2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

# Cooperative DynDE for Temporal Data Clustering

Kristina S. Georgieva and Andries P. Engelbrecht

*Abstract*— **Temporal data is common in real-world datasets. Clustering of such data allows for relationships between data patterns over time to be discovered. Differential evolution (DE) algorithms have previously been used to cluster temporal data. This paper proposes the cooperative data clustering dynamic DE algorithm (CDCDynDE), which is an adaptation to the data clustering dynamic DE (DCDynDE) algorithm where each population searches for a single cluster centroid. The paper applies the proposed algorithm to a variety of temporal datasets with different frequencies of change, severities of change, dataset dimensions and data migration types. The clustering results of the cooperative data clustering DynDE are compared against the original data clustering DynDE, the re-initialising data clustering DE and the standard data clustering DE. A statistical analysis of these results shows that the cooperative data clustering DynDE algorithm obtains better data clustering solutions to the other three algorithms despite changes in frequency, severity, dimension and data migration types.**

## I. Introduction

Data clustering is a task that aims to determine relationships among objects by grouping similar objects together [1]. This task has been previously performed with a variety of computational intelligence (CI) algorithms, including neural networks [2], swarm intelligence algorithms [3][4][5], artificial immune systems [6][7] and evolutionary algorithms [8][9].

Temporal data [10], which is data that changes with time, introduces additional complexities for data clustering algorithms. The changing data results in the clustering solution differing from one time-step to another, requiring the algorithms to track the new centroids. Changes in the environment, which refer to a change in the constitution of the clusters within that environment, may include clusters becoming larger, smaller, appearing, disappearing, moving, and patterns that migrate to other clusters. These changes make it difficult for algorithms to cluster data due to the loss of diversity and outdated memory [11] suffered by the population-based algorithms. The loss of diversity refers to individuals in the population of an algorithm converging to a solution, making the population less diverse. This phenomenon is problematic in temporal environments as few or no individuals explore the search space for better solutions when the environment changes. Outdated memory refers to the optimal values found by an algorithm's population becoming obsolete as the environment changes.

Few computational intelligence algorithms for clustering temporal data have previously been implemented. Examples of such algorithms include the local network neighbourhood artificial immune system [6], re-initialising data clustering PSO [12], multi-population data-clustering PSO [12], cooperative-multipopulation data-clustering PSO [12], re-initialising data clustering DE [13], data clustering dynamic DE (DCDynDE) [13] and a combination of self-organising maps (SOMs) and autoregressive integrated moving average (ARIMA) time-series models [14].

This paper focuses on differential evolution (DE) approaches to cluster temporal data. Georgieva and Engelbrecht [13] proposed two adaptations to Hanuman et al's [9] data clustering DE in order to handle temporal data. The first algorithm proposed was the re-initialising data clustering DE (RDCDE), which re-initialises part of a population when a change occurs. The second algorithm proposed was the data clustering dynamic DE (DCDynDE), which uses repulsion and Brownian individuals in order to re-diversify the population. This paper proposes the cooperative data clustering DynDE (CDCDynDE), which adapts the DCDynDE by making each population search for a single optimum instead of for all optimums in parallel to the other populations. This allows for various areas of the search space to be exploited for clusters, while the repulsion and Brownian individuals promote exploration in the temporal environment.

An empirical analysis of the CDCDynDE in comparison with Hanuman et al's [9] data clustering DE, the RDCDE and the DCDynDE is done in this paper. The results show that the CDCDynDE performed the clustering task more effectively than the other algorithms. The algorithm's performance remained good despite changes in frequency, severity, dimension and pattern migration types.

The article begins with Section II summarises the DE, Hanuman et al's [9] data clustering DE, RDCDE and DynDE in Section. These summaries are followed by the proposed CDCDynDE algorithm in Section III, the experimental set-up in Section IV and the results of the performed experiments in Section V. Lastly, Section VI summarises the article's findings along with possible future work.

## II. Differential Evolution

The DE is a population-based evolutionary algorithm (EA) that has been used to solve a wide range of optimization problems [13][15][16][17]. The DE, like other EAs makes use of evolutionary operators to adapt individuals. These operators are mutation, crossover, and selection. DE differs from other EAs by applying these operators in a different order, as well as introducing diversity via the mutation of each parent individual and producing a trial vector. Mutational step sizes are based on distances that randomly selected individuals are from one another. The trial vector is recombined with the parent individual, using a discrete crossover operator, to produce a single offspring. The offspring then competes with the parent individual, through a selection process, to survive to the next generation.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

The mutation operator creates a trial vector for each parent individual $\mathbf{x}_i$ using

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) \quad (1)$$

where $\mathbf{x}_{i_1}(t)$ is the target vector, which in this paper is a randomly selected individual, $\mathbf{x}_{i_2}(t)$ and $\mathbf{x}_{i_3}(t)$ are two randomly selected individuals used to create a difference vector with $i_1 \neq i_2 \neq i_3$, and $\beta$ is a constant, called the scaling factor.

The crossover operator implements a discrete recombination of the parent individual and the trial vector using

$$\mathbf{x}'_{ij}(t) = \begin{cases} \mathbf{u}_{ij} & \text{if } j \text{ is } \in J \\ \mathbf{x}_{ij}(t) & \text{if } j \text{ is } \notin J \end{cases} \quad (2)$$

where $\mathbf{u}_i$ is the trial vector and $J$ is a set of crossover points. The crossover points are determined differently depending on whether exponential or binomial crossover is used. For binomial crossover a crossover probability is used to randomly determine which points are to be crossed over. For exponential crossover an index is randomly selected and a sequence of adjacent indexes are chosen to be crossed over. The algorithms described in this paper use binomial crossover.

An elitist selection operator is used to accept the best individual between the parent and offspring to survive to the next generation.

DE has been developed to solve continuous-valued optimization problems, and has been shown to be very efficient [18][19][20][21]. Data clustering can be defined as an optimization problem, where the objective is to find cluster centroids that maximise the compactness and separation of clusters. Since data clustering is in effect an optimization problem, DE can be applied to find optimal cluster centroids.

Hanuman et al [9] adapted the DE in order to solve data clustering problems, referred to in this paper as the standard data clustering DE (SDCDE) algorithm. The first change to the algorithm is the representation of an individual. In the DE algorithm an individual is represented as an $n$-dimensional vector, where $n$ is the dimensionality of the problem. The solution to a clustering problem, however, is a group of cluster centroids. Each cluster centroid is an $n_x$-dimensional vector. Therefore the dimension of a SDCDE individual is $n = n_x * n_K$, where $n_K$ is the number of cluster centroids.

The second change to the DE algorithm is the use of the dataset to guide the search. Before the fitness of an individual is calculated, each data pattern is assigned to the cluster centroid closest to it, which is determined using the Euclidean distance measure. The fitness is then calculated using the distances between data patterns and the centroids that the patterns are assigned to. This paper uses the quantization error as the fitness measure, which is calculated using [22]

$$J_e = \begin{cases} \frac{\sum_{k=1}^{n_K} \frac{\sum_{\forall \mathbf{z}_p \in C_k} d(\mathbf{z}_p, \mathbf{c}_k)}{|C_k|}}{n_K} & \text{if } |C_k| > 0 \\ \infty & \text{if } |C_k| = 0 \end{cases} \quad (3)$$

where $\mathbf{z}_p$ is a data pattern, $C_k$ is a cluster, $\mathbf{c}_k$ is the cluster centroid of $C_k$ and $|C_k|$ is the total number of data patterns in cluster $C_k$.

DE algorithms suffer from loss of diversity in temporal environments [23], rendering it inefficient to cluster temporal data unless a mechanism is incorporated to re-diversify the population. Georgieva and Engelbrecht [13] proposed two adaptations to the SDCDE:

- The re-initialising data clustering DE (RDCDE) adapts the DCDE by re-initialising a percentage of the population if a change in the environment occurs. The re-initialisation places a percentage of the population at randomly selected positions in the search space, allowing re-initialised individuals to explore the environment for new solutions.
- The data clustering dynamic DE (DCDynDE) combines the DCDE with Mendes and Mohais' [24] DynDE algorithm. The DynDE is a multi-population DE that adds Brownian individuals and exclusion to the DE in order to increase the diversity of the population. The DynDE selects a percentage of the weakest individuals in a sub-population to become Brownian individuals during each iteration. Brownian individuals are individuals generated by adding Gaussian noise to the position of best individual of a sub-population. Additionally, if two sub-populations are within a certain radius of each other, the weakest sub-population is excluded by being re-initialised.

  The DCDynDE adapts the DynDE by changing the representation of the individuals to contain a group of centroid positions. Additionally, the data patterns are assigned to their closest centroid and the fitness of each solution is determined using the quantization error defined in Equation (3).

The datasets used are automatically generated datasets with known characteristics. The time-step when a change occurs is, therefore, priorly known and was provided to the RDCDE. The DCDynDE, on the other hand, is capable of tracking changes in the environment through repulsion and Brownian individuals, requiring no prior knowledge of the time-step when a change occurs.

## III. Cooperative Data Clustering Dynamic Differential Evolution

The DCDynDE [13] algorithm consists of multiple populations working in parallel, where the final solution is the best global best over all sub-populations. This section proposes an adaptation of the DCDynDE, by letting each sub-population search for a single centroid instead of all of the centroids. By doing this, the task of each sub-population is reduced to finding an $n_x$-dimensional vector instead of an $n_x * n_K$-dimensional solution, focusing the search. This approach is similar to the cooperative PSO developed by Van den Bergh and Engelbrecht [25].

The CDCDynDE algorithm initialises $n_K$ sub-populations of $n_x$-dimensional individuals, where $n_K$ is the total number

of clusters. The quantization error requires knowledge of all cluster centroid positions in order to determine the suitability of a clustering solution. By making each population search for a single centroid position, this knowledge is lost. In order to calculate the fitness of an individual, CDCDynDE introduces a context individual, inspired by the cooperative PSO [25]. A context individual is a $n_x * n_K$-dimensional individual comprised of the best $n_x$-dimensional centroids from each sub-population.

To calculate the fitness of a solution belonging to sub-population $S_k$, the dimension $k$ of the context individual is replaced with that solution. The remainder of the context individual consists of the best solutions of the other populations. The fitness of this adapted context individual is then calculated and assigned as the fitness of the solution.

During each iteration of the CDCDynDE the distance between the global best solutions of each sub-population is calculated. If the distance between two sub-populations is smaller than a pre-defined exclusion radius, $r_{excl}$, the weakest of the two sub-populations is re-initialised. The re-initialisation involves placing all individuals within the sub-population at random positions in the search space. The re-initialised sub-population therefore restarts its search for an optimal centroid. This exclusion mechanism, taken from the DCDynDE algorithm, allows for each sub-population to find a different centroid and promotes separability between clusters.

After the exclusion has taken place, the sub-populations that were not re-initialised are adapted using the standard mutation, crossover and selection DE mechanisms. A percentage, $b_{perc}\%$, of the weakest individuals in each sub-population are then selected to become Brownian individuals. These Brownian individuals are adapted using

$$w_{ij} = y_{ij} + N(0, \sigma) \tag{4}$$

where $\mathbf{w}_{ij}$ is the new value for dimension $i$ of the Brownian individual's centroid at index $j$, $\mathbf{y}_{ij}$ is the value for dimension $i$ of the global best individual's centroid at index $j$ and $N(0, \sigma)$ is Gaussian noise with mean 0 and standard deviation $\sigma$.

The CDCDynDE is summarised in Algorithm 1.

## IV. Experimental Setup

Auto-generated temporal datasets provided by Graaff [6] were used. The datasets include three data migration types, namely pattern migration, cluster migration and centroid migration. For pattern migration a single data pattern migrates from the cluster to which it belongs to a randomly selected cluster. For cluster migration all data patterns in a selected cluster migrate to other randomly selected clusters. For centroid migration all data patterns of a cluster migrate to new positions where the patterns still belong to the same cluster. Clusters appear, disappear, grow and shrink in the various datasets.

The datasets consisted of 8000 patterns, 100 time-steps, 80 data patterns per time-step and a maximum of eight clusters at any point in time. Patterns consisted of either three, eight or fifteen attributes. Frequencies, which refer to the rate at which changes occur, ranged from one to five, where a larger number implies less changes. The severities of change, which refer to the magnitude of the change, ranged from one to five. The iteration at which a change occurs was calculated as $\frac{f}{10} * T$, where $f$ is the frequency of change, and $T$ is the total number of iterations. A total of 225 artificial datasets were clustered as there are three migration types, three dimensions, five frequencies and five severities.

**Algorithm 1** CDCDynDE

```
Initialise S sub-populations of individuals
while stopping condition is not reached do
    Generate a context individual by replacing each dimension with the best solution of each population respectively, and denote as b(k, S_k.ŷ_i), where k is the population index and S_k.ŷ_i is the best solution for that population.
    for each sub-population S_k do
        for each individual S_k.x_i do
            Replace dimension k of b(k, S_k.ŷ_i) with S_k.x_i to create b(k, S_k.x_i)
            for each data pattern z_p do
                Calculate the Euclidean d(z_p, S_k.x_{ik_j}) of z_p and each cluster centroid S_k.x_{ik_j} of b(k, S_k.x_i)
                Assign pattern z_p to centroid S_k.x_{ik_j}, such that d(z_p, S_k.x_{ik_j}) = min_{∀k_j=1...N_{k_j}} {d(z_p, S_k.x_{ik_j})}
            end for
            Calculate the fitness of b(k, S_k.x_i)
        end for
    end for
    Compare the global best individuals from each sub-population to each other
    if the global best positions of both sub-populations are within r_excl of each other then
        re-initialise the sub-population with the worst global best
    else
        for each sub-population that was not re-initialised do
            for each individual S_k.x_i do
                Update S_k.x_i using the mutation, crossover and selection mechanisms of the DE
            end for
            Update context individual
            Make a percentage of the weakest individuals Brownian Individuals and update them using Equation (4)
        end for
    end if
end while
```

A population of 50 individuals was initialised within the bounds of the dataset, which are defined per dimension by

the lowest and highest value of each attribute in the dataset. Algorithms were run for a total of 1000 iterations and results were averaged over 30 independent runs. Centroids leaving the boundaries of the search space were re-set to remain on the boundary. Ten percent of the population was re-initialised in the RDCDE and 10% of the population was selected to become Brownian individuals in both the DCDynDE and the CDCDynDE. An exclusion radius of 5.0 was used along with a scaling factor and crossover probability of 0.5. The DCDynDE and CDCDynDE both used a total of eight populations as the maximum number of clusters in the datasets at any point in time was eight.

The quality of the clustering solution was measured using three measures [6], namely the inter-cluster distance, the intra-cluster distance and the Ray-turi validity index. The inter-cluster distance refers to the distance between clusters, which is maximised and calculated using

$$J_{iter} = \frac{2}{n_K(n_K-1)} \sum_{k=1}^{n_K-1} \sum_{k_2=k+1}^{n_K} d(\mathbf{c}_k, \mathbf{c}_{k_2}) \quad (5)$$

where $\mathbf{c}_k$ and $\mathbf{c}_{k_2}$ are cluster centroids.

The intra-cluster distance refers to the distance between data patterns within a cluster. This distance is minimised and calculated using

$$J_{intra} = \frac{\sum_{k=1}^{n_K} \sum_{\forall \mathbf{z} \in C_k} d(\mathbf{z}_p, \mathbf{c}_k)}{|P|} \quad (6)$$

where $|P|$ is the total number of data patterns in the dataset.

The last measure is the Ray-turi validity index. This measure combines the inter-cluster and intra-cluster distances for a more accurate representation of the quality of the clustering solution. The Ray-turi validity index is minimised and calculated using

$$Q_D = \frac{J_{intra}}{inter_{min}} \quad (7)$$

where $inter_{min}$ is the smallest inter-cluster distance found using Equation (5).

Wins and losses per algorithm based on the Ray-turi validity index were determined based on an approach proposed by Helbig and Engelbrecht [26]. For each sample, the Ray-turi validity index of all iterations before a change to the environment occurred was averaged. This gives a more accurate representation of the performance of the algorithm over time than only taking the last iteration's value. For each problem each algorithm's performance was compared against each other algorithm using 30 independent samples. First a Kruskal-Wallis test was performed to determine if there is a statistically significant difference between the algorithms' performance. If there was a statistically significant difference, pair-wise Mann-Whitney U tests were performed and the resulting U-values were used to determine the winning and losing algorithm for a particular problem.

## V. Results and Discussion

The average inter-cluster distance, intra-cluster distance and Ray-turi validity index values are reported in Table I. The table shows that the CDCDynDE obtained the highest inter-cluster distance and, therefore, had the most separated clusters. This result has a large standard deviation value, implying that the resulting inter-cluster distances varied significantly for the problems used. The DCDynDE, on the other hand, obtained the lowest inter-cluster distance making its resulting clusters the least separate. Lastly, the SDCDE and RDCDE obtained similar values for the inter-cluster distance.

The DCDynDE obtained the lowest intra-cluster distance and, therefore, the most compact clusters. The CDCDynDE, SDCDE and RDCDE clusters resulted in similar values for the intra-cluster distance, with the RDCDE obtaining the highest value. Once again, CDCDynDE has a large standard deviation, implying a significant difference between the resulting intra-cluster distances for the problems used.

The Ray-turi validity measures the performance of the algorithm by combining the inter-cluster and intra cluster distance measures. This is a more accurate measure because a clustering solution is required to have clusters that are very compact and highly separated from each other. The CDCDynDE obtained the lowest Ray-turi validity index, indicating that the algorithm performed the clustering task most effectively. The variation of the Ray-turi validity index across all the problems is much less significant than that of the inter-cluster and intra-cluster distances for the CDCDynDE algorithm. The DCDynDE performed the clustering task better than the SDCDE and RDCDE, but worse than the CDCDynDE.

Table II summarises the total statistically significant wins, $\#wins$, and losses, $\#losses$, obtained by each algorithm as well as the $Diff$ value, which is calculated using $Diff = \#wins - \#losses$. The CDCDynDE obtained the most wins and least losses when compared against the other algorithms. The DCDynDE obtained 70 more losses than wins, while the SDCDE and RDCDE both obtained significantly more losses than wins. These results are illustrated in Figure 1 for a visual interpretation.

The problems used consisted of various severities, frequencies, dimensions and pattern migration types. Figures 2-5 illustrate the behaviour of each algorithm as the various parameters change.

Figure 2 illustrates $Diff$ of each algorithm for dimensions 3, 8 and 15. For the three dimensional problems the CDCDynDE and the DCDynDE both have a high $Diff$ and, therefore, significantly more wins than losses. The SDCDE and RDCDE, on the other hand, obtained significantly more losses than wins for three-dimensional problems. As the dimensions increase the performance of the DCDynDE abruptly decreases, while the performance of the SDCDE and RDCDE slightly increases. The CDCDynDE remains the best performing algorithm throughout, obtaining the most wins and least losses for all-dimensional problems, and is the only algorithm with a positive $Diff$ value throughout.

TABLE I

AVERAGES AND STANDARD DEVIATION FOR INTER-CLUSTER DISTANCE, INTRA-CLUSTER DISTANCE AND RAY-TURI VALIDITY FOR EACH ALGORITHM

| **Algorithm** | **Inter-cluster Distance** | **Intra-cluster Distance** | **Ray-Turi Validity** |
|---|---|---|---|
| SDCDE | 22.983 $\pm$ 3.266 | 10.679 $\pm$ 3.981 | 1.141 $\pm$ 0.178 |
| RDCDE | 22.963 $\pm$ 3.257 | 10.682 $\pm$ 3.979 | 1.154 $\pm$ 0.198 |
| DCDynDE | 19.830 $\pm$ 3.346 | **3.877** $\pm$ **2.995** | 0.979 $\pm$ 0.207 |
| CDCDynDE | **30.072** $\pm$ **11.323** | 10.39107 $\pm$ 8.908 | **0.677** $\pm$ **0.276** |

TABLE II

TOTAL STATISTICALLY SIGNIFICANT WINS AND LOSSES PER ALGORITHM

| **Attribute** | **SDCDE** | **RDCDE** | **DCDynDE** | **CDCDynDE** |
|---|---|---|---|---|
| Wins | 203 | 204 | 300 | 632 |
| Losses | 466 | 466 | 370 | 37 |
| Wins - losses | -263 | -262 | -70 | 595 |

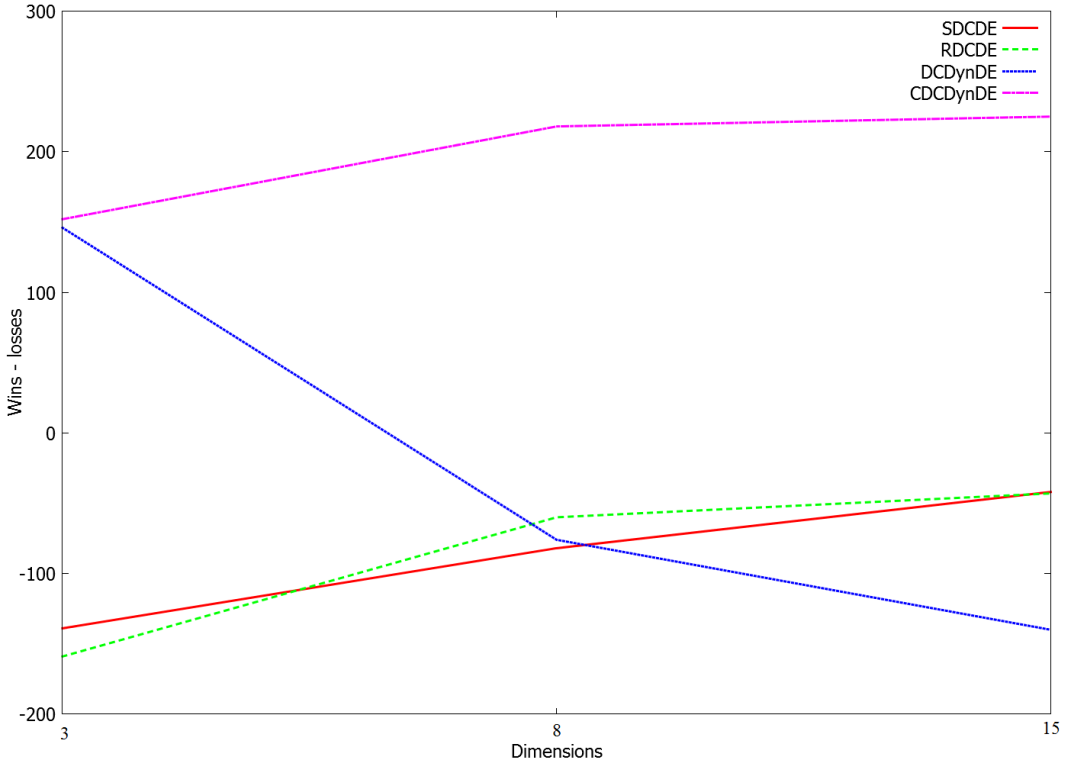


Fig. 2. $Diff$ per algorithm as dimensions change

Figure 3 illustrates $Diff$ of each algorithm for frequencies 1, 2, 3, 4 and 5, where a higher value means that less changes occur in the environment. The CDCDynDE obtained the highest $Diff$ value for all frequencies. The DCDynDE, on the other hand, obtained more losses than wins when less changes occurred in the environment and improved performance as the changes increased (with lower frequency values). The SDCDE and RDCDE both obtained more losses than wins for all frequencies, decreasing performance as more changes in the environment occurred.

Figure 4 illustrates $Diff$ of each algorithm for severities 1, 2, 3, 4 and 5. The CDCDynDE obtained the most wins and the least losses for all severities of change. The DCDynDE obtained increased performance on higher severities, but still remained with more losses than wins for the majority of the severities, while the CDCDynDE's performance decreased by a small amount.

Figure 5 illustrates the total wins and losses per algorithm for pattern migration, centroid migration and cluster migration problems. The CDCDynDE obtained the most wins and least losses over all three migration types. The DCDynDE, SDCDE and RDCDE all obtained more losses than wins for

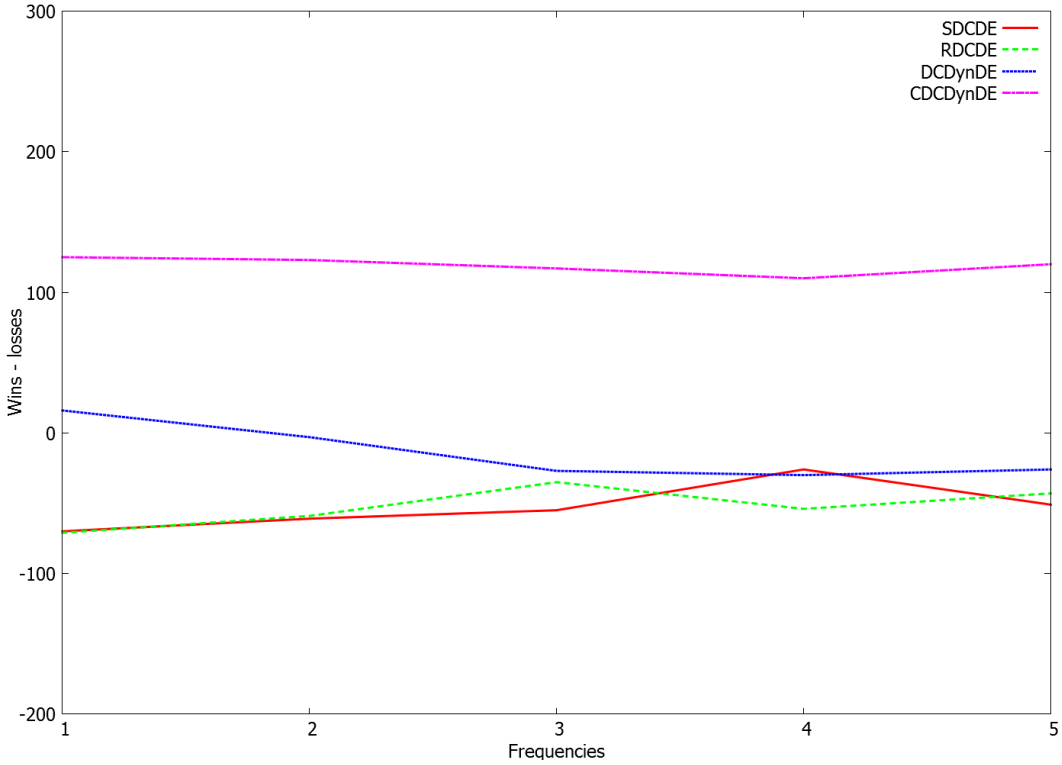


Fig. 3. $Diff$ per algorithm as frequencies change

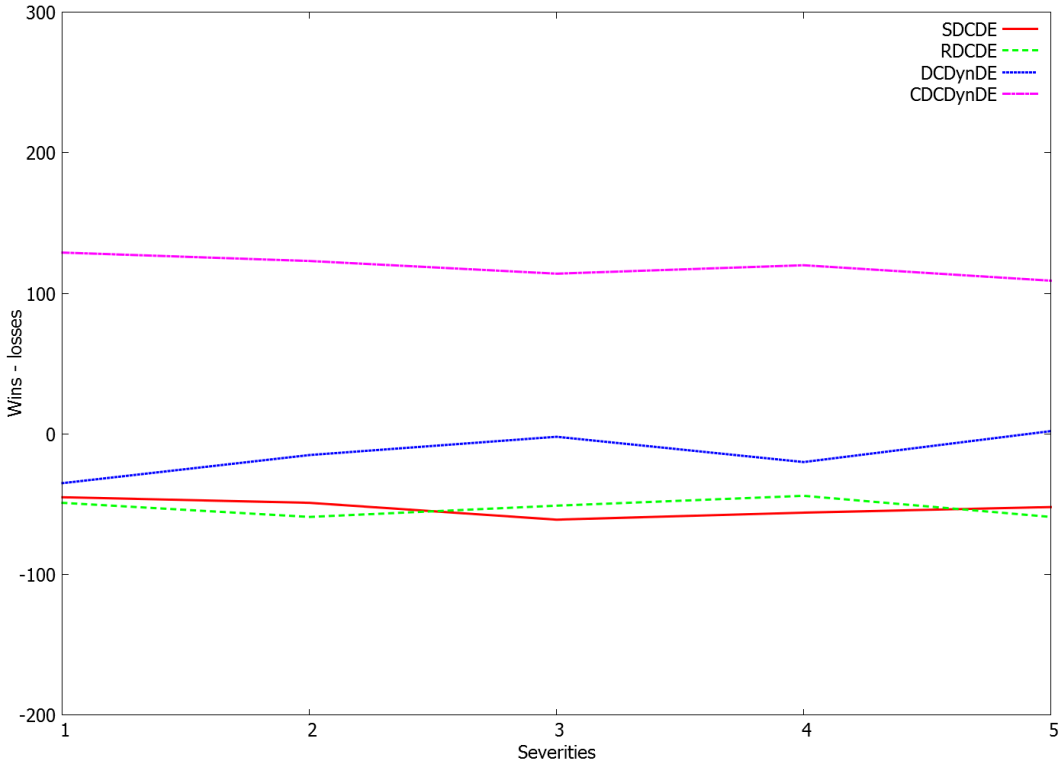


Fig. 4. $Diff$ per algorithm as severities change

pattern migration problems. The DCDynDE improved performance compared to the SDCDE and RDCDE when used on centroid migration and cluster migration problems, but still obtained more losses than wins due to the CDCDynDE

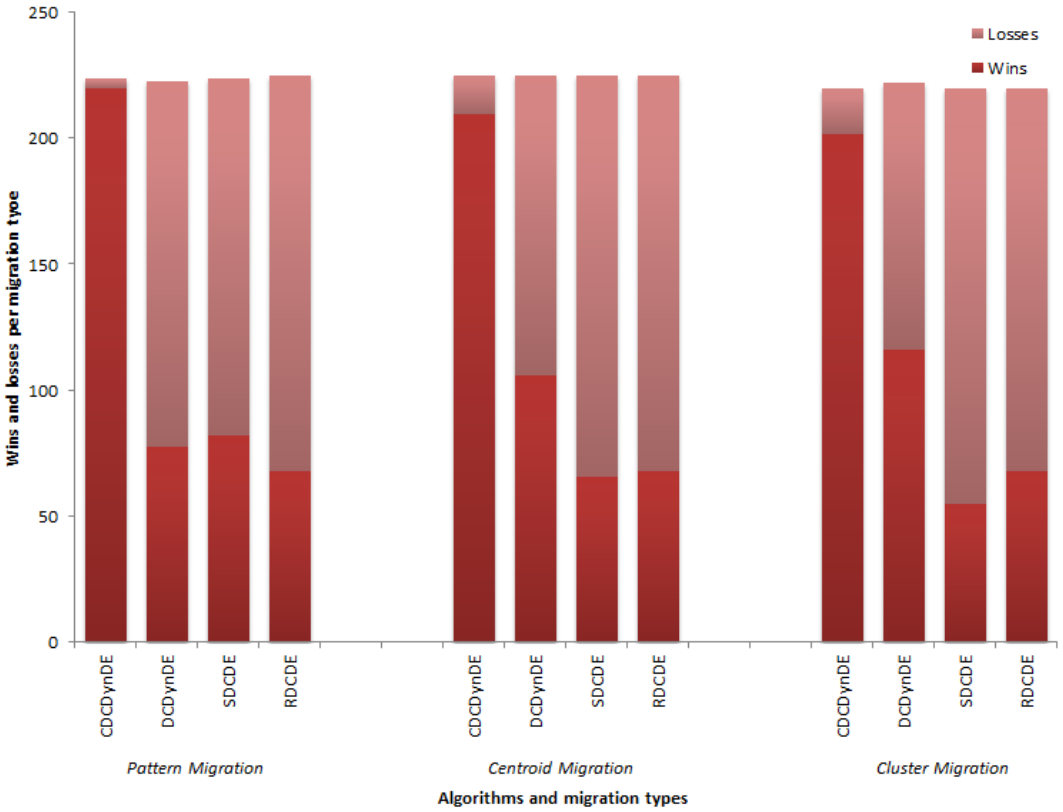


Fig. 5. Wins and losses per algorithm for different migration types

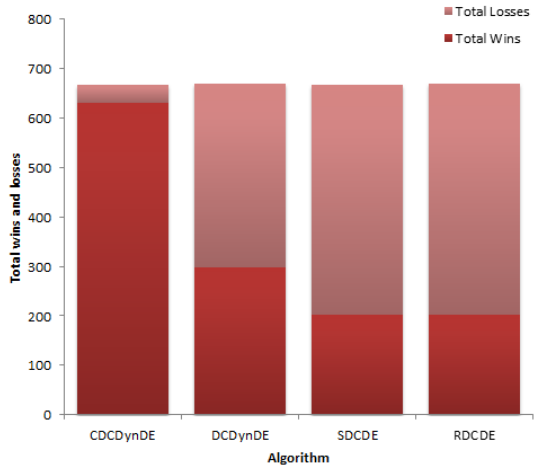


Fig. 1. Total wins and losses per algorithm over all problems

performing better on all three migration types.

## VI. Conclusions and Future Work

This article introduced the CDCDynDE, an adaptation to Georgieva and Engelbrecht's [13] DCDynDE. It evaluated the performance of the proposed algorithm against the DCDynDE, SDCDE and RDCDE applied to a variety of temporal datasets. The datasets included different severities of change, frequencies of change, dimensions and pattern migration types.

The average inter-cluster distance showed that the CDCDynDE obtained the most separable clusters, while the average intra-cluster distance showed that the DCDynDE obtained the most compact clusters. However, both measures need to be considered in order to determine the quality of a clustering solution. For this reason the Ray-turi validity index was used to analyse the quality of the clustering solutions in more detail. The average Ray-turi validity index value showed that the CDCDynDE performed the clustering task better than the DCDynDE, SDCDE and RDCDE.

Statistical tests were performed to determine the total wins and losses obtained by each algorithm when compared to each other algorithm. The Ray-turi validity index was used for these tests. The CDCDynDE obtained the most wins and least losses overall. When analysed in more detail it was shown that the CDCDynDE obtained the most wins and losses for all changes in severity, dimension, frequency and migration type.

Future work may include comparing the algorithm to other temporal data clustering algorithms, such as PSO, SOM and AIS. A more detailed study can be performed to analyse the algorithm's performance over various parameters such as frequency, severity and dimension. The CDCDynDE can also be adapted to allow it to optimize the number of clusters and can be applied to real-world datasets. Lastly, an analysis of the effect of regenerating the context individual when the best individual of the population is updated, instead of at the

beginning of every iteration, can be done.

## ACKNOWLEDGEMENT

The support of SAP P&I BIT Mobile Empowerment and the SARChI research chair in AI is hereby acknowledged. The views and conclusions expressed in this paper are those of the authors and are not necessarily to be attributed to the mentioned parties.

## REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, 2009.

[2] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns.," *Bioinformatics*, vol. 17, no. 2, pp. 126–136, 2001.

[3] D. W. Van Der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *2003 Congress on Evolutionary Computation 2003 CEC 03 (2003)*, vol. 1, pp. 215–220, 2003.

[4] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*, pp. 54–59, IEEE, 2009.

[5] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial bee colony (abc) algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 652–657, 2011.

[6] A. Graaff, *A Local Network Neighbourhood Artificial Immune System*. PhD thesis, University of Preotria, June 2011.

[7] L. N. de Castro and F. J. Von Zuben, "ainet: an artificial immune network for data analysis," *Data mining: a heuristic approach*, vol. 1, pp. 231–259, 2001.

[8] P. Scheunders, "A genetic c-means clustering algorithm applied to color image quantization," *Pattern Recognition*, vol. 30, no. 6, pp. 859–866.

[9] S. A. Hanuman, V. A. Babu, A. Govardhan, and S. C. Satapathy, "Data clustering using almost parameter free differential evolution technique," *International Journal of Computer Applications*, vol. 8, pp. 1–7, October 2010.

[10] J. Patel, "Temporal database system," Master's thesis, Department of Computing, Imperial College, University of London,, June 2003.

[11] T. Blackwell, "Particle swarm optimization in dynamic environments," *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 49, pp. 29–49, 2007.

[12] K. Georgieva and A. P. Engelbrecht, "A cooperative multi-population approach to clustering temporal data," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1983–1991, 2013.

[13] K. Georgieva and A. P. Engelbrecht, "Dynamic differential evolution algorithm for clustering temporal data," in *9th International Conference on Large-Scale Scientific Computations*, 2013.

[14] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[15] R. Storn, "On the usage of differential evolution for function optimization," in *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*, pp. 519–523, IEEE, 1996.

[16] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Differential evolution methods for unsupervised image classification," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2, pp. 966–973, IEEE, 2005.

[17] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[18] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, 2011.

[20] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[21] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[22] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, vol. 1, pp. 215–220, 2003.

[23] M. C. du Plessis and A. P. Engelbrecht, "Improved differential evolution for dynamic optimization problems," in *IEEE Congress on Evolutionary Computation IEEE World Congress on Computational Intelligence*, 2008.

[24] R. Mendes and A. S. Mohais, "Dynde: a differential evolution for dynamic optimization problems," in *IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2808–2815, 2005.

[25] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, 2004.

[26] M. Helbig and A. P. Engelbrecht, "Issues with performance measures for dynamic multi-objective optimisation," in *2013 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pp. 17–24, IEEE, 2013.