997). Forking genetic algorithms: GAs with search space division schemes. *Evolut. Comput., 5*(1), 61–80. doi:10.1162/evco.1997.5.1.61.

Turky, A. M., & Abdullah, S. (2014a). A multi-population electromagnetic algorithm for dynamic optimisation problems. *Applied Soft Computing, 22*, 474–482.

Turky, A. M., & Abdullah, S. (2014b). A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Information Sciences, 272*, 84–95.

Turky, A., Abdullah, S., & Dawod, A. (2018). A dual-population multi operators harmony search algorithm for dynamic optimization problems. *Computers & Industrial Engineering, 117*, 19–28.

Ursem, R. K. (1999). Multinational evolutionary algorithms. In *proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1633–1640). doi:10.1109/CEC.1999.785470.

Uyar, A. Ş., & Harmanci, A. E. (2005). A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing, 9*(11), 803–814.

Wang, H., Wang, D., & Yang, S. (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing, 13*(8-9), 763–780.

Woldesenbet, Y. G., & Yen, G. G. (2009). Dynamic evolutionary algorithm with variable relocation. *IEEE Transactions on Evolutionary Computation, 13*(3), 500–513.

Yang, S. (2006a). Associative memory scheme for genetic algorithms in dynamic environments. In F. Rothlauf, J. Branke, F. Cagnoni, E. Costa, C. Cotta, & R. Drechsler, et al. (Eds.), *Applications of evolutionary computing* (pp. 788–799). Berlin: Springer Heidelberg.

Yang, S. (2006b). On the design of diploid genetic algorithms for problem optimization in dynamic environments. In *proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1362–1369).

Yang, S. (2008). Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evolutionary Computation, 16*(3), 385–416.

Yang, S., Cheng, H., & Wang, F. (2010). Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40*(1), 52–63.

Yang, S., & Li, C. (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation, 14*(6), 959–974.

Yang, S., & Yao, X. (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing, 9*(11), 815–834.

Yang, X-S. (2009). Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms* (pp. 169–178). doi:10.1007/978-3-642-04944-6_14.

Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., & Meybodi, M. R. (2013). A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing, 13*(4), 2144–2158.

Yu, E. L., & Suganthan, P. N. (2009). Evolutionary programming with ensemble of explicit memories for dynamic optimization. In *Proceedings of IEEE Congress on Evolutionary Computation* (pp. 431–438). doi:10.1109/CEC.2009.4982978.