Received: 21 February 2017 | Revised: 22 December 2017 | Accepted: 16 March 2018

DOI: 10.1111/exsy.12286

ORIGINAL ARTICLE

WILEY Expert Systems

Fuzzy adaptive cat swarm algorithm and Borda method for solving dynamic multi-objective problems

Maysam Orouskhani | **Daming Shi**

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

Correspondence
Daming Shi, College of Computer Science and Software Engineering, Shenzhen University, 3688 Nanhai Ave, Nanshan Qu, Shenzhen Shi, Guangdong Sheng, China 518060.
Email: dshi@szu.edu.cn

Funding information
Shenzhen Science and Technology Innovation Commission (SZSTI), Grant/Award Number: JCYJ20170302153752613

Abstract

The main goal of this paper is to introduce a novel dynamic multi-objective optimization algorithm. First, after detecting the environmental changes, Borda count ranking method is applied to population in order to assign the Borda score to each individual, and then the lowest score individuals are removed from population and replaced with new created solutions. Furthermore, fuzzy adaptive multi-objective cat swarm optimization algorithm is used to estimate the Pareto-optimal front in which its parameters are tuned to new environment by Mamdani fuzzy rules when a change occurs. Performance of the proposed algorithm is tested on dynamic multi-objective benchmarks and is compared with recent achievements. The simulations show the quite satisfactory results and higher performance of the proposed method in comparison with traditional approaches.

KEYWORDS

Borda method, dynamic multi-objective algorithm, fuzzy CSO

1 | INTRODUCTION

Some multi-objective optimization problems exhibit dynamic behaviour in objective functions, constraints, or in the number of variables and objectives. In this case, the optimal solutions are likely to change over time, and the main goal is to approximate and track the time-varying solutions. Problems with such characteristics are known as dynamic multi-objective optimization problems (DMOOPs; Helbig & Engelbrecht, 2013). As examples of DMOOP in real-world engineering optimization, these two problems can be mentioned: regulation of a lake–river system in order to control the changing rate of inflow and outflow of water and keep at the ideal rate (Mantysaari & Hamalainen, 2001), and heating DMOOP, which the main goal is to keep the time-varying indoor temperature within comfortable limits based on three objectives that includes heating consumption, cost of heating, and distance from ideal temperature as dynamic objective (Mantysaari & Hamalainen, 2002).

How to response to change after environmental changes is the main issue addressed in this study. Many authors developed two most used strategies for handling this issue in population-based algorithms: predictive approaches and re-initialization methods. Algorithms based on prediction models usually use historical information of previous steps to estimate the solutions or population of next generation. These algorithms assume that the solutions of new environment (the environment after change) are depended to the solutions of previous environments and try to find nonlinear relations between them. In this case, great time and space complexity is unavoidable, and it is acceptable when the volume of change is as small as possible. It is likely that when a great change occurs, the obtained knowledge of previous environments is not applicable to determine the new solutions. For example, Muruganantham, Zhao, Gee, and Qiu (2013) proposed a hybrid approach of an evolutionary algorithm and Kalman filter that uses historical information to predict for future generations. Also, Liu introduced two predictive models: The first is an orthogonal model that estimates the new individuals after the change (Liu, Nui, Fan, Mu, & Jiao, 2014), whereas the second is a modified prediction model using the historical information of last two times to initialize the new population (Liu, Fan, & Jiao, 2015).

On the other side, re-initialization methods re-initialize some percentage of population when a change occurs. So the time and complexity is as low as possible, and there is no nonlinear estimation to predict the new population. Here, at the time of change, algorithm selects some individuals randomly and replaces with new solutions. For example, Deb, Rao, and Karthik (2007) introduced a dynamic version of NSGAII that re-initializes

30% of population by new or mutated solutions to increase the diversity rate. Although this method has lower complexity than former, the main drawback is random selection of solutions that should be re-initialized that may mislead the trajectory of algorithm. Also, this random selection may pick and remove the elite solutions from population, which may deteriorate the rate of convergence. This drawback is the main issue addressed in this paper.

The main contribution of this paper is to use a wise strategy (instead of random selection) to select the individuals that should be re-initialized and apply a simple fuzzy model (instead of fix values) to adapt the algorithm to new environment. Borda count is an optimal rank aggregation method that determines the best candidates from some lists. After calculating the Borda score of each individual, solutions with the lowest Borda score are replaced by new ones. The new improved population is then given to optimization phase where it is responsible of producing the diverse solutions, which is done by cat swarm algorithm (CSO). Due to simultaneous local and global search ability, CSO has the better convergence and performance in comparison with other traditional population-based algorithms such as particle swarm optimization (PSO; Chu, Tsai, & Pan, 2006). Moreover, one of the main issues in dynamic environment is existence of optimal solutions in the search space. In addition, when a change occurred, the optimization algorithm should be able to adapt itself to new environment. In this paper, this adaption is done by fuzzy rules made of change volume and previous performance. The fuzzy model used in this paper included four inputs, two outputs, and three membership functions for each input and output.

The rest of paper is written as follows: Section 2 presents background information of dynamic multi-objective optimization (DMOO), whereas the proposed algorithm is presented in Section 3. Information about the experiments including the parameters of the CSO, benchmark functions, and performance metrics and simulations results is provided in Section 4. Finally, conclusion about this research is presented in the last section.

2 | DYNAMIC MULTI-OBJECTIVE OPTIMIZATION

2.1 | Static multi-objective problem

Solving multi-objective problems requires simultaneous optimization of vector of objectives that are usually in conflict with another. So the main goal is to find a set of compromised solutions called non-dominated set or Pareto-optimal set (POS) and the corresponding values in the objective space named Pareto-optimal front (POF). Given the objective vector *f*, Pareto-optimal solution set, *P*, and *O* refers to the objective space, and the POF $PF^* \subseteq O$ is defined as follows (Helbig & Engelbrecht, 2013):

$$PF^* = \{f = (f_1(x^*), f_2(x^*), ..., f_k(x^*)) \mid x^* \in P\}. \tag{1}$$

Therefore, the POF contains the set of objective vectors that corresponds to the POS, that is, the set of decision vectors that are non-dominated.

2.2 | Dynamic multi-objective problem

Contrary to static multi-objective problems, a DMOOP has at least one time-varying objective function to be optimized simultaneously. In other words, DMOOP is as an extension of the static multi-objective problem with an additional time parameter (Farina, Deb, & Amato, 2004). Let the n_x-dimensional decision space be represented by $S \subseteq R^{n_x}$ and the feasible space represented by $F \subseteq S$, where $F = S$ for boundary constrained optimization problems. Let $x = (x_1, x_2, ..., x_{n_x}) \in S$ represent a decision vector, and let a single objective function be defined as $f_k : R^{n_x} \rightarrow R$. Let an objective vector containing n_k objective function evaluations be represented by $(x) = (f_1(x), f_2(x), ..., f_{n_k}(x)) \in O \subseteq R^{n_k}$, with *O* representing the objective space. Then a boundary constrained DMOOP is defined as

$$\text{Minimize } F(X, W(t)), \quad W(t) = (w_1(t), ..., w_N(t)) \text{ subject to } g_i(x) \le 0, \ i = 1, ..., n_g h_j(x) = 0, \ j = ng + 1, ..., n_h x \in [x_{\min}, x_{\max}]^{n_x}, \tag{2}$$

where *W*(*t*) is a vector of time-dependent control parameters of an objective function at time (*t*), *F*(*X*) is the vector of objective functions, n_x is the number of decision variables, $x \in R^{n_x}$, n_g is the number of inequality constraints, *g*, n_h is the number of equality constraints, *h*, and $x \in [x_{\min}, x_{\max}]^{n_x}$ refers to the boundary constraints. Unlike dynamic single optimization problem with only one objective function, DMOOP has many objective functions changing over time. Therefore, in order to solve the DMOOP, the goal is to track the changing POF. In this case, the definition of the dynamic POF is written as follows:

$$PF^* = \left\{ f(t) = \left(\left(f_1\left(x^*, w_1(t)\right), \left(f_2\left(x^*, w_2(t)\right), ..., \left(f_N\left(x^*, w_N(t)\right)\right) \right| x^* \in P^* \right\}. \tag{3}$$

Therefore, it is clear that the position, shape (convexity or concavity), and continuity of dynamic POF may be changed over time and this is the main difference between static POF and dynamic POF. It means that some non-dominated solutions may be dominated by others when an environmental change occurs.

2.3 | Types of dynamic environments

This section discusses the categorization of DMOOPs, as well as the possible influences of a change in the environment on the POF. Four types of dynamic environment are as follows: (a) The optimal set of decision variables (POS) changes, but the corresponding objective function values (POF) remain unchanged; (b) both the POS and the POF change; (c) the POS remains unchanged, but the POF changes; and (d) both the POS and the POF remain unchanged. Also when a change occurs in the environment, the POF can change as one the following way over time:

- Existing solutions in the POF become dominated.
- The shape of the POF remains the same, but its location changes over time.
- The shape of the POF changes over time.

2.4 | Review on dynamic multi-objective algorithms

This section investigates the traditional approaches of solving DMOOPs as follows.

Some static multi-objective optimization algorithms are adapted for solving dynamic problems such as multi-objective genetic algorithm (MOGA) that was adapted for DMOO (Avdagic, Konijicija, & Omanovic, 2009). MOGA was the first multi-objective genetic algorithm that uses Pareto-ranking. The main advantage of MOGA is a simple fitness assignment and the easy application of MOGA to other types of optimization problems. Also, a new version of NSGA-II was adapted for DMOO (Deb et al., 2007). In this algorithm, few solutions are randomly selected and re-evaluated, and if there is any change in the objective values, it is assumed that a change in the environment has occurred. So to increase the rate of diversity, some particles are replaced by randomly created individuals (DNSGAII-A) or mutated solutions (DNSGA$_{II}$-B; Deb et al., 2007).

Some DMOO algorithms use swarm intelligence optimization such as DMOO using PSO that investigates the effect of various approaches to manage boundary constraint violations on the performance of the dynamic vector evaluated PSO (DVEPSO; Helbig & Engelbrecht, 2013). The algorithm uses a cooperative PSO and includes sub-swarms where each swarm optimizes only one objective function. Knowledge of the best solutions is then shared with the other swarms. The algorithm shows that deflection and periodic boundary handling approaches lead to bad performance whereas random and re-initialization are the best. In addition, same authors proposed the Heterogeneous version of DVEPSO (Helbig & Engelbrecht, 2014) that each particle has a different behaviour and is randomly selected from a behaviour pool. These various behaviours may lead to a mixture of particles with a more exploring behaviour and particles with a more exploiting behaviour as follows: standard, social, cognitive, and Bare-bones. The proposed method outperformed the default DVEPSO with regard to accuracy and converged much better to discontinuous POFs than DVEPSO.

Some authors proposed the algorithms that convert a dynamic multi-objective problem into various static multi-objective optimization problems. In these methods, the time period of the DMOOP is divided into several smaller time intervals. For each time interval, a statics optimization problem is defined by limiting the DMOOP to the specific time interval. Each static problem is then transformed into a new two-objective optimization problem where one objective is to increase the diversity of the solutions and the other objective is to increase the quality of the found non-dominated solutions (Dang & Wang, 2008).

Various contributions were made to improve dynamic multi-objective evolutionary algorithms. For example, four re-initialization strategies that can be applied to any dynamic multi-objective evolutionary algorithms when a change in the environment is detected were proposed by Zhou, Zhang, Sendhoff, and Tsang (2007) as follows: randomly re-initialize all new solutions (RND), add Gaussian noise (VAR), sample solutions around predicted locations of the POS (PRE), and create half of the solutions using PRE and the other half using VAR (V&P). The simulations show that RND did not perform well, because all previous knowledge were removed by re-initializing all individuals, whereas V&P can be a better selection. Also, Li and Wang (2009) proposed four memory-based strategies to reuse past knowledge of the found Pareto-optimal solutions (POS) when a change occurs: restart scheme, explicit scheme, local search scheme, and hybrid scheme.

Prediction-based approaches have been proposed to decrease the number of function evaluations without decreasing the quality of found solutions. These methods incorporate a forecasting technique into an evolutionary algorithm to make rapid response and predict the population or solutions of next generation. Recently, Liu introduced two predictive models as follows: The first (PDNNIA; Liu et al., 2015) is an orthogonal predictive model that estimates the new individuals after the change occurred, whereas the second is a modified prediction model using the information of last two times to initialize the new population (Liu et al., 2014). The main drawback of the latter is to investigate the knowledge of just two previous times to estimate the new generation.

2.5 | Cat swarm algorithm

CSO is a swarm-based algorithm that models the behaviour of cats into “Seeking mode” and “Tracing mode.” These two modes are combined by mixture ratio, which decides how many cats will be moved into seeking mode process and tracing mode (Chu et al., 2006). CSO can be described as follows: First, the population of cats is created, initialized, and evaluated. Then, according to the rate of mixture ratio, cats are moved to seeking or tracing mode, and these two modes are processed. Then the new obtained cats compose the refresh population. This process continues until the last iteration.

2.5.1 | Seeking mode

For modelling the behaviour of cats in resting time and being alert, we use the seeking mode. This mode is a time for thinking and deciding about next move. This mode has four main parameters: seeking memory pool, seeking range of the selected dimension (SRD), counts of dimension to change (CDC), and self-position consideration. The process of seeking mode is described as follows: First, suppose that we have k cats. So make j copies of the present position of Cat_k, where $j = SMP$. If the value of self-position consideration is true, $j = (SMP - 1)$, then retain the present position as one of the candidates. Then, for each copy, according to CDC, randomly plus or minus SRD per cent to the present values and replace the old ones. Calculate the fitness values (FS) of all candidate points, and then, if all FS are not exactly equal, calculate the selecting probability of each candidate point (P) by Equation (4). Finally, randomly pick the point to move from the candidate points, and replace the position of Cat_k. In the minimization problems, $FS_b = FS_{\max}$, otherwise $FS_b = FS_{\min}$.

$$P_i = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}}. \tag{4}$$

2.5.2 | Tracing mode

In this mode, cats desire to trace targets and foods. The process of tracing mode can be described as follows: update the velocities for every dimension according to ((5)). Then update the position of Cat_k according to ((6)) where k is index of cat, d is related to dimension, $V_{k,\,d}$ is the velocity of Cat_k in dimension of d, $X_{k,\,d}$ is the position of Cat_k in dimension of d, $X_{best,\,d}$ is the position of the cat, who has the best fitness value, C_1 is an acceleration coefficient equal to 2.05, and r_1 is a random value in the range of [0, 1]. The value of each parameter used in this study is as the same as original CSO.

$$V_{k,d} = V_{k,d} + r_1 c_1 \left(X_{best,d} - X_{k,d} \right). \tag{5}$$

$$X_{k,d} = X_{k,d} + V_{k,d}. \tag{6}$$

3 | THE PROPOSED ALGORITHM

The first step of solving a dynamic multi-objective problem is to detect an environmental change, which is done by sentry particles in this study (Helbig & Engelbrecht, 2013). After change detection, the main issue is how to response to the change. To do this, this paper uses two phases: Borda count ranking method and fuzzy logic. In order to increase the diversity of solutions after change, the proposed method selects some percentage of population and replaces with new solutions. In contrast to previous methods, this selection process is done by a wisely method called Borda count ranking method. Borda method gives a score to each particle of population just before and after change and sorts the particles based on their scores. Finally, particles with the worst Borda score are the best candidates to be removed from generation and replaced with new solutions. This scenario for the first time was proposed by the same author (Orouskhani, Teshnehlab, & Nekoui, 2017).

In addition to previous work, this paper uses an extra phase to determine the parameters of optimization algorithm (CSO) in new environment, which is done by Mamdani fuzzy logic. If the optimization parameters are adjusted based on the volume of change, better performance in the new environment is obtainable. For this purpose, the knowledge obtained from population just before and after change is gathered to create the fuzzy rules. Here, two most important factors include the volume of change and performance of the optimization algorithm just before a change. For example, if the volume of change is small, it can be concluded that by using the same parameters, the same performance will be obtained in new environment.

Contribution of the work can be summarized as follows: in order to introduce a DMOOP, two steps should be done: response to change by removing the worst cats by applying Borda count method and adding some new cats to increase the diversity of population, and estimating the optimal Pareto front using fuzzy CSO.

3.1 | Applying Borda count

In a voting system, voters rank the candidates in order of relative preference, and the winner can be determined by different approaches. One of the most used is Borda count ranking method, invented in 1770 by the French mathematician and physicist Jean-Charles, chevalier de Borda (Borda, 1781). Borda is a voting method based on ranks, that is, it assigns a weight corresponding to the ranks in which a candidate appears within each voter's ranked list. Computationally, they are very easy, as they can be implemented in linear time. The Borda count method has been considered in the context of the rank fusion problem and works as follows: Each voter ranks a fixed set of c candidates in order of preference. For each voter, the top ranked candidate is given c points; the second ranked candidate is given $c - 1$ points, and so on. If there are some candidates left unranked by the voter (this is a default definition of Borda count method and will not happen in assigning the rank for particles of population), the remaining points are divided evenly among the unranked candidates. The candidates are ranked in decreasing order of total points.

Given full ordered lists: τ1, …, τL, each a permutation of the underlying space T, we let $R_{\tau l}$ (u) be the rank of element $u \in T$ in list τl. We let $B_l(u)$ denote the Borda's score in general, with $B_l(u) = R_{\tau l}$ being a special case. Let $Bu = f\ (B_1(u), B_2(u), \ldots, B_l(u))$ be an aggregate function of the Borda scores. Then one sorts the Bu's to obtain an aggregate ranked list τ (T). Frequently suggested aggregation functions are included of median and P-norm as follows (Shili, 2010):

$$f(x_1, \ldots, x_l) = median\{|x_1|, \ldots, |x_L|\}. \tag{7}$$

$$f(x_1, \ldots, x_l) = \sum_{l=1}^{L} \frac{|x_l|^P}{L}. \tag{8}$$

In the proposed method, when a change occurred, the fitness value of each particle just before and after change is calculated and stored in two separate lists (A, B). Then, rank of each particle is calculated by non-dominated sorting and crowding distance method. Now, Borda method gives a score to each particle based on its rank obtained by previous step. This scenario is done on list A, and B. Then, by using a linear function as aggregating function (f) of list A and B, the final Borda score for particles of population is determined and sorted in decreasing order. Finally, particles with the lowest Borda score are selected to be removed from population and re-initialized by new created solutions.

Figure 1 shows the selection process by Borda count. First of all, ranks and then Borda scores of all cats just before the change are calculated (A, A′). As the same way, when a change occurred, rank and score of each cat is calculated by new environment (B, B′). Then, by aggregating the obtained before and after change score of each cat, the average score is calculated (C). Finally, scores are sorted from the best to the worst rank (D). Figure 1 shows that Cat_5, Cat_{22}, and Cat_1 have the best score whereas the grey region includes the cats with the worst rank and should be removed from population and replaced with new ones.

3.2 | Fuzzy adaptive CSO

In this research in order to produce diverse solutions in the search space, multi-objective version of CSO is used. To get this, non-dominated sorting method is applied to obtain the solutions as close as to POF, and the crowding distance approach is used to keep and find solutions in less crowded parts of Pareto front. Moreover, it is clear that when a change occurred, the optimization algorithm should be able to increase the diversity of current solutions by concentrating on global search ability. This can be obtained by increasing the number of cats involving in seeking mode. Therefore, the optimization algorithm should be able to adapt its parameters to new environment that is traditionally done by linear or nonlinear formulas. In contrast to general methods, this paper uses fuzzy logic to determine the parameters of optimization algorithm where the obtained knowledge of particles is shared. The main reason of using fuzzy logic is to involve the obtained performance of all particles and value of the change in adapting the algorithm to new environment.

The fuzzy rules used in this section (shown in Figure 2) include four input variables and two outputs where each has three membership functions: low, medium, and high. The main issue in this section is to investigate the size of change as one of the inputs of fuzzy rules. Actually, the main goal of making parameters adaptive in static search space is to move from global to local search over iterations, whereas considering the change between two consecutive environments is the most important factor to determine the value of parameters. This is why this paper uses fuzzy logic to investigate the volume of change to determine the parameters of CSO. It means that it is likely that by using the same parameters for two different environments, the results are not equal. In the proposed method, the CDC and SRD values of after change are determined by

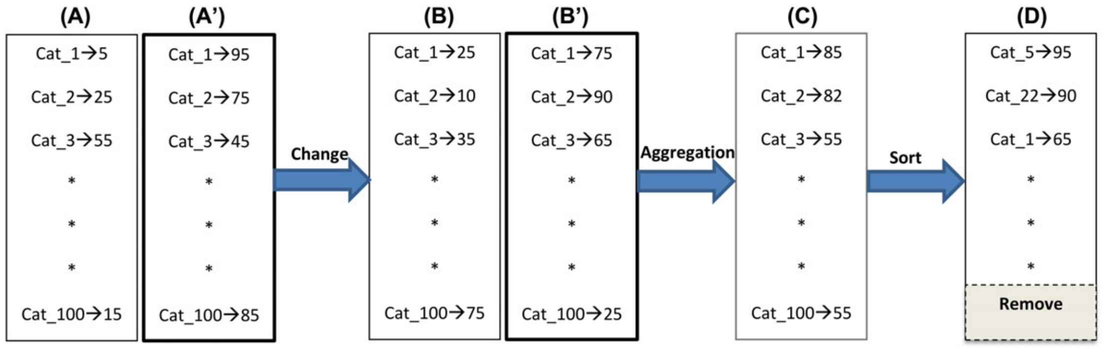

FIGURE 1 Ranks before change (A), scores before change (A′) // ranks after change (B), scores after change (B′), population with aggregated Borda scores (C) // population sorted by Borda score. Grey region includes the cats with the worst Borda score and should be removed (D)

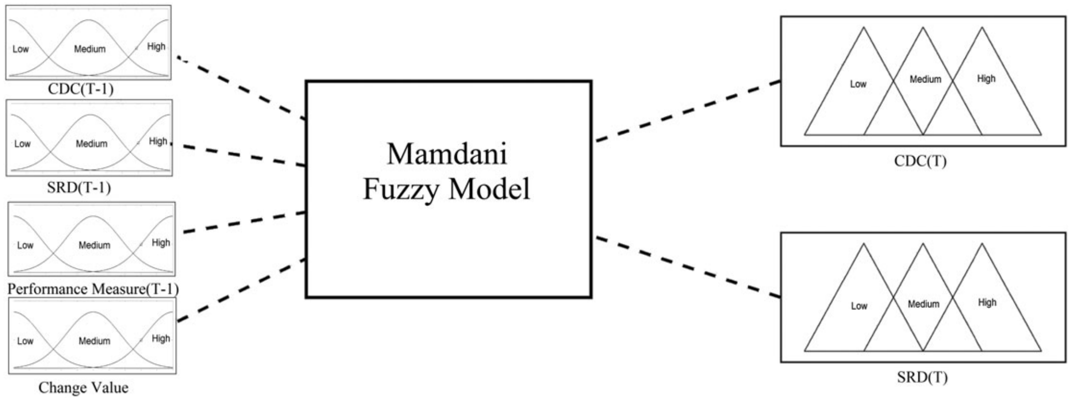

FIGURE 2 Mamdani fuzzy model includes four inputs and three membership functions to determine the counts of dimension to change (CDC) and seeking range of the selected dimension (SRD) of cat swarm algorithm when a change occurs

Mamdani fuzzy rules where CDC, SRD, and Hyper Volume Ratio (HVR) values of just before change and change size are investigated as inputs. Here, three membership functions of low, medium, and high are considered for each input and output variable, and 19 fuzzy rules are defined. For example, the last rule indicates that if the obtained performance of just before change is high, and the size of change is low (medium), it is concluded that the previous CDC and SRD is the good option for new environment and likely has the same performance. Due to page limitation, only three rules are indicated as follows:

1. If the size of change is *high*, then new CDC and SRD are *high*.
2. If the size of change is *low or medium*, and performance measure (HVR) of just before change is *low or medium*, and CDC and SRD of just before change is *low*, then the new CDC and SRD is not *low*.
3. If the size of change is *low or medium*, and performance measure (HVR) of just before change is *high*, and just before change CDC is *high* and just before change SRD is *high*, then the new CDC is *high* and new SRD is *high*.

3.3 | Calculating the volume of change

In this section, size of the environmental changes is calculated. As mentioned in fuzzy rules, change is defined by three membership functions: low, medium, and high. Because it is necessary to determine the upper and lower bound of each variable in fuzzy rule, a normalization formula is proposed to measure the volume of change as

$$\text{Volume of Change} = \frac{\left|\overline{X}_{Old} - \overline{X}_{New}\right|}{\overline{X}_{Old} + \overline{X}_{New}}, \tag{9}$$

where $\overline{X}_{Old}$ and $\overline{X}_{New}$ are the average fitness values just before and after change, respectively. The volume of change is performed on each fitness function, and finally, the average of volume of change for all functions is calculated and is given to fuzzy system as size of change. The flowchart of the proposed method is shown in Figure 3.

4 | SIMULATION RESULTS

All experiments consisted of 20 independent cycles, and each cycle consisted of 1,000 iterations. For all benchmark functions, the severity of change (n_t) is set to 5 and 10, and the frequency of change (τ_t) is set to either 15 or 25. Population size is set to 100, and the CSO parameters are set to *SMP* = 4, *CDC* = 0.8, *SRD* = 0.2, *MR* = 0.95 (95 cats in seeking and just 5 cats in tracing mode), and c_1 = 2. Here, 10 cats play the role of sentry cats for detecting the environmental changes, and 20 cats are selected by Borda count method to be re-initialized after a change occurred. Also, codes are implemented by Java Eclipse and in a PC that has an Intel Processor of 2.7 GHz, Quad core CPU, and 12 GB of Memory.

Table 1 mentions the performance measures used in this study to quantify the performance of the proposed algorithm as follows: "Variable generational distance" (VGD) measures the distance between POF* (approximated POF) and true POF where n is the number of found non-dominated solutions at time t and d_i is the Euclidean distance between i-th solution of POF* and the nearest solution of POF, "Spacing" measures how evenly the non-dominated solutions are distributed along the POF* where n_c is the number of changes, "Maximum Spread" (MS) measures how well POF* covers the POF where M is the number of objectives, and "Stability" measures the effect of the changes in the environment on the accuracy of the algorithm where HV computes the size of the region that is dominated by a set of non-dominated solutions, based on a reference vector.

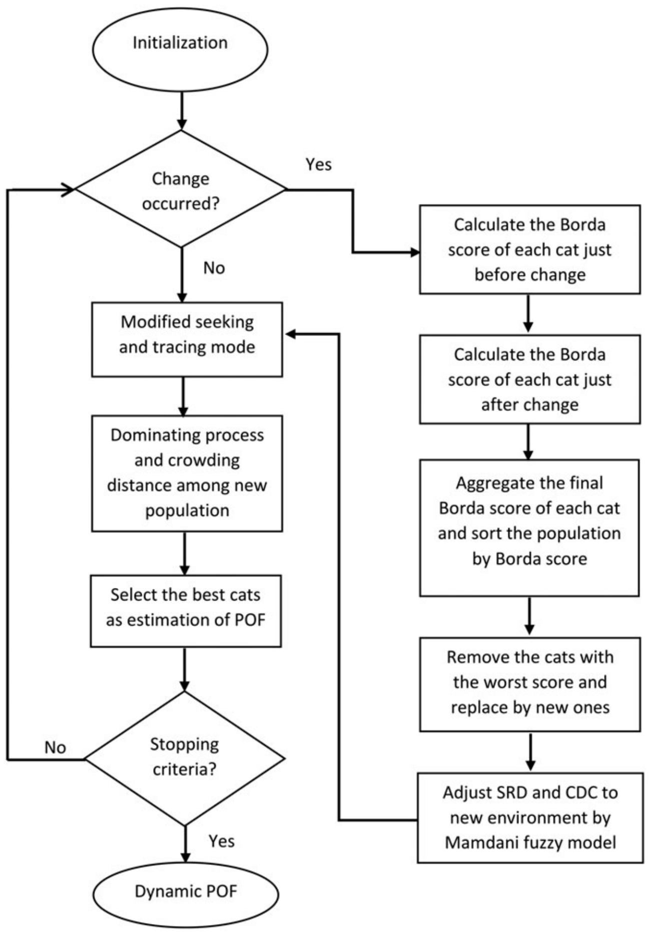
Initialization
Change occurred?
Yes
No
Calculate the Borda score of each cat just before change
Modified seeking and tracing mode
Calculate the Borda score of each cat just after change
Dominating process and crowding distance among new population
Aggregate the final Borda score of each cat and sort the population by Borda score
Select the best cats as estimation of POF
Remove the cats with the worst score and replace by new ones
No
Stopping criteria?
Adjust SRD and CDC to new environment by Mamdani fuzzy model
Yes
Dynamic POF

FIGURE 3 Flowchart of the proposed algorithm. POF = Pareto-optimal front; CDC = counts of dimension to change; SRD = seeking range of the selected dimension

TABLE 1 Performance measures [12]

Performance measure	Formula
VGD	$VD(t) = \frac{1}{\tau}\sum_{t=0}^{\tau} VD(t)I(t)$, $VD(t) = \frac{1}{n}\sqrt{n\sum_{i=1}^{n} d_i(t)^2}$, $I(t) = \left\{ \begin{array}{ll} 1, & \text{if } t\%\tau_t = 0 \\ 0, & \text{otherwise} \end{array} \right\}$
Spacing	$\overline{S}^i = \frac{1}{n_c}\sum_{j=1}^{n_c} S^i{}_j$, $S = \frac{1}{n}\sqrt{\frac{1}{n}\sum_{i=1}^{n} (d_i - \overline{d})^2}, \overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$
Maximum Spread	$\sqrt{\frac{1}{n_k}\sum_{k=1}^{n_k}\left[\frac{\min\left[\overline{POF^*}_k, \overline{POF'}_k\right] - \max\left[\underline{POF^*_k, POF'_k}\right]}{\overline{POF'}_k - \underline{POF'_k}}\right]^2}$
Stab	$Stab = \frac{1}{n_c}\sum_{i=1}^{n_c} stab(t)$, $stab(t) = \max\{0, acc(t-1) - acc(t)\}$
Accuracy	$\lvert HV(POF(t)) - HV(POF^*(t))\rvert$

Note. POF = Pareto-optimal front; VGD = variable generational distance.

In order to determine whether an algorithm can solve DMOOPs efficiently, DMOOPs should be used that test the ability of the algorithm to overcome certain difficulties. Table 2 investigates the DMOO benchmarks with continuous POF as follows: "FDA$_1$" has a convex POF with POF: $1-\sqrt{f_1}$, and POS is $x_i = G(t)$, $\forall x_i \in x_{II}$; "dMOP$_1$" has the POF: $(1 - f_1^{H(t)})$ that changes from convex to concave over time whereas the POS remains the same and is $x_i = 0$, $\forall x_i \in x_{II}$. In "dMOP$_2$," POF changes from convex to concave, where the values in both the POS and the POF change. POF of this function is $1 - f_1^{H(t)}$ similar to dMOP$_1$, and the POS is $x_i = G(t)$, $\forall x_i \in x_{II}$ as FDA$_1$. "HE$_2$" has a discontinuous POF with various disconnected continuous subregions. In this function, POF changes over time, but POS remains the same as $x_i = 0$, $\forall x_i \in x_{II}$ (Helbig, 2012).

TABLE 2 Dynamic multi-objective benchmarks [12]

Function	Formula	POF
$dMOP_1$	$\begin{cases} \text{Minimize } f(x,t)=(f_1(X_1), g(X_2).h(f_1(X_1), g(X_2,t))) \\ f_1(X_1,t)=x_1 \\ g(X_2,t)=1+9\sum_{x_i \epsilon X_2}(x_i)^2 \\ h(f_1,g,t)=1-\left(\frac{f_1}{g}\right)^{H(t)}, \\ \text{where} \\ H(t)=0.75\sin(0.5\pi t)+1.25, t=\frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ x_1\in[0,\ 1], X_2=(x_2,\ \ldots,\ x_n)\in[0,\ 1]^{n-1} \end{cases}$	$1-f_1^{H(t)}$
$dMOP_2$	$\begin{cases} \text{Minimize } f(x,t)=(f_1(X_1), g(X_2,t).h(f_1(X_1), g(X_2,t))) \\ f_1(X_1)=x_1 \\ g(X_2,t)=1+\sum_{x_i \epsilon X_2}(x_i-G(t))^2 \\ h(f_1,g)=1-\sqrt{\frac{f_1}{g}}, \\ \text{where} \\ H(t)=0.75\sin(0.5\pi t)+1.25, G(t)=\sin(0.5\pi t), t=\frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ x_1\in[0,\ 1], X_2=(x_2,\ \ldots,\ x_n)\in[-1,\ 1]^{n-1} \end{cases}$	$1-f_1^{H(t)}$
FDA_1	$\begin{cases} \text{Minimize } f(x,t)=(f_1(X_1), g(X_2,t).h(f_1(X_1), g(X_2,t))) \\ f_1(X_1)=x_1 \\ g(X_2,t)=1+\sum_{x_i \epsilon X_2}(x_i-G(t))^2 \\ h(f_1,g)=1-\sqrt{\frac{f_1}{g}}, \\ \text{where} \\ G(t)=\sin(0.5\pi t), t=\frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ x_1\in[0,\ 1], X_2=(x_2,\ \ldots,\ x_n)\in[-1,\ 1]^{n-1} \end{cases}$	$1-\sqrt{f_1}$
HE_2	$\begin{cases} \text{Minimize } f(x,t)=(f_1(X_1), g(X_2).h(f_1(X_1), g(X_2)), t))) \\ f_1(X_1)=x_1 \\ g(X_2)=1+\frac{9}{n-1}\sum_{x_i \epsilon X_2}x_i \\ h(f_1,g,t)=1-\left(\sqrt{\frac{f_1}{g}}\right)^{H(t)}-\left(\frac{f_1}{g}\right)^{H(t)}\sin(10\pi f_1), \\ \text{where} \\ H(t)=0.75\sin(0.5\pi t)+1.25, t=\frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ x_1\in[0,\ 1], X_2=(x_2,\ \ldots,\ x_n)\in[0,\ 1]^{n-1} \end{cases}$	$1-\sqrt{f_1}^{H(t)}-f_1^{H(t)}\times\sin(0.5\pi f_1)$

Note. POF = Pareto-optimal front.

4.1 | Analysing the results

Table 3 compares the obtained results by two proposed algorithms (using Borda method with normal/fuzzy adaptive CSO parameters) with five algorithms called $DNSGA_{II}$(A) (Deb et al., 2007) that randomly selects some particles and replaces by randomly created solutions, $DNSGA_{II}$(B) (Deb et al., 2007) that randomly selects some particles to be re-initialized and creates some new by mutating the current solutions, DVEPSO that hybrids the vector evaluating strategies with PSO (Helbig & Engelbrecht, 2013), dCOEA (Goh & Tan, 2009) that hybridizes competitive and cooperative mechanisms observed in nature, and PDNNIA (Liu et al., 2015) that estimates the new population after a change occurred. The parameters of each algorithm are the same as the reference paper. The simulations are performed on $dMOP_1$, $dMOP_2$, FDA_1, and HE_2 and compared by measures of VGD, Spacing, MS, and Stability. The results are indicated by mean and variance value, and the best found solutions are in bold.

4.1.1 | Results per function

The simulation results of function FDA_1 indicate that the proposed method gets the best results in comparison with other algorithms by mean value. It shows that using Borda count as the selection method and CSO with fuzzy adaptive parameters gets the best performance for measures of spacing and VGD at n_t = 10 and τ_t = 10 whereas Borda selection with normal CSO parameters obtained the best value for spacing, VGD, and MS at change frequency of 50. For function $dMOP_1$, the proposed method gets the best value of spacing for change frequency of 10, whereas PDNNIA obtains the better POF based on measures of VGD, and MS. Also, CSO with fuzzy parameters achieved the best value of Spacing, VGD, and MS for τ_t = 50. For $dMOP_2$, the proposed algorithm obtained the best value for measures of Spacing and VGD at change frequency

TABLE 3 Comparison between the proposed method and other approaches

Function	$n_t - \tau_t$	Algorithm		Spacing	Stab	VGD	MS
FDA$_1$	10–10	DVEPSOc	Mean	0.00043	**0.00154**	0.06593	0.9761
			SD	3.34E-3	2.890E-2	2.944E-2	4.568E-3
		DNSGAII(a)	Mean	0.00494	0.00339	0.83219	0.7869
			SD	3.69E-2	3.19E-2	2.369E-1	1.132E-3
		DNSGAII(b)	Mean	0.000612	0.00543	1.1339	1.1947
			SD	3.256E-2	4.123E-3	1.09E-2	2.22E-3
		dCOEA	Mean	0.00132	0.01328	1.1318	**2.485**
			SD	5.78E-5	5.13E-4	4.95E-4	3.36E-2
		PDNNIA	Mean	0.00027	**0.0054**	0.0668	**0.9989**
			SD	4.17E-4	7.03E-4	6.367E-3	5.50E-5
		Just Borda	Mean	0.00049	0.01030	0.05052	0.9483
			SD	4.124E-4	3.903E-4	1.95E-5	8.93E-5
		Borda + Fuzzy	Mean	**0.000158**	0.0076	**0.0418**	0.9971
			SD	5.18E-5	9.902E-5	1.23E-4	2.748E-4
	10–50	DVEPSOu	Mean	0.00033	0.00126	0.15148	0.9107
			SD	1.89E-5	3.339E-4	5.41E-4	1.123E-4
		DNSGAII(a)	Mean	0.00032	0.000030	0.1716	0.9885
			SD	2.901E-4	3.67E-4	5.78E-4	4.45E-4
		DNSGAII(b)	Mean	0.00033	**0.000022**	0.17261	0.9877
			SD	2.90E-4	3.956E-4	2.89E-4	9.34E-3
		dCOEA	Mean	0.00026	0.00017	0.1515	0.95904
			SD	1.1E-6	2.928E-5	2.55E-5	4.67E-4
		PDNNIA	Mean	0.000315	0.00279	0.089	0.9991
			SD	7.34E-5	3.456E-4	4.91E-4	4.45E-4
		Just Borda	Mean	**0.000134**	0.00522	**0.07578**	**0.9992**
			SD	7.1E-5	2.66E-3	3.69E-4	9.33E-5
		Borda + Fuzzy	Mean	0.000234	0.0154	0.0919	0.9978
			SD	2.45E-4	1.69E-4	4.24E-4	1.375E-4
dMOP$_1$	10–10	DNSGAII(a)	Mean	0.00577	**0.000036**	0.1521	0.9834
			SD	3.56E-3	1.45E-3	3.59E-3	6.70E-3
		DNSGAII(b)	Mean	0.00497	0.000059	0.1535	0.9397
			SD	3.23E-3	7.45E-3	7.59E-4	9.89E-4
		dCOEA	Mean	0.00045	0.00253	0.0389	0.8623
			SD	6.69E-4	4.61E-4	4.83E-4	7.89E-4
		DVEPSOc	Mean	0.0040	0.00035	0.2634	0.8790
			SD	5.56E-4	4.78E-4	5.12E-4	3.45E-4
		PDNNIA	Mean	0.000495	0.000125	**0.0094**	**0.9991**
			SD	5.12E-4	6.19E-4	5.93E-4	4.46E-5
		Just Borda	Mean	0.000399	0.00082	0.03517	0.9957
			SD	3.35E-4	5.70E-4	3.56E-5	8.03E-5
		Borda + Fuzzy	Mean	**0.000312**	0.000349	0.0755	0.9969
			SD	1.92E-5	2.89E-5	2.34E-5	3.99E-5
	10–50	DNSGAII(a)	Mean	0.00034	0.000013	0.1178	0.9832
			SD	3.68E-4	8.68E-4	7.12E-4	3.56E-4
		DNSGAII(b)	Mean	0.00032	**0.000012**	0.121	0.9833
			SD	3.56E-4	5.56E-4	4.67E-4	5.90E-4
		dCOEA	Mean	0.00023	0.00021	0.0957	0.9783
			SD	2.90E-4	1.45E-4	1.89E-4	1.34E-4
		DVEPSOu	Mean	0.00126	0.0088	1.603	0.6624
			SD	4.34E-4	3.56E-4	5.78E-4	5.56E-4
		PDNNIA	Mean	0.000416	0.00059	0.0693	0.9990
			SD	9.91E-5	2.65E-5	2.95E-5	2.23E-4
		Just Borda	Mean	0.000110	0.00010	0.0529	**1**
			SD	1.12E-4	1.45E-4	5.34E-4	0
		Borda + Fuzzy	Mean	**0.00008**	0.000062	**0.0216**	**1**
			SD	**1.01E-5**	9.67E-5	1.12E-4	0
dMOP$_2$	10–10	DNSGAII(a)	Mean	0.00095	0.00064	0.9041	**1.456**
			SD	4.59E-3	3.89E-4	3.78E-4	1E-5
		DNSGAII(b)	Mean	0.00212	0.00068	1.037	1.439
			SD	2.67E-4	3.9E-4	9.04E-5	8.89E-5
		dCOEA	Mean	0.0011	0.00213	0.8129	1.409
			SD	4.89E-4	5.90E-4	5.65E-4	4.44E-4
		DVEPSOd	Mean	0.00062	**0.00042**	0.0740	0.9793
			SD	3.95E-4	2.83E-5	1.23E-4	3.39E-4
		PDNNIA	Mean	0.00389	0.00110	0.0414	0.9989
			SD	1E-4	1.56E-4	3.94E-4	3.87E-4
		Just Borda	Mean	0.000332	0.00496	0.06779	0.9876
			SD	1.37E-5	9.45E-4	4.56E-5	2.23E-4
		Borda + Fuzzy	Mean	**0.000239**	0.0079	**0.0389**	0.9975
			SD	**1.03E-6**	2.05E-5	8.56E-5	9.93E-4
	10–50	DNSGAII(a)	Mean	0.00032	0.00014	0.1564	0.9955
			SD	7.23E-4	3.56E-4	5.55e-3	6.78E-5

(Continues)

TABLE 3 (Continued)

Function	$n_t - \tau_t$	Algorithm		Spacing	Stab	VGD	MS
		DNSGAII(b)	Mean	0.00032	**0.00012**	0.1406	0.9963
			SD	4.23E-4	2.12E-4	4.78E-3	4.12E-4
		dCOEA	Mean	0.00027	0.0022	0.1524	0.9543
			SD	2.23E-4	7.74E-4	5.6E-5	3.43E-4
		DVEPSOu	Mean	**0.00016**	0.044	0.1593	0.8733
			SD	1.01E-4	2.90E-4	3.56E-5	8.94E-5
		PDNNIA	Mean	0.000215	0.0029	0.081	0.9991
			SD	4.9E-4	5.78E-4	5.12E-4	5.98E-5
		Just Borda	Mean	0.000155	0.00310	0.02570	0.99808
			SD	1.9E-4	3.84E-4	5.34E-5	4.34E-5
		Borda + Fuzzy	Mean	0.000188	0.0157	**0.0216**	**0.9993**
			SD	6.3E-4	1.87E-4	3.94E-5	3.12E-4
HE_2	10–10	DNSGAII(a)	Mean	0.00062	0.00213	0.20337	0.9193
			SD	5.23E-4	4.78E-4	4.13E-4	3.79E-5
		DNSGAII(b)	Mean	0.00061	0.00206	0.20331	0.9208
			SD	2.45E-4	3.23E-4	4.63E-4	3.59E-5
		dCOEA	Mean	0.0045	0.0159	0.2345	0.6925
			SD	3.13E-4	2.58E-5	5.90E-4	9.45E-5
		DVEPSOc	Mean	0.01986	0.0130	1.524	0.9878
			SD	4.45E-4	5.12E-4	7.53E-4	1.03E-5
		PDNNIA	Mean	0.000119	0.00279	0.0366	**0.9595**
			SD	3.45E-4	6,03E-4	4.91E-4	3.53E-4
		Just Borda	Mean	0.000089	0.00014	0.04126	0.8613
			SD	4.80E-5	3.12E-5	2.63E-5	9.79E-5
		Borda + Fuzzy	Mean	**0.000081**	**0.00009**	**0.0123**	0.8901
			SD	1.93E-6	1.93E-5	3.12E-5	3.94E-4
	10–50	DNSGAII(a)	Mean	0.00063	0.00048	0.1913	0.91808
			SD	1.12E-4	4.89E-4	4.87E-4	4.95E-4
		DNSGAII(b)	Mean	0.00062	0.00048	0.1753	0.9179
			SD	1.93E-4	3.94E-4	3.72E-4	5.2E-4
		dCOEA	Mean	0.0014	0.00346	0.2538	0.8177
			SD	9.34E-4	3.89E-5	5.79E-5	2.23E-3
		DVEPSOp	Mean	0.01708	0.00244	1.7375	**1.1905**
			SD	3.94E-4	3.33E-3	4.23E-4	1.04E-4
		PDNNIA	Mean	0.000190	0.000612	0.0415	0.9614
			SD	3.78E-4	4.94E-4	6.23E-4	4.78E-5
		Just Borda	Mean	**0.000081**	0.000136	0.03363	0.84303
			SD	1.0E-6	3.14E-5	4.12E-5	2.93E-5
		Borda + Fuzzy	Mean	**0.000081**	**0.000082**	**0.0142**	0.8900
			SD	7.43E-5	3.83E-5	1.12E-4	4.95E-5

Note. Results are indicated by the mean and standard deviation (*SD*) value. VGD = variable generational distance; MS = Maximum Spread.

of 10, whereas it achieved the best VGD and MS at change frequency of 50. Finally, simulation results of HE_2 show that using Borda selection with fuzzy adaptive CSO obtains the best performance for measures of Spacing, Stab, and VGD for frequencies of change of 10 and 50. Also, standard deviation as a statistical comparison is done to identify the best performance of the algorithms. These results are indicated in Table 3 and state that Borda with fuzzy adaptive parameters has the best deviation at most cases. For example, at frequency of change 10 of function HE_2, the lowest standard deviation belongs to DVEPSO, Borda with normal CSO, and Borda with fuzzy CSO for measures of MS, VGD, and {Spacing–Stab}, respectively.

4.1.2 | Results per performance measure

The result of Table 3 can be analysed by diversity and accuracy of found solutions. For example, for FDA_1 at $n_t = 10$ and $\tau_t = 10$, the proposed method obtains the best results for measures of spacing for all conditions except dMOP2 at change frequency of 50. Also, this method has the better performance of VGD and MS in comparison with other algorithms for all conditions at change frequency of 50.

4.1.3 | Performance results over change

This section compares the performance of the proposed method with $DNSGA_{II}$ over consecutive changes (at $n_t = 10$ and $\tau_t = 10$) by measure of accuracy. Figure 4 shows the accuracy of obtained solutions over time where accuracy measures the quality of the solutions as a relation between the HV of POF^* and the maximum HV that has been found so far. In this figure, the horizontal and vertical axes are number of change and accuracy value after each change, respectively. As the figure indicates the proposed method in comparison with $DNSGA_2$ gets the quite satisfactory results for all four functions. For example, in FDA_1, the changes greatly affect the accuracy measure of the $DNSGA_2$ at change number of 2 and 21, whereas there is no huge change on the performance of proposed method. Also, environmental changes have corrupting influence on the accuracy performance of $DNSGA_2$ for function of $dMOP_1$ and $dMOP_2$ at change number of 16, 30 and 2, 22, respectively.

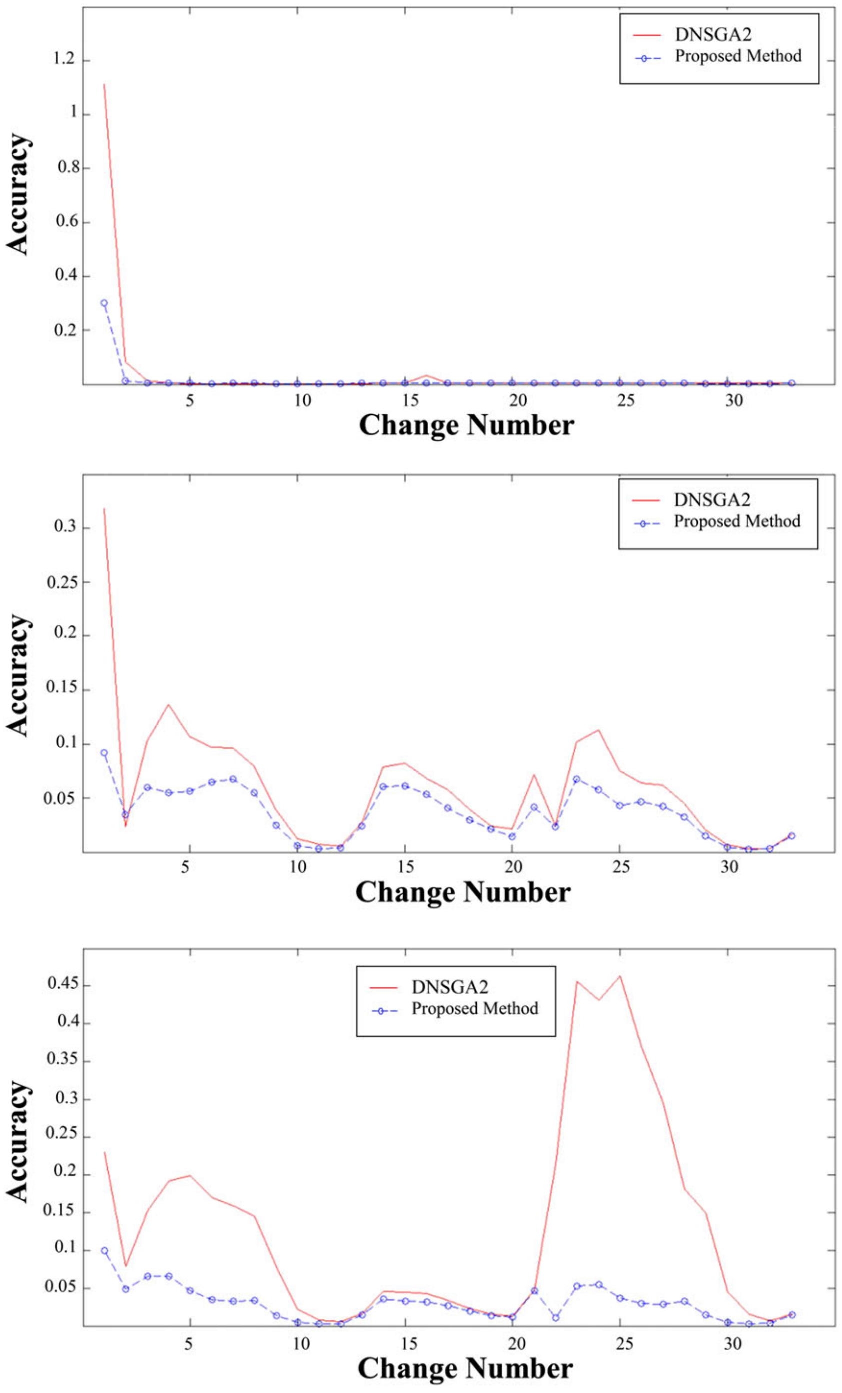

FIGURE 4 Comparison of proposed method (Blue: --o) with DNSGA$_2$ (Red: --) by measure of Stability for consecutive changes: Up to down: dMOP$_1$, dMOP$_2$, FDA$_1$

4.1.4 | Comparison of execution time

This section investigates the execution time of two proposed methods and compares with DNSGAII(a), which is based on the second and obtained at severity of change 10, frequency of change 10 and 50, and number of iterations 1,000. Both Table 4 and Figure 5 indicate that using Borda count method in order to select the worst individuals of population increases the running time of the algorithm. It is clear that calculating the score of each individual just before and after change and sorting the aggregated scores may increase the execution time. Also, when a change occurs, adapting the CSO parameters to new environments by fuzzy rules may increase the setup time of CSO. For example, execution time of normal CSO for FDA$_1$ at frequency and severity of change 10 is 28.8 s, whereas it is 33.1 when using fuzzy CSO.

4.1.5 | Statistical analysis

This section uses Kruskal–Wallis test to investigate the statistical analysis that was done on the values of performance measures. Kruskal–Wallis test (one-way analysis of variance on ranks) as a nonparametric method is used for comparing two or more independent samples of equal or different sample sizes. A significant Kruskal–Wallis test indicates that at least one sample stochastically dominates one other sample. So this test was

TABLE 4 Comparison of the execution time (second) between DNSGAII(a) and two proposed algorithms

$n_t - \tau_t$	Function	FDA_1	$dMOP_1$	$dMOP_2$	HE_2
10–10	DNSGAII(a)	21.6	24.4	29	34.1
	Just Borda	28.8	30.6	37.3	44.8
	Borda + Fuzzy	33.1	39.5	42.5	80.2
10–50	DNSGAII(a)	15	16.5	27.2	33.3
	Just Borda	15.9	17.1	28.4	35.2
	Borda + Fuzzy	18.3	20	31.1	40.6

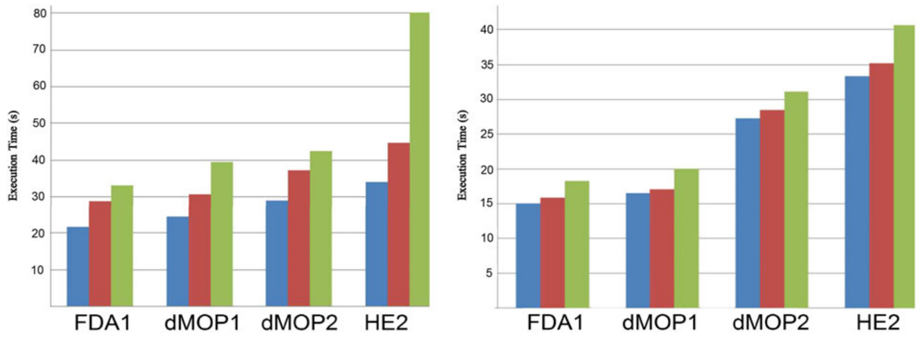

FIGURE 5 Comparison of execution time (second) between DNSGAII(a) and two proposed methods. Blue: DNSGAII(a), red: Just Borda, and green: Borda + Fuzzy cat. Left: frequency of change: 10, severity of change: 10, and number of iterations: 1,000. Right: frequency of change: 50, severity of change: 10, and number of iterations: 1,000

performed to determine whether there was a statistically significant difference (confidence level of 95%) between the values obtained by "DNSGAII(a)," "DNSGAII(b)," "Just Borda," and "Borda + Fuzzy" for a performance metric for a specific function at a specific τ_t. The obtained *p* values by Kruskal–Wallis tests are presented in Table 5 where bold values indicate the statistically significant difference. Table 5 presents the obtained *p* values of performance measures for functions: FDA_1, $dMOP_1$, $dMOP_2$, and HE_2 at frequency of change 10 and 50. It shows that there is a statistically significant difference between all of the algorithms for a change frequency of 10 and 50 except for Stab and VGD for function $dMOP_2$.

When the *p* value of the Kruskal–Wallis test indicated that there was a statistically significant difference, Mann–Whitney *U* tests were performed to determine which algorithms' performance metric values there were a statistically significant difference. This test is also performed under confidence level of 95%. Tables 6a, 6b, 6c, 6d present the results of the Mann–Whitney *U* tests at τ_t = 10. In all these tables, "-" indicates that there was no statistically significant difference between the specific performance metric values. Here "D(a)," "D(b)," "B," and "BF" refer to DNSGAII(a), DNSGAII(b), Just Borda, and Borda + Fuzzy, respectively.

Table 6a shows that for FDA_1, there is a statistically significant difference between almost all the algorithms for Spacing, Stab, and VGD. Table 6b also indicates the statistically significant difference between almost all algorithms for running $dMOP_1$. Table 6c presents that there is no statistically significant difference when comparing four algorithms based on Stab for function $dMOP_2$, but there is a statistically significant difference when comparing the algorithms for other performance measures. Table 6d shows the significant difference between almost all algorithms for HE_2. For example, for measure of MS, each pair of algorithms has statistically significant difference except {DNSGAII(b), Just Borda}.

TABLE 5 *p* Values of Kruskal–Wallis test

τ_t	Function	Spacing	Stab	VGD	MS
10	FDA_1	**1.51E-4**	**2.4E-3**	**1E-3**	**1.9E-2**
	$dMOP_1$	**6.48E-4**	**7.9E-3**	**1.8E-2**	**2.8E-2**
	$dMOP_2$	**1.01E-3**	0.0701	**9.4E-3**	**4.1E-2**
	HE_2	**6.98E-3**	**1.8E-2**	**2.3E-3**	**9E-3**
50	FDA_1	**1.302E-3**	**2.4E-2**	**2.85E-4**	**1.95E-3**
	$dMOP_1$	**2.84E-5**	**1.1E-4**	**3.914E-4**	**2.22E-3**
	$dMOP_2$	**2.29E-3**	**3.47E-3**	0.0627	**1E-2**
	HE_2	**8.91E-4**	**2.87E-4**	**1.95E-3**	**2.74E-4**

Note. VGD = variable generational distance; MS = Maximum Spread.

TABLE 6A Results of Mann–Whitney *U* test for FDA_1 at $r\ \tau_t = 10$

Measure	Algorithm	D(a)	D(b)	B	BF
Spacing	D(a)	n/a			
	D(b)	2.3E-5	n/a		
	B	3.12E-3	2.1E-3	n/a	
	BF	1.93E-4	4.5E-2	-	n/a
Stab	D(a)	n/a			
	D(b)	-	n/a		
	B	2.1E-3	-	n/a	
	BF	3.7E-3	1.1E-2	4.1E-2	n/a
VGD	D(a)	n/a			
	D(b)	1.6E-3	n/a		
	B	2.7E-3	4.9E-3	n/a	
	BF	8.6E-3	6.2E-3	-	n/a
MS	D(a)	n/a			
	D(b)	-	n/a		
	B	2.4E-2	-	n/a	
	BF	2.2E-2	3.5E-3	-	n/a

Note. VGD = variable generational distance; MS = Maximum Spread.

TABLE 6B Results of Mann–Whitney U test for $dMOP_1$ at r $\tau_t = 10$

Measure	Algorithm	D(a)	D(b)	B	BF
Spacing	D(a)	n/a			
	D(b)	1.1E-4	n/a		
	B	1.9E-5	2.8E-4	n/a	
	BF	2.12E-3	3.1E-3	-	n/a
Stab	D(a)	n/a			
	D(b)	-	n/a		
	B	1.7E-3	2.7E-2	n/a	
	BF	3.6E-3	4.9E-2	-	n/a
VGD	D(a)	n/a			
	D(b)	-	n/a		
	B	-	3.1E-3	n/a	
	BF	5.1E-3	6.1E-3	1.2E-2	n/a
MS	D(a)	n/a			
	D(b)	7.3E-3	n/a		
	B	3.2E-2	8.4E-3	n/a	
	BF	2.5E-2	-	1E-2	n/a

Note. VGD = variable generational distance; MS = Maximum Spread.

TABLE 6C Results of Mann–Whitney *U* test for $dMOP_2$ at $r\ \tau_t = 10$

Measure	Algorithm	D(a)	D(b)	B	BF
Spacing	D(a)	n/a			
	D(b)	5.2E-4	n/a		
	B	9.2E-3	-	n/a	
	BF	2.23E-3	1.9E-2	4.9E-2	n/a
Stab	D(a)	n/a			
	D(b)	-	n/a		
	B	-	-	n/a	
	BF	-	-	-	n/a
VGD	D(a)	n/a			
	D(b)	-	n/a		
	B	2.6E-3	2.8E-2	n/a	
	BF	1.9E-4	1.1E-2	-	n/a
MS	D(a)	n/a			
	D(b)	-	n/a		
	B	1.1E-2	-	n/a	
	BF	5.2E-2	3.8E-2	9.4E-3	n/a

Note. VGD = variable generational distance; MS = Maximum Spread.

TABLE 6D Results of Mann–Whitney *U* test for HE_2 at $r\ \tau_t = 10$

Measure	Algorithm	D(a)	D(b)	B	BF
Spacing	D(a)	n/a			
	D(b)	1.3E-4	n/a		
	B	3E-2	2.6E-2	n/a	
	BF	7.3E-3	3.1E-2	4.7E-2	n/a
Stab	D(a)	n/a			
	D(b)	-	n/a		
	B	2.7E-3	4E-2	n/a	
	BF	3.6E-3	3.3E-2	-	n/a
VGD	D(a)	n/a			
	D(b)	6.1E-3	n/a		
	B	-	1.7E-2	n/a	
	BF	2.9E-3	-	-	n/a
MS	D(a)	n/a			
	D(b)	2.2E-2	n/a		
	B	9.4E-3	-	n/a	
	BF	4.2E-2	6.2E-3	2E-2	n/a

Note. VGD = variable generational distance; MS = Maximum Spread.

5 | CONCLUSIONS

The main goal of this paper is to introduce a novel DMOO algorithm. The proposed algorithm applies Borda count as an optimal rank aggregation method on the population of CSO to determine the cats with the most appropriate fitness to be re-initialized and uses Mamdani fuzzy logic to adjust the parameters of optimization algorithm to new environment in order to enhance the diversity of population when a change occurred. Experimental results indicate that the proposed method has the best performance in comparison with other algorithms and gets the quite satisfactory results based on measures of VGD, Spacing, and MS for benchmarks with continuous and discontinuous POF.

ACKNOWLEDGEMENT

This work is supported by Shenzhen Science and Technology Innovation Commission (SZSTI) project JCYJ20170302153752613.

CONFLICT OF INTEREST

None.

ORCID

Maysam Orouskhani http://orcid.org/0000-0003-3341-3904

REFERENCES

Avdagic, Z., Konijicija, K., & Omanovic, S. (2009). Evolutionary approach to solving nonstationary dynamic multi-objective problems. *Foundations of Computational Intelligence*, *3*, 267–289.

Borda, JC (1781) Memoire sur les elections au scrutin. Histoire de l'Academie des Sciences

Chu, S., Tsai, P., & Pan, J. (2006). Cat swarm optimization. *Cat Swarm Optimization, Springer Lecture Note in Artificial Intelligence*, *4099*, 854–858.

Dang, Y., & Wang, C. (2008). An evolutionary algorithm for dynamic multi-objective optimization. *Applied Mathematics and Computation*, *25*, 6–18.

Deb, K., Rao, N., & Karthik, S. (2007). Dynamic mylti-objective optimization and decision making using modified NSGA2: A case study on hydro thermal power scheduling. *Lecture Note on Computer Science*, *4403*, 803–817.

Farina, M., Deb, K., & Amato, p. (2004). Dynamic multiobjective optimization problems: Test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, *8*(5), 425–442.

Goh, C., & Tan, K. (2009). A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *13*(1), 103–127.

Helbig, M. (2012). *Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation*. Pretoria: University of Pretoria.

Helbig, M., & Engelbrecht, A. (2013). Dynamic multi-objective optimization using PSO. In E. Alba (Ed.), *Methaheuristic for dynamic optimization* (pp. 147–188). Berlin Heidelberg: Springer.

Helbig, M., & Engelbrecht, A. (2014). Heterogeneous dynamic vector evaluated particle swarm optimisation for dynamic multi-objective optimisation. IEEE Congress on Evolutionary Computation (CEC). China

Li, Y., & Wang, B. (2009). Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment. In *Proceedings of Congress on Evolutionary Computation*, 630–637.

Lin, S. (2010). Rank aggregation methods. *WIREs Computational Statistics*, *2*, 555–570. https://doi.org/10.1002/wics.111

Liu, R., Fan, J., & Jiao, L. (2015). Integration of improved predictive model and adaptive differential evolution based dynamic multi-objective evolutionary optimization algorithm. *Applied Intelligence*, *43*, 192–207.

Liu, R., Nui, X., Fan, J., Mu, C., & Jiao, L. (2014). An orthogonal predictive model-based dynamic multi-objective optimization algorithm. *Soft Computing*, *19*, 3083–3107.

Mantysaari, J., & Hamalainen, R. (2001). A dynamic interval goal programming approach to the regulation of a lake-river system. *Journal of Multi-Criteria Decision Analysis*, 75–86.

Mantysaari, J., & Hamalainen, R. (2002). Dynamic multi-objective heating optimization. *European Journal of Operational Research*, 1–15.

Muruganantham A, Zhao Y, Gee S, & Qiu X. (2013). Dynamic multiobjective optimization using evolutionary algorithm with Kalman filter. 17th Asia Pacific Symposium on Intelligent and Evolutionary Systems, IES 2013

Orouskhani, M., Teshnehlab, M., & Nekoui, M. A. (2017). Evolutionary dynamic multi-objective optimization algorithm based on Borda count method. *International Journal of Machine Learning and Cybernetics*. https://doi.org/10.1007/s13042-017-0695-3

Zhou, A., Zhang, Q., Sendhoff, B., & Tsang, E. (2007). Prediction-based population reinitialization for evolutionary dynamic multi-objective optimization. The 4th International Conference on Evolutionary Multi-criterion Optimization.

Maysam Orouskhani is PhD of Computer Engineering in the field of Artificial Intelligence who is an expert in the field of Evolutionary Algorithms, Computational Intelligence, and Multi-objective optimization and its applications on engineering problems.

Daming Shi is a Distinguished Professor at Research Institute for Future Media Computing. He is an active researcher in Machine Learning and Computer Vision for many years.

How to cite this article: Orouskhani M, Shi D. Fuzzy adaptive cat swarm algorithm and Borda method for solving dynamic multi-objective problems. *Expert Systems*. 2018;e12286. https://doi.org/10.1111/exsy.12286