Comparative results between GenClust++ and Multi-objective GenClust++ algorithms

1. Multi-objective GenClust++

A multi-objective version of GenClust++ (MGenClust++) algorithm has been implemented and compared to the original GenClust++ clustering method showing promising results. MGenClust++ optimizes two objectives, connectivity and cohesion, instead of only one (i.e. Davies-Bouldin Index) utilized in the originally proposed algorithm.

The concept of MaxiMin strategy is applied to determine pareto optimal solutions. This strategy is utilized in chromosome selection operation, which merges all chromosomes between two populations: the last (most recent) population and the resulting generation from all operations. MaxiMin is used here to choose the best chromosomes from the merged populations for the next population. First, non-dominated solutions are chosen and added to the next population. At this point, next population may not be filled up completely, and therefore weakly dominated solutions are picked randomly and added to the next population until it's filled up. The MaxiMin is also used to determine the final best chromosome, used as the initial solution of a final full-length K-Means to deliver the final clustering solution: the solution with the minimum distance to utopia point (best values obtained for the objective functions during optimization process) is chosen is as the final best chromosome.

To enable the genetic operations, a fitness function is calculated using a weighted sum of normalized values of objectives functions. Values of objectives in MGenClust++ are normalized using the best and worst values of objectives found so far, since the value of cohesion is not constrained in the certain interval.

2. Datasets and Performance metrics

To test the algorithms, a set of ten diverse real-world and synthetic datasets from the UCI Machine Learning Repository (UCI) and from the Clustering Repository of the Speech and Image Processing Unit, at the University of Eastern Finland is collected. Glass Identification (Glass) and Breast Cancer Wisconsin (Wdbc) are standard real-world datasets wildly used in literature for clustering analysis. The latter one has a high percentage of outliers. Dim064 and Dim256 are high-dimensional synthetic datasets, having 64 and 256 attributes respectively. S1 and S3 are synthetic datasets used to test algorithms against the datasets with different degree of cluster overlap. Flame, Compound, Pathbased and Jain datasets are non-linearly separable synthetic datasets with different shapes and clusters. The results are compared based on three cluster validity indices, namely Adjusted Rand Index, Davies-Bouldin Index and Silhouette Coefficient, as well as, the number of clusters automatically determined by the algorithms. Thus, the results consist of both detecting the proper number of clusters and the quality of clustering solutions.

Table 1. A brief description of the datasets.

Name of dataset	Total No. of records	No. of attributes	No. of classes
Glass	214	10	6
Wdbc	569	30	2
Flame	240	2	2
Compound	399	2	6
Pathbased	300	2	3
Jain	373	2	2
S1	1000	2	15
S3	1000	2	15
Dim064	1024	64	16
Dim256	1024	256	16

3. Experimental setup

The values of parameters MGenClust++ share with GenClust++ are the same as recommended the original paper, except that a maximum number of generations is set to 20 to decrease the runtime of the algorithm, as the quality of clusters doesn't improve much after 20 generations. The maximum number of generations for GenClust++ is also set to 20. As suggested, the number of short K-Means iterations after cloning is 15, while the final K-Means on the best chromosome has its number of iterations bound to 50. The number of chromosomes in the population is set to 30. For all experiments and all algorithms that use some form of hill-climber of K-Means, the termination condition, that the difference of SSE of two consecutive iterations is less than 0.005, is imposed. The fitness function for GenClust ++ is DB Index, while MGenClust++ is using weighted sum of objectives. In addition, just as in the original paper, K-means with the Manhattan metric as the distance between among attributes and random seed initialization is used.

4. Experimental results

Each technique is run thirty times on each dataset recording their cluster quality for each run in terms of four evaluation metric/criteria: ARI, Silhouette Coefficient, DB Index and the difference between the real and the detected by the algorithms number of clusters. Tables 2-5 present the average results of the thirty clustering solutions on each dataset for each technique. Statistically significant values are highlighted in bold. The statistical two-tailed Wilcoxon signed rank test is carried out for the significance level α= 0.05 meaning 95% significance level. Tables 6-9 depict the Wilcoxon test results on 10 datasets. Since the number of observations is 30, the critical value is set to 137. For Tables 6 and 8 (higher ARI and Silhouette is better), R^+ score lower or equal to critical value = 137 indicates significantly better MGenClust++ performance and higher or equal to 328 – better GenClust++ performance, while the for Tables 7 and 8, R^+ score lower or equal to critical value = 137 indicates significantly better GenClust++ performance and higher or equal to 328 – better MGenClust++ performance.

In addition, Fig.1 presents the average results over all datasets for each algorithm in terms of three clustering evaluation techniques, as well as the differences between the real and the detected number of clusters.
Multi-objective GenClust++ performs significantly better in terms of ARI scores on four datasets than GenClust++ and shows similar results on three other datasets. On the other hand, GenClust++ shows a significantly better performance in terms of DB Index on six datasets including high dimensional datasets, and MGenClust++ have a better performance only on real-world Glass and Wdbc datasets. However, GenClust++ uses DB Index as a fitness function and therefore should result in better DB Index score than MGenClust++. For this reason, another robust internal cluster validity index, namely Silhouette Coefficient, is utilized. Table 4 shows that GenClust++ and MGenClust++ performs significantly better than the opponent on the equal number of datasets.
In terms of achieving optimal number of clusters, MGenClust++ produces better clustering solutions for six datasets, including all the datasets with different shapes and clusters (Flame, Compound, Pathbased and Jain), while GenClust++ - for datasets with higher number of clusters, which can be attributed to the fact that, that MGenClust++ optimizes two objectives at the same time, connectivity and cohesion, while GenClust optimizes only one, DB Index in this particular configuration.

When considering the average results over all datasets, MGenClust++ achieves better average results on two out three validity indices, namely ARI and Silhouette Coefficient, and nearly same average DB Index. Moreover, MGenClust++ finds closer to optimal number of clusters and is more robust and constistent than GenClust++.

Thus, MGenClust++ shows promising results, producing good quality clustering solutions and in many cases outperforming original GenClust++ algorithm on the selected benchmark datasets, as well as determining nearly optimal number of clusters.

Table 2. Mean and standard deviation of ARI (higher the better) measured on the outputs of GenClust++ and MGenClust++ (over 30 independent runs).

Dataset	GenClust++	MGenClust++
Glass	0.27 ± 0.0831	**0.5309** ± 0.1856
Wdbc	**0.7616** ± 0.0019	0.6984 ± 0.0392
Flame	0.3997 ± 0.0811	0.4316 ± 0.052
Compound	0.7164 ± 4.0E-4	0.7245 ± 0.0178
Pathbased	0.2972 ± 0.0661	**0.4486** ± 0.0169
Jain	0.1425 ± 0.0229	**0.5437** ± 0.0489
S1	0.8678 ± 0.1031	**0.9588** ± 0.0316

S3	0.6212 ± 0.0638	0.6472 ± 0.0315
DIM064	**0.9813** ± 0.0361	0.8827 ± 0.2126
DIM256	**0.9979** ± 0.0115	0.8273 ± 0.3035

Table 3. Mean and standard deviation of DB Index (lower the better) measured on the outputs of GenClust++ and MGenClust++ (over 30 independent runs).

Dataset	GenClust++	MGenClust++
Glass	1.2986 ± 1.0282	**0.5095** ± 0.0033
Wdbc	1.1263 ± 8.0E-4	**1.1477** ± 0.1063
Flame	**0.7202** ± 0.0264	0.8781 ± 0.1334
Compound	**0.5525** ± 0.0014	0.5893 ± 0.0637
Pathbased	0.7461 ± 0.0321	0.7474 ± 0.0537
Jain	**0.6971** ± 0.0149	0.7818 ± 0.0265
S1	0.4517 ± 0.065	0.4627 ± 0.0755
S3	**0.6884** ± 0.0221	0.7854 ± 0.0498
DIM064	**0.1675** ± 0.2123	0.3988 ± 0.3197
DIM256	**0.0427** ± 0.0942	0.5676 ± 0.6385

Table 4. Mean and standard deviation of Silhouette coefficient (higher the better) measured on the outputs of GenClust++ and MGenClust++ (over 30 independent runs).

Dataset	GenClust++	MGenClust++
Glass	0.2657 ± 0.1435	**0.5768** ± 0.0241
Wdbc	0.3812 ± 3.0E-4	**0.385** ± 0.0179
Flame	**0.4341** ± 0.027	0.3963 ± 0.0148
Compound	**0.6045** ± 2.0E-4	0.5912 ± 0.0249
Pathbased	0.3795 ± 0.0252	**0.4936** ± 0.045
Jain	0.4489 ± 0.0146	**0.5078** ± 8.0E-4
S1	0.653 ± 0.0484	**0.6804** ± 0.0232

S3	**0.4615** ± 0.0201	0.4461 ± 0.0173
DIM064	**0.9475** ± 0.035	0.8717 ± 0.1443
DIM256	**0.9806** ± 0.0111	0.8439 ± 0.2122

Table 5. Mean and standard deviation of the average number of clusters (over 30 independent runs) for GenClust++ and MGenClust++

Dataset	GenClust++	MGenClust++
Glass	3.1667 ± 2.4642	**3.3** ± 0.781
Wdbc	2.0 ± 0.0	2.1 ± 0.3
Flame	4.8 ± 2.3007	**2.7333** ± 0.4422
Compound	3.0 ± 0.0	**3.2667** ± 0.5121
Pathbased	13.3 ± 2.3402	**3.5333** ± 0.5617
Jain	10.4 ± 1.8903	**2.0333** ± 0.1795
S1	13.2333 ± 1.82	**15.7** ± 0.9713
S3	**12.7333** ± 2.0806	19.2667 ± 2.0645
DIM064	**15.7** ± 0.5859	15.1667 ± 2.5701
DIM256	**15.9667** ± 0.1795	15.1 ± 2.8792
Average difference between the real and the detected numbers of clusters	3.2767 ± 1.1837	1.5367 ± 1.0168

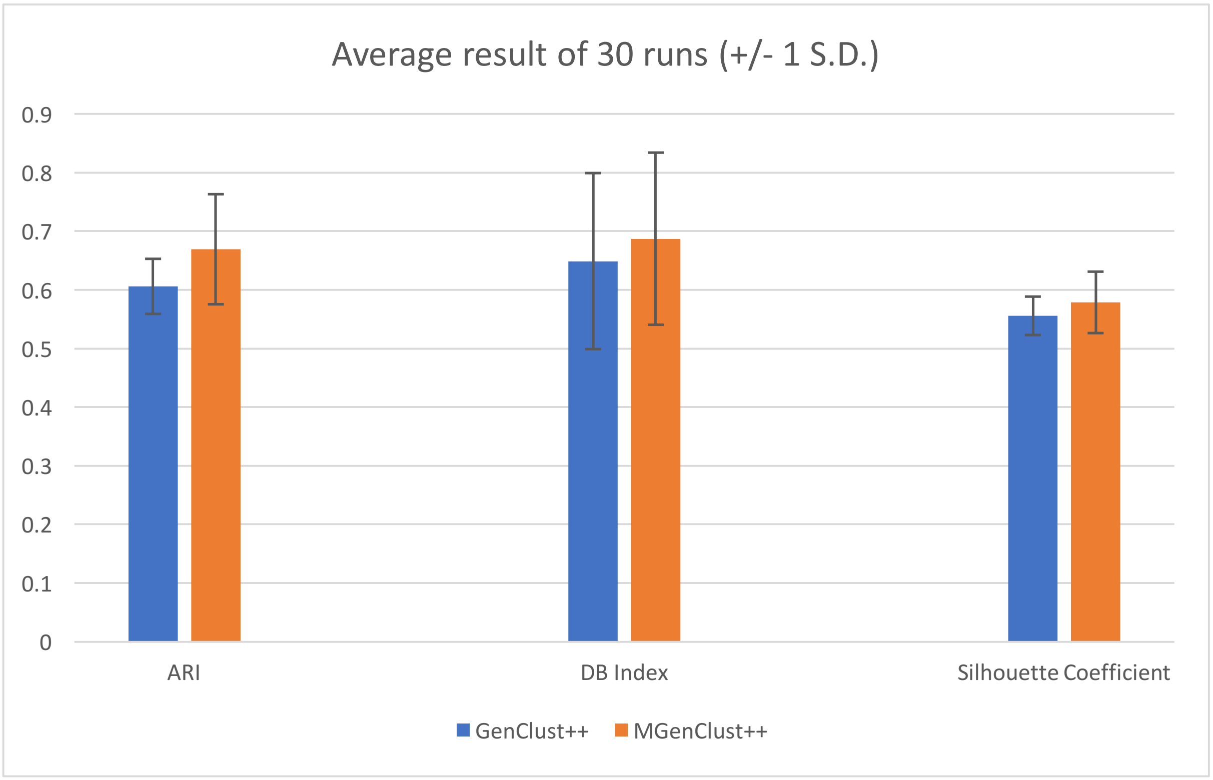

Fig. 1. Comparative average results between GenClust++ and MGenClust++ on 10 datasets based on ARI, DB Index and Silhouette Coefficient

Table 6. Wilcoxon signed ranks test results using ARI scores.

Dataset	R^+	R^-	*p*-value
Glass	7	458	3.1656e-06
Wdbc	465	0	1.6157e-06
Flame	186	279	0.3385
Compound	242.5	222.5	0.4872
Pathbased	7	458	3.5152e-06
Jain	0	465	1.7268e-06
S1	63.5	401.5	0.0005
S3	156	309	0.1156
DIM064	352	113	0.0138
DIM256	410	55	0.0002

Table 7. Wilcoxon signed ranks test results using DB Index scores.

Dataset	R^+	R^-	*p*-value
Glass	465.0	0	1.552e-06
Wdbc	378.0	87	1.6157e-06
Flame	1.0	464	1.8841e-06
Compound	10.5	454.5	2.9566e-06
Pathbased	284.0	181	0.2895
Jain	7.0	458	3.5007e-06
S1	217.5	247.5	0.7577
S3	5.0	460	2.8786e-06
DIM064	96.0	369	0.0049
DIM256	51.0	414	0.0002

Table 8. Wilcoxon signed ranks test results using Silhouette Coefficient scores.

Dataset	R^+	R^-	*p*-value
Glass	0.0	465.0	1.552e-06
Wdbc	87.0	378.0	0.0003
Flame	437.0	28.0	2.5574e-05
Compound	454.5	10.5	2.9566e-06
Pathbased	3.0	462.0	2.3534e-06
Jain	0.0	465.0	1.7268e-06
S1	125.5	339.5	0.0278
S3	347.0	118.0	0.0185
DIM064	369.0	96.0	0.0049
DIM256	411.0	54.0	0.0002

Table 9. Wilcoxon signed ranks test results using the difference between the real and the detected number of clusters scores.

Dataset	R^{+}	R^{-}	*p*-value
Glass	410.5	54.5	0.0002
Wdbc	189.0	276.0	0.3251
Flame	465.0	0.0	6.7718e-07
Compound	338.5	126.5	0.0395
Pathbased	464.5	0.5	1.511e-06
Jain	465.0	0.0	1.5253e-06
S1	345.0	120.0	0.0198
S3	76.0	388.0	0.0012
DIM064	82.0	383	0.002
DIM256	55.5	409.5	0.0002