Memory-Assisted Dynamic Multi-Objective Evolutionary Algorithm for Feature Drift Problem

Shaaban Sahmoud
Computer Engineering Department
Fatih Sultan Mehmet Vakif University
Istanbul, Turkey
ssahmoud@fsm.edu.tr

Haluk Rahmi Topcuoglu
Computer Engineering Department
Marmara University
Istanbul, Turkey
haluk@marmara.edu.tr

***Abstract*—In this paper, we propose an enhanced feature selection algorithm able to cope with feature drift problem that may occur in data streams, where the set of relevant features change over time. We utilize a dynamic multi-objective evolutionary algorithm to continuously search for the updated set of relevant features after the occurrence of every change in the environment. An artificial neural network is employed to classify the new instances based on the up-to-date obtained set of relevant features efficiently. Our algorithm exploits a detection mechanism for the severity of changes to estimate the severity level of occurred changes and adaptively replies to these changes by introducing diversity to algorithm solutions. Furthermore, a fixed-size memory is used to store the good solutions and reuse them after each change to accelerate the convergence and searching process of the algorithm. The experimental results using three datasets and different environmental parameters show that the combination of our improved feature selection algorithm with the artificial neural network outperforms related work.**

***Index Terms*—dynamic multi-objective evolutionary algorithms, learning in non-stationary environments, severity of changes, feature drift, memory-based algorithms**

I. Introduction

Dynamic Multi-Objective Optimization Problems (DMOPs) are increasingly being used to tackle real-world multi-objective optimization problems that have non-stationary nature [1]. The existence of dynamism is the main difference between a normal Multi-Objective Optimization Problem (MOP) and a Dynamic Multi-Objective Optimization Problem (DMOP), where it can occur due to various factors such as changes in objective functions, changes in problem variables, or changes in constraints [2]. In real life, there are various examples from different domains for multi-objective optimization problems which are dynamic in nature including scheduling problems [3], routing problems in networking [4], resource management systems [5], control problems [6], hydro-thermal power systems [7], and mobile ad-hoc networks [8].

Population-based meta-heuristic optimization algorithms inspired by the evolution of the species are among the common and popular techniques that have been adapted to solve dynamic multi-objective optimization problems [1]. The existence of dynamism in the environment makes solving DMOPs more challenging since the main goal becomes to obtain a diverse set of candidates/non-dominated solutions and to track them after each environmental change. This is because the set of good solutions that the algorithm obtained before a change may not be good or valid solutions for the new environment after the change. In multi-objective optimization environments, the set of candidates/non-dominated solutions is called the Pareto Optimal Set (POS) in the decision space and the Pareto Optimal Front (POF) in the objective space. In literature, there are a large number of Dynamic Multi-Objective Evolutionary Algorithms (DMOEAs) that are proposed to solve DMOPs, where these algorithms can be classified as diversity introduction [7], [9], diversity maintenance [10], memory-based [11], [12], prediction-based [13], [14], and multi-population approaches [15], [16].

In this paper, we propose an enhanced dynamic multi-objective evolutionary algorithm to solve the dynamic feature selection problem that occurs in data streams. The proposed DMOEA keeps searching for the set of relevant features that change over time after each environmental change and send it to the predictive model to readapt on the new coming environments during data stream classification. We proposed the Dynamic Filter-Based Feature Selection (DFBFS) [17] algorithm for handling this problem which selects the set of up-to-date relevant features using a filter-based feature selection mechanism and an Artificial Neural Network (ANN). It was observed that while the performance and accuracy of the DFBFS algorithm are better than other related algorithms, it needs a relatively higher time to accomplish the same task. The proposed algorithm in this paper is an enhanced version of the DFBFS algorithm to overcome the high execution time and enhance the classification accuracy. In our new proposed evolutionary algorithm, we exploit a detection mechanism for severity of changes to estimate the extent of occurred feature drift and adaptively reply to it by introducing diversity to solutions. Moreover, a fixed size memory is used to store the good solutions and reuse them after each change to accelerate the searching process of the algorithm and rapidly find the relevant features in the case of cyclic or repeated feature drifts that possibly occur in data streams. To classify the new arrived samples, an ANN model is incorporated with our proposed DMOEA.

The remainder of the paper is organized as follows. Section 2 gives a background for the feature drift problem and

DMOPs. Our algorithm is presented in Section 3. Section 4 presents the experiment settings, benchmarks, and the results of our empirical study for evaluating the performance of the proposed algorithm. Finally, the paper conclusion and future research work are given in Section 5.

II. Background

A. Feature Selection in Data Streams

Data streams have been attracting the attention of data experts due to its widespread use in different areas including fraud detection, internet search logs, network monitoring, social networks, video surveillance, and sensor networks [18]. Usually, data samples of data streams arrive rapidly and continuously which makes it very difficult to store the upcoming samples. Therefore, this data needs to be processed online and sequentially on a sample-by-sample basis or over sliding time windows. The non-stationary nature of newly arriving data is one of the common challenges of data streams; since in many real-world applications, the arriving data keep changing over time as in the problem of spam detection given in [19], [20]. As a result, to derive an efficient classifier for non-stationary data streams, the classifier should be continuously updated to cope with new arriving data samples.

To process or classify the samples of data streams, a machine learning model usually needs a set of features or attributes. These features may have varied significance levels which require a suitable selection mechanism called *feature selection* to choose the relevant features only from the original feature set [21]. Feature selection mechanisms have a significant role in minimizing the dimensionality of data and enhancing the performance of classifiers. *Feature drift* is a type of data stream drifts that occur when a subset of features switches partially/completely from relevant to irrelevant/redundant or vice versa. Therefore, to keep processing the new arriving instances correctly, the used classifier should be periodically updated with the new subset of relevant features after every feature drift [22]. In addition, since the samples of data continuously arrive, the classifier should re-adapt its model as fast as possible to minimize the number of misclassified samples after feature drift occurs [23]. In literature, feature selection methods are divided into two main categories which are *filter-based* and *wrapper-based* [21]. The filter-based methods [24], [25] apply one or more statistical techniques to rank the features, and then select features with high ranks to represent the set of relevant features. Their main advantage is the low computational costs. In wrapper-based methods, a classification algorithm is used to evaluate the relevance of the tested features [26], [27]. While wrapper-based methods are computationally more expensive than the filter-based methods, they usually achieve better performance [28].

B. Dynamic Multi-objective Optimization Problems

A dynamic multi-objective optimization problem (DMOP) is an optimization problem with two or more objective functions and at least two of them are conflicting with each other, where the objective(s), variables, constraints, and parameters of the problem may change over time. The following equation gives the formal definition of a DMOP:

$$\begin{aligned}
&\text{minimize f(x,t)} = \{f_1(x,t), f_2(x,t), ...f_M(x,t)\}\\
&\text{subject to}\\
&\qquad g(x,t) \leq 0,\\
&\qquad h(x,t) = 0, \qquad\qquad\qquad\qquad\qquad (1)\\
&\qquad x = (x_1, x_2, ...x_n) \quad and \quad x \in [x_{min}, x_{max}]
\end{aligned}$$

where x is the vector of problem input variables, and M is the number of objectives in the problem. f(x,t) is the set of dynamic objectives with respect to time t. $g(x,t)$ and $h(x,t)$ represent the inequality and equality constraints of the problem.

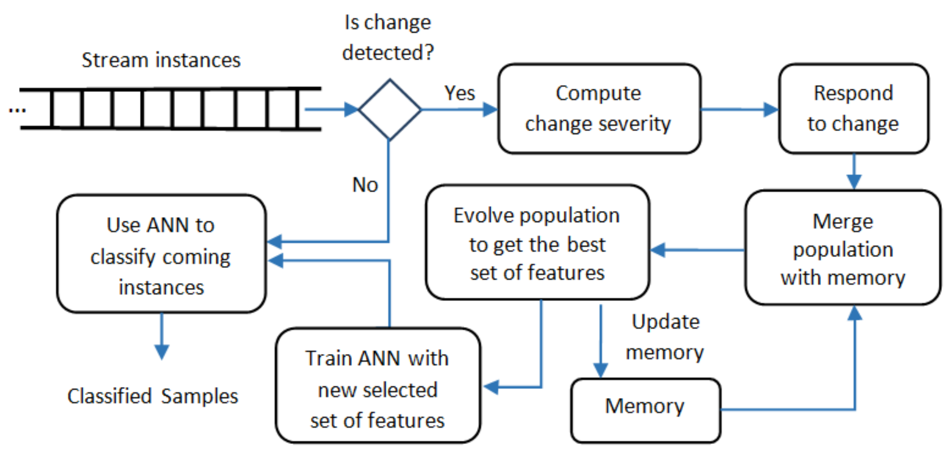
For a DMOP, a change in the landscape may affect the Pareto Optimal Set (POS), the Pareto Optimal Front (POF) or both of them. Therefore, a dynamic multi-objective evolutionary algorithm (DMOEA) should track the dynamic POF after each environmental change. Specifically, a good DMOEA should be able to evolve a diverse set of solutions to converge to the new POF rapidly before the next change occurs. Based on where the change happens, Farina et al. [29] classified the DMOPs into four types as follow:

- *Type I.* The POF remains static where the POS changes.
- *Type II.* The POF and the POS change over time.
- *Type III.* The POF changes over time where the POS remains static.
- *Type IV.* The POF and the POS do not change over time.

III. Memory-Assisted Dynamic Multi-Objective Evolutionary Algorithm

Our proposed algorithm is a Dynamic Multi-Objective Evolutionary Algorithm (DMOEA) that uses a dynamic change response mechanism, and memory for collecting a set of previous good solutions to select the dynamic set of relevant features, where the population is dynamically updated with respect to the severity of environmental changes. It basically employs a feature drift detection mechanism to check if there are any significant changes in the current set of relevant features that used by the classifier and enhances the population of the DMOEA to quickly search for the new set of relevant features for the new environment. It can efficiently handle a variety of non-stationary environments that may occur as a result of different types of feature drift, such as slow or fast feature drifts, variable rate or cyclical feature drifts, and low or high severity of feature drifts. Moreover, it is one of the few evolutionary algorithms that developed to handle the dynamic feature selection problem. Incorporating a memory scheme and a dynamic change response mechanism enables our proposed algorithm to overcome the limitations of the DFBFS algorithm [17], which requires relatively higher time and has no mechanism to store the best solutions from previous environments (set of relevant features for old environments).

Figure 1 presents the flow of our algorithm. As shown in the figure, a change detection mechanism is continuously



Stream instances
...
Is change detected?
Yes
No
Compute change severity
Respond to change
Merge population with memory
Evolve population to get the best set of features
Update memory
Memory
Use ANN to classify coming instances
Train ANN with new selected set of features
Classified Samples

Fig. 1. The flow of our dynamic multi-objective evolutionary algorithm.

performed to detect if there is any feature drift. If a feature drift is detected, then the severity of the change is computed by comparing the extent of the change in the old and the current environments. In this paper, we used our previously proposed detection mechanism for severity of changes which developed to deal with changes of DMOPs [30]. It randomly selects a set of solutions called (sensors), then, for each objective function, the difference between the values of each sensor before and after the change is computed separately. The severity for each objective function is estimated first and then the maximum severity is selected to be the overall severity of the current environmental change. According to results given in [30], this method outperforms the other methods that detect the severity of change and overcomes their problems. According to the severity of change value, a variable and a convenient number of new randomly generated solutions are inserted into the current population. Adding new random solutions to the population helps the evolutionary algorithm in exploring new areas in the fitness space during the searching process of the new relevant features for the new environment. To avoid misleading population or getting undesired results, the number of newly inserted solutions is computed according to the current change severity. As a result, if the severity of change has a low value, then there is a slight change in the environment and a small number of random solutions is generated and added to the population. If the severity is high, it indicates a sever change occurrence and a large number of new random solutions are inserted. To control the number of new randomly generated solutions and prevent the current population from full replacement, we limit the number of newly inserted solutions to be less than 20% of the population size. The pseudocode of the proposed algorithm is given in Algorithm 1.

To evaluate the solutions of the population, two objectives are utilized which are reducing the number of features and maximizing the relevance between the class label and features. Because of the limited available processing time when

Algorithm 1 The Proposed Dynamic Multi-Objective Evolutionary Algorithm for Feature Selection with Feature Drifts

```
begin
Initialize randomly parent and offspring populations;
while Data stream instances are still arriving do
    if Change is detected in the stream (feature drift) then
        Estimate the severity of feature drift;
        Respond to change dynamically using the estimated severity by
        replacing a number of population solutions;
        Select the best solution for the new environment from memory and
        insert it to population;
        while Stop condition is not satisfied do
            Evaluate population solutions;
            Merge children and parents populations;
            Rank population solutions using non-domination sorting and
            crowding distance mechanisms;
            Select the best N solutions for next-generation;
            Apply the crossover and the mutation operators;
        end while
        Select the best solution from the non-dominating solutions of the last
        generation to determine the optimal set of features;
        Update the memory by adding the best solution to it;
        Train the ANN model using the new selected set of relevant features
        obtained from the best solution;
        Classify the current instance using the trained ANN;
    else
        Classify this instance using the previously trained ANN model;
    end if
end while
end
```

dealing with online data streams, it becomes unhelpful to use a wrapper-based feature selection method since it requires high computational time. Therefore, we utilized a filter-based feature selection method to select the best set of relevant features in the second objective of the proposed algorithm. In our algorithm, the mutual information measure is used as described in [17]. The solutions are encoded using the binary representation where each solution represents N features by a string of bits. For each bit, zero value indicates the exclusion of the represented feature, and one value indicates the inclusion of it. The stopping criterion of the proposed algorithm is determined dynamically for each new environment during the population evolving process. Using a dynamic stop criterion is more suitable for online data streams than a fixed number of generations using a constant value because the processing time is very limited and in some cases, the same environment may occur multiple times. As a result, if we have a new change that occurred before, then we can use the memory to fetch the set of relevant features for this change and run the evolutionary algorithm for a small number of generations. On the other hand, if the change occurs for the first time, then more generations should be performed to find its new set of relevant features. In this work, we stop the evolving process if all solutions of the population become in rank one (non-dominated) for 10 consecutive generations, which means that the minimum number of executed generations is 10, and the maximum number of generations is selected to be 100.

A. A Memory-Based Extension for Our Algorithm

Memory-based techniques are very effective when dealing with optimization problems that have periodic environmental changes. In data streams that exhibit feature drifts, it is possible that a set of relevant features changes to redundant or irrelevant features and then return again to be relevant after a certain number of environmental changes. When a feature drift occurs in the environment, the new set of relevant features can be:-

- completely new feature set that did not occur before and is not close to any previously used feature sets, or
- completely repeated set that occurred before in previous environments, or
- a feature set that did not occur before but it is very close to one of the previously used feature sets.

While in the first scenario the memory mechanisms do not add any value to the performance of the algorithm, in the second and third scenarios, the memory becomes very effective and can significantly enhance the performance of the evolutionary algorithms. Moreover, in data streams usually, the set of relevant features has a high probability to reoccur in the future since the data stream is continuously coming and the size of features is relatively small or limited. In the proposed algorithm an explicit memory scheme is utilized to store the previous best solutions (set of relevant features in old environments) and to reuse them later after other environmental changes occur. As shown in Figure 1, the memory updates itself after each change with good solutions and provides the population with efficient solutions from the previous environments. Specifically, after each change detection, solutions of the memory are reevaluated and the best one is inserted to the current population by replacing a random solution. Then, the evolutionary algorithm starts evolving the solutions of the population to find the best solution, where the memory continuously stores the good solutions just before the detection of the next change. The memory stores the set of relevant features for each different environment during data stream processing. Note that this memory scheme is not utilized to handle the periodic changes only, but it can handle all previously occurred changes if they recurred again even if they are not periodic which is a common behavior in real-world streaming problems.

IV. Experiments and Discussion

Several datasets simulating different dynamic scenarios (such as the severity of change denoted by n_t and the frequency of change denoted by τ_t) have been generated to validate the performance of our algorithm. Mainly, two well-known benchmarks are used, which are the Binary Generator with Feature Drift (BGFD) [22] and the SEA generator with Feature Drift (SEAFD) [31]. The BGFD is a benchmark designed based on Binary Generator (BG) [32] and used to generate synthetic data streams with a dynamic set of relevant features that change over time. In BGFD, at every different environment, there is only a subset S of relevant features, where $S \subset F$ and F is the set of all available features. Different datasets can be generated by considering different relations between relevant features and class labels. Using BGFD, we generated two datasets (BGFD1 and BGFD2) using the following two equations:

$$Y_{BGFD1} = \begin{cases} 1, & if(X_1 \wedge X_2) \\ 0, & otherwise \end{cases} \tag{2}$$

$$Y_{BGFD2} = \begin{cases} 1, & if(X_1 \vee X_2) \\ 0, & otherwise \end{cases} \tag{3}$$

where $S = \{X1, X2\}$, is the set of relevant features. Y_{BGFD1} and Y_{BGFD2} are the class labels of the two generated datasets.

The third dataset is generated using SEAFD [31] which is an extension of the SEA generator [33]. In SEAFD, the features are numbers between 0 and 10, where the set of relevant features are selected randomly after each environmental change using the following Equation:

$$Y_{SEAFD} = \begin{cases} 1, & if((X_1 + X_2) \leq \theta) \\ 0, & otherwise \end{cases} \tag{4}$$

where θ is a threshold determined once for all instances of the generated stream.

We compared the performance of the proposed algorithm with the performance of three algorithms which are the Very Fast Decision Tree (VFDT) [34], the Hoeffding Adaptive Tree (HAT) [35], and the DFBFS [17]. The VFDT is a fast algorithm that is developed based on decision trees and it is widely used in classifying data streams. The HAT algorithm is designed based on ADWIN algorithm [36] and utilizes a decision tree with a sliding window to classify the data stream instances. The DFBFS algorithm is a recently proposed algorithm that uses a DMOEA to handle feature drifts in data streams.

Two metrics are used to measure the performance of compared algorithms which are the *average classification accuracy* and the *average window classification accuracy*. The former computes the average classification accuracy for all instances, whereas the latter computes the average accuracy over a sliding window of arriving instances. The two metrics

TABLE I
COMPARING THE CLASSIFICATION ACCURACY OF ALGORITHMS FOR THE BGFD1 DATASET

(τ_t, n_t)	Proposed	HAT	VFDT	DFBFS
(1000,1)	**96.52**	90.73	85.74	94.42
(1000,2)	91.05	80.53	79.82	**92.34**
(2000,1)	**95.71**	92.69	87.81	90.33
(2000,2)	**96.27**	83.51	80.97	93.72
(4000,1)	**97.24**	95.61	89.97	94.41
(4000,2)	**96.34**	89.97	82.37	95.17
(6000,1)	94.94	**96.42**	89.89	95.83
(6000,2)	94.74	**95.88**	83.81	95.31
(8000,1)	**97.31**	96.05	90.72	96.45
(8000,2)	92.25	95.86	86.59	**96.12**

TABLE II
COMPARING THE CLASSIFICATION ACCURACY OF ALGORITHMS FOR THE BGFD2 DATASET

(τ_t, n_t)	Proposed	HAT	VFDT	DFBFS
(1000,1)	**94.37**	92.04	87.32	92.22
(1000,2)	**94.87**	79.61	78.92	92.94
(2000,1)	**96.78**	93.07	87.98	96.29
(2000,2)	**96.57**	83.67	80.84	95.08
(4000,1)	**96.31**	95.45	89.67	95.68
(4000,2)	**94.54**	88.42	81.91	93.23
(6000,1)	**96.79**	96.49	89.78	96.44
(6000,2)	94.78	95.71	83.29	**96.62**
(8000,1)	**97.24**	96.18	91.02	96.48
(8000,2)	**97.06**	96.08	85.21	96.82

are applied based on the stream prequential test-then-train mechanism [37] and computed using Equation 5 where TP, FP, TN and FN denote true positives, false positives, true negatives, and false negatives, respectively.

$$\text{Accuracy Rate} = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{5}$$

Tables 1, 2, and 3 present the results of comparing the performance of the proposed algorithm with the other three algorithms in our framework using the average classification accuracy metric. The number of instances in each stream is selected to be $NC * \tau_t$, where NC is the number of environmental changes and τ_t is the frequency of change.

TABLE III
COMPARING THE CLASSIFICATION ACCURACY OF ALGORITHMS FOR THE SEAFD DATASET

(τ_t, n_t)	Proposed	HAT	VFDT	DFBFS
(1000,1)	**92.91**	82.92	78.21	86.79
(1000,2)	**91.26**	75.21	74.23	89.65
(2000,1)	**93.06**	87.42	80.27	91.37
(2000,2)	**93.18**	78.75	75.88	92.68
(4000,1)	**95.83**	90.03	81.26	91.55
(4000,2)	**94.43**	82.49	77.81	93.08
(6000,1)	**94.57**	91.44	82.72	92.96
(6000,2)	**95.87**	90.31	79.46	93.43
(8000,1)	**94.81**	91.03	83.49	92.44
(8000,2)	**96.03**	90.72	81.51	93.78

TABLE IV
COMPARING THE AVERAGE EXECUTION TIME OF ALGORITHMS IN SECONDS

Dataset	n_t	Proposed	HAT	VFDT	DFBFS
BGFD1	1	4.18	3.37	3.11	11.82
	2	4.01	3.37	3.11	11.82
BGFD2	1	4.17	3.34	3.05	11.34
	2	3.91	3.34	3.05	11.34
SEAFD	1	4.45	4.73	4.52	11.45
	2	4.15	4.73	4.52	11.45

In the proposed algorithm, the population size is set to 50 and the number of generations is dynamically fixed based on the stability of non-dominated solutions as described in the previous section. The number of features/variables is selected to be 10 for all tested data streams. The proposed algorithm is implemented based on the NSGA-II algorithm which uses the bit-flip mutation and the single-point crossover and available in [38]. For the HAT and the VFDT algorithms, we used the implementation of the Massive Online Analysis (MOA) framework [37].

The statistical significance between results is computed using the Wilcoxon rank-sum test [39] with a 0.05 significance level, and the best values in each test instance are bolded. To test the performance of the compared algorithms in different scenarios, five frequency of change values and two levels of change severity are used for each dataset. In our experiments, the frequency of change is defined as the number of new arriving samples between two successive feature drifts, where the severity of change is defined as the number of relevant features that become irrelevant after feature drift occurs.

As illustrated in Table 1, the results of the BGFD1 dataset show that the proposed algorithm obtains higher performance than the other algorithms in 6 out of 10 cases, where both of the HAT and the DFBFS algorithms obtain the best result in two cases. The proposed algorithm shows a good performance of adaption for slight changes where only one feature becomes irrelevant. This observation can be explained by the dynamic diversity adaptation mechanism in the proposed algorithm that maintains the population diversity during the search process. It can balance between the newly introduced solutions and the level of severity for the occurred feature drift which prevents the degradation of the population in the case of slight changes. By examining the results of Table 1, we can also observe that with more slow changes, the performance difference between our algorithm and other algorithms becomes smaller and smaller. This is due to the sufficient available time in the case of slow changes (higher values of frequency of change) which gives all algorithms enough time to adapt to the new environment.

Table 2 shows the results of the BGFD2 dataset, where the proposed algorithm obtains the best results in the majority of tested cases. As in the results of Table 1, the proposed algorithm shows better performance than other algorithms in the fast environmental changes (low values of frequency of

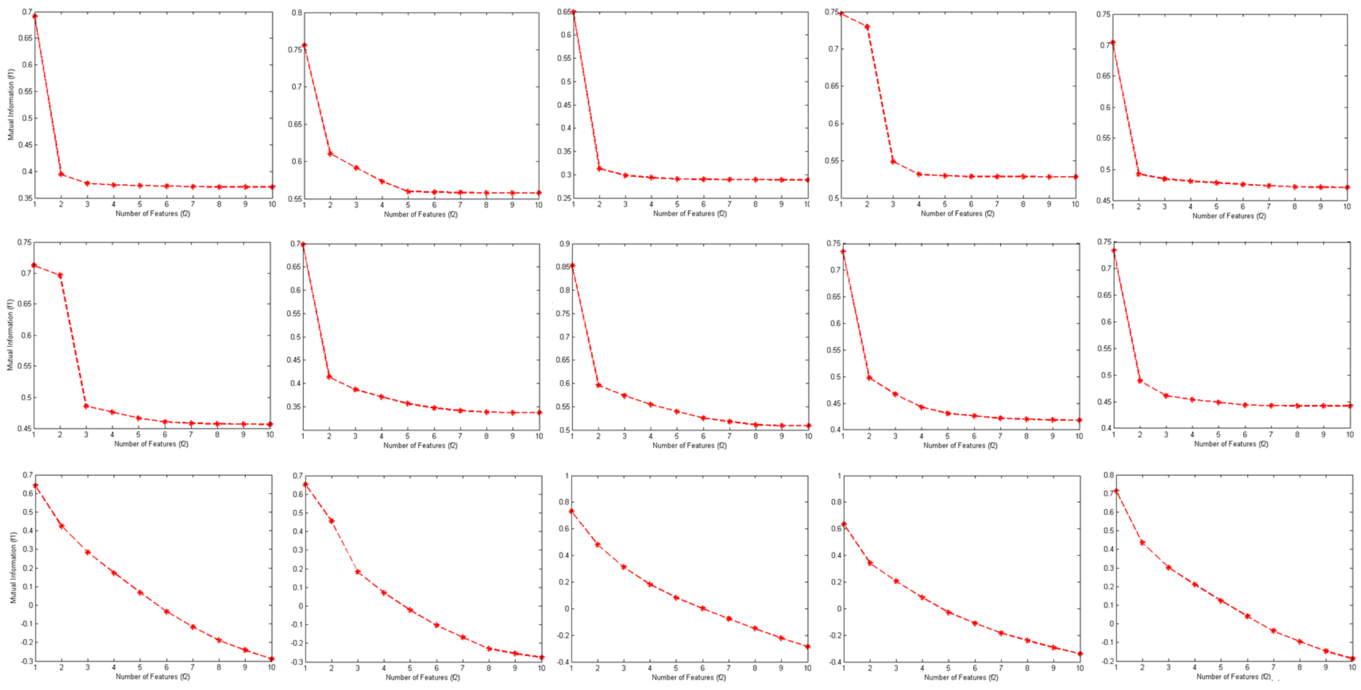


Fig. 2. Pareto front evolved by the proposed algorithm for five successive changes and using three datasets (the first row is for the BGFD1 dataset, the second row is for the BGFD2 dataset, and the third row is for the SEAFD dataset).

change), and its performance becomes close to the HAT and the DFBFS algorithms in slow changes. The results of Table 3 (SEAFD dataset) are similar to the results of Table 2 since the proposed algorithm achieves the best results in all tested cases. These results show the direct effect of the dynamic diversity introduction and memory mechanism on the performance of the proposed algorithm by making it converges rapidly to a diverse set of solutions. Furthermore, the proposed algorithm performance remains stable and does not affect by the level of change severities between the old and the new environments. Figure 2 shows the evolved Pareto front from our proposed algorithm using the three tested datasets for five successive changes. As shown in the figure, the Pareto front after each feature drift changes slightly or significantly, which indicates that this dynamic problem is Type 2 since both POS and POf change. Additionally, we can note that the shape of Pareto fronts is different for each dataset.

In online data streams, the processing time is another important metric that should be minimized due to the limited available time during the arrival of stream samples. Table 4 presents the comparison between algorithms in our framework by considering the average execution time for each algorithm. While the VFDT algorithm achieves the lowest execution time in all cases, the proposed algorithm shows acceptable results. Furthermore, we can observe that the proposed algorithm achieves a significant enhancement over the DFBFS algorithm, which may reflect the rapid convergence of proposed algorithm after adding the dynamic change response and memory mechanisms. Moreover, using a dynamic stopping criterion

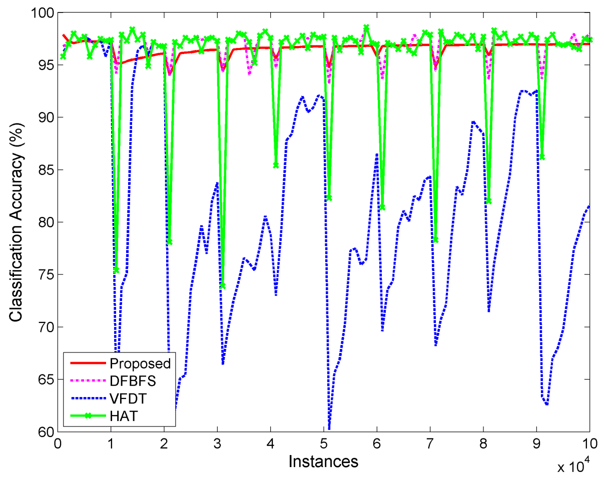


Fig. 3. Average window accuracy for compared algorithms using the BGFD1 dataset.

instead of using a constant number of generations during the evolving process significantly decreases the number of executed generations after each change occurrence.

In another experiment, the algorithms are compared using the window accuracy of sample classification, where results are given in Figures 3, 4, and 5. The number of samples is set to 100000 for each dataset, and a feature drift occurs after every 10000 samples. To avoid the degradation occurred in the first generations and to test the performance of our algorithm

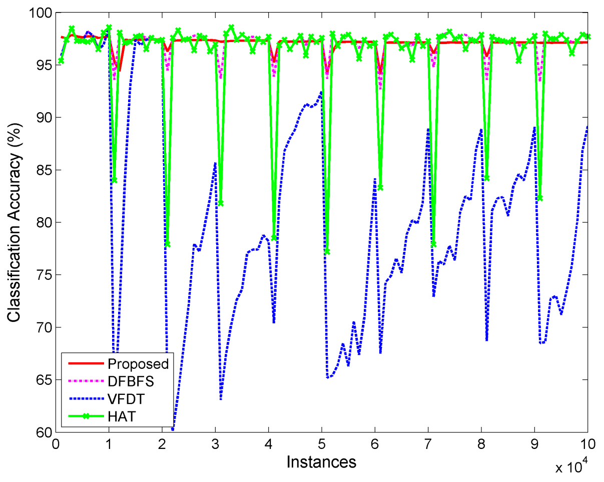


Fig. 4. Average window accuracy for compared algorithms using the BGFD2 dataset.

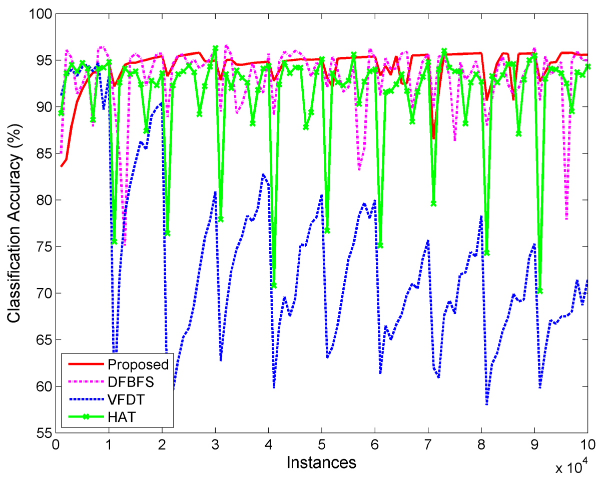


Fig. 5. Average window accuracy for compared algorithms using the SEAFD dataset.

after the memory becomes full, we first run our algorithm for 200 generations, and then start the evaluation process. As it is expected, the classification accuracy of all algorithms drops after the occurrence of feature drifts. However, the differences between algorithms occur in two factors, the extent of accuracy degradation, and the ability of rapid recovery after each feature drift. Our proposed algorithm and the DFBFS algorithm show rapid recovery and lower performance degradation than the HAT and the VFDT algorithms. The VFDT algorithm is the worst algorithm since it spends more time to recover its performance than other algorithms. This result ensures the active impact of the dynamic severity introduction and the explicit memory scheme that incorporated with our dynamic multi-objective evolutionary algorithm.

V. Conclusion

In this paper, we proposed an enhanced dynamic multi-objective evolutionary algorithm for the dynamic feature selection problem in data streams. The proposed DMOEA continuously searches for the set of relevant features that change over time after each environmental change (i.e. feature drift). We incorporated the proposed DMOEA with a dynamic severity introduction mechanism, and a memory scheme, and an artificial neural network. A detection mechanism for severity of changes is utilized to estimate the extent of feature drift and adaptively respond to it by adding a suitable number of random solutions to the population. The memory is used to store the good solutions and reuse them after each change to accelerate the convergence of solutions in the case of cyclic or repeated feature drifts. The Artificial Neural Network (ANN) model is incorporated with our proposed DMOEA to classify the new arriving samples of the stream. The obtained results for three datasets by using different environmental parameters demonstrate the validity of the proposed algorithm for feature drift problems in data steams. In our future work, we plan to test the other alternatives in filter-based feature selection algorithms and measure their performance for solving dynamic feature selection problem.

References

[1] S. Yang and X. Yao, *Evolutionary Computation for Dynamic Optimization Problems*. Springer-Verlag, 2013.

[2] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[3] C.-L. Chen and W.-C. Lee, "Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices," *Computers & Chemical Engineering*, vol. 28, no. 6, pp. 1131–1144, 2004.

[4] G. Valentini, C. J. B. Abbas, L. G. Villalba, and L. Astorga, "Dynamic multi-objective routing algorithm: a multi-objective routing algorithm for the simple hybrid routing protocol on wireless sensor networks," *IET communications*, vol. 4, no. 14, pp. 1732–1741, 2010.

[5] S. Palaniappan, S. Zein-Sabatto, and A. Sekmen, "Dynamic multiobjective optimization of war resource allocation using adaptive genetic algorithms," in *SoutheastCon 2001. Proceedings. IEEE.* IEEE, 2001, pp. 160–165.

[6] R. P. Hämäläinen and J. Mäntysaari, "Dynamic multi-objective heating optimization," *European Journal of Operational Research*, vol. 142, no. 1, pp. 1–15, 2002.

[7] K. Deb, U. Bhaskara Rao N., and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified nsga-ii: a case study on hydro-thermal power scheduling," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 803–817.

[8] D. Constantinou, "Ant colony optimisation algorithms for solving multi-objective power-aware metrics for mobile ad hoc networks," Ph.D. dissertation, University of Pretoria, 2011.

[9] M. Liu, J. Zheng, J. Wang, Y. Liu, and L. Jiang, "An adaptive diversity introduction method for dynamic evolutionary multiobjective optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress*. IEEE, 2014, pp. 3160–3167.

[10] J. J. Grefenstette *et al.*, "Genetic algorithms for changing environments," in *PPSN*, vol. 2, 1992, pp. 137–144.

[11] Y. Wang and B. Li, "Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 630–637.

[12] S. Sahmoud and H. R. Topcuoglu, "A memory-based nsga-ii algorithm for dynamic multi-objective optimization problems," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 296–310.

[13] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 832–846.

[14] A. Muruganantham, Y. Zhao, S. B. Gee, X. Qiu, and K. C. Tan, "Dynamic multiobjective optimization using evolutionary algorithm with kalman filter," *Procedia Computer Science*, vol. 24, pp. 66–75, 2013.

[15] M. Greeff and A. P. Engelbrecht, "Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 2917–2924.

[16] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 103–127, 2009.

[17] S. Sahmoud and H. R. Topcuoglu, "A general framework based on dynamic multi-objective evolutionary algorithms for handling feature drifts on data streams," *Future Generation Computer Systems*, vol. 102, pp. 42–52, 2020.

[18] S. Muthukrishnan, "Data streams: Algorithms and applications," *Foundations and Trends in Theoretical Computer Science*, vol. 1, no. 2, pp. 117–236, 2005.

[19] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Dynamic feature space and incremental feature selection for the classification of textual data streams," *Knowledge Discovery from Data Streams*, pp. 107–116, 2006.

[20] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 97–106.

[21] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[22] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, "A survey on feature drift adaptation: definition, benchmark, challenges and future directions," *Journal of Systems and Software*, vol. 127, pp. 278–294, 2017.

[23] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, and L. Wan, "Heterogeneous ensemble for feature drifts in data streams," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2012, pp. 1–12.

[24] B. Xue, L. Cervante, L. Shang, W. N. Browne, and M. Zhang, "Multi-objective evolutionary algorithms for filter based feature selection in classification," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 04, p. 1350024, 2013.

[25] G. Sun, J. Li, J. Dai, Z. Song, and F. Lang, "Feature selection for iot based on maximal information coefficient," *Future Generations Computer Systems*, vol. 89, no. 1, pp. 606–616, 2018.

[26] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[27] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Future Generations Computer Systems*, vol. 36, pp. 156–169, 2014.

[28] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection: An ever evolving frontier in data mining," in *Feature Selection in Data Mining*, 2010, pp. 4–13.

[29] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

[30] S. Sahmoud and H. R. Topcuoglu, "Exploiting characterization of dynamism for enhancing dynamic multi-objective evolutionary algorithms," *Applied Soft Computing*, vol. 85, p. 105783, 2019.

[31] J. P. Barddal, H. M. Gomes, and F. Enembreck, "Analyzing the impact of feature drifts in streaming learning," in *International Conference on Neural Information Processing*. Springer, 2015, pp. 21–28.

[32] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.

[33] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 377–382.

[34] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2000, pp. 71–80.

[35] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *International Symposium on Intelligent Data Analysis*. Springer, 2009, pp. 249–260.

[36] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of The 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 443–448.

[37] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[38] K. Deb, "Kanpur genetic algorithms laboratory," *Multi-objective NSGA-II code in C. Original Implementation (for Windows and Linux)*, 2009.

[39] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.