Supplementary Materials for WC-KNNG-PC

In this material, information about the WC-KNNG-PC method and the comparison algorithms: K-means, DBSCAN, OPTICS, RNN-DBSCAN (RNN), CHKNN, ADBSCAN, cutESC, SNN-DPC, are presented, such as pseudo code of WC-KNNG-PC, the parameters used, and the noise ratio and their performance. In the following table, the horizontal line "−" indicates that the algorithm does not produce corresponding results or cannot run on the data set. The names of the datasets used here are denoted by their first 4 letters.

1. Description of data sets

Table I Description of data sets: Number of data points (N), Number of Dimensions (D), Number of Clusters (C)

	Data set	N	D	C		Dataset	N	D	C
Synthetic data	Aggregation	788	2	7	Real data	Spectrometer	531	100	48
	CMC	500	2	3		Ecoli	336	7	8
	Compound	399	2	6		Ionosphere	351	34	2
	D31	3100	2	31		Iris	150	4	3
	Flame	240	2	2		Libras movement	360	90	15
	Jain	373	2	2		Seeds	210	7	3
	Pathbased	300	2	3		Segmentation	2,310	19	7
	Spiral	312	2	3		Glass	214	9	7
	R15	600	2	15		Wdbc	569	30	2
	S2	5000	2	15		Wine	178	13	3
Image data	Mnist (test)	10000	55	10	Image data	Olivetti face	400	28	40
	Usps	9298	50	10					

2. Pseudo code of WC-KNNG-PC

Algorithm 1 WC-KNNG-PC

Input: dataset D, parameter *t* and k $(0 < t \leq k < n)$

Output: clustering results: catchment basins: $\Sigma = \{B_1, \cdots, B_b, \cdots, B_l\}$, L

1: Apply Algorithm 2 to construct KNNG for a given dataset

2: Apply Algorithm 3 to construct catchment basins

3: Apply Algorithm 4 to detect invalid basin immersions

4: Apply Algorithm 5 to merge catchment basins

Algorithm 2 Construct *k* nearest neighbor graph $G_{kNN}(V, E)$

Input: dataset *D*, nearest neighbor parameter *k* and *t*

Output: $G_{kNN}(V, E)$, *LA*

1: Calculate $N_t(x_i)$, $N_k(x_i)$, $pN_t(x_i)$, $pN_k(x_i)$ of every vertex $x_i \in D$,

2: Calculate the 1st, 2nd and 3rd weight of each edge in $G_{kNN}(V, E)$ from $SNN_t(x_i, x_j)$, $SNN_k(x_i, x_j)$, $RNN_t(x_i, x_j)$, $RNN_k(x_i, x_j)$, $RNNS_t(x_i)$, $RNNS_k(x_i)$

3: Calculate naïve altitude, refined altitude, and edge weight 4 of $G_{kNN}(V, E)$ according to Eqs.5-12 and 16

4: Calculate node attribute 2 and 3 of $G_{kNN}(V, E)$ according to Eqs. **错误!未找到引用源。-错误!未找到引用源。**

5: Calculate local anomalies according to Eq. **错误!未找到引用源。**

Algorithm 3 Detect catchment basins

Input: $G_{kNN}(V, E)$, *DL*, *t*, *k*, *LA*

Output: $\Sigma = \{B_1, \cdots, B_l\}$, $\Upsilon = \{BA_1, \cdots, BA_l\}$, $\Psi = \{Bcp_1, \ldots, Bcp_l\}$, Λ, L, O

1: Initialize a basin label list of all data points $L = [-1, -1, \cdots]$, outlies $O = \emptyset$

2: Sort all nodes in ascending order by their altitudes into the queue $DL = (x_1', x_2' \cdots, x_n')$

3: **while** *DL* is not empty **do**

4: $q \leftarrow$ Pop a point from the *T*'s head

5: **if** $\tau(q)$'s level is 1, **then** go to step 4

6: **if** $L[q] == -1$ **then** New a Catchment basin: B_q, $BA_q = \emptyset$, $Bcp_q = \emptyset$, and set $B_q = \{q\}$, $L[q] = q$

7: **for** each $z \in pN_k(q)$ **do**

8: **if** $\tau(z) = 0$, **then** skip to next *z* and go to step 7

9: **if** $IPS(\alpha(q), \alpha(z), k) == 1$ **then**

10: **if** *z* doesn't belong any catchment basin, **then**

11: **if** $IIM(q, z)$ **then**

12: Add *z* to $B_{L[q]}$, set $L[z] = L[q]$, and **if** $z \in LA$, **then** add *z* to $BA_{L[q]}$

13: **else then**

14: **if** $L[z] \neq L[q]$ **then** Add (*q*, *z*) to Λ, z to $Bcp_{L[q]}$

15: **end if**

16: **end if**

17: **end for**

18: **end while**

19: **for each** $z \in DL$ do

20: **if** $L[z] < 0$, **then** add *z* to *O*

21: **end for**

In Algorithm 3, if the immersion stability of x_i belongs to level 1 (see **错误!未找到引用源。**), x_i is processed in two ways: (1) if $0 < \tau(x_i) < 0.2$, x_i can only be used as the immersion point; (2) x_i must be an outlier when $\tau(x_i) = 0$. Therefore, if $\tau(x_i)$ belongs to level 1, point x_i may be an outlier that cannot be clustered by other basins.

Algorithm 4 Detection of invalid catchment basin immersions

Input: $G_{kNN}(V, E)$, Σ, Υ, Λ, Υ, Ψ

Output: invalid catchment basins immersions Γ

1: Initialize the invalid immersions in all catchment basins $\Gamma = \emptyset$

2: Calculate $BpN_t(x_i)$ of point $x_i \in D$ according to **错误!未找到引用源。**

3: **for** each $BA \in \Upsilon$ **do**

4: **if** $|BA| > 0$ **then**

5: **for each** $p \in B_b$ **do**

6: **if** $\vartheta(p) \in level1$，**then** add $(p, x_i) \in \Lambda$ to Γ

for each z∈ $pN_t(p)$ **do**

if $\vartheta(z) \in level1$，**then** add $(x_i, x_j) \in \Lambda$ to Γ

end for

end for

end if

end for

Algorithm 5 Merging catchment basins

Input: $G_{kNN}(V,E)$, Σ, Π, Λ, Γ

Output: L, Σ

Sort Λ in ascending order of $min\left(\alpha(x_i), \alpha(x_j)\right)$ into the queue Λ'

for each $r \in \Lambda'$ **do**

if $r \notin \Gamma$ **then**

if $BM(x_i, x_j)$ **then**

if $\alpha(L[r[0]]) \leq \alpha(L[r[1]])$ **then**

Set the basin label of all points in $B_{L[r[1]]}$ to $L[r[0]]$ **and** merge $B_{L[r[1]]}$ into $B_{L[r[0]]}$

else then

Set the basin label of all points in $B_{L[r[0]]}$ to $L[r[1]]$ **and** merge $B_{L[r[0]]}$ into $B_{L[r[1]]}$

end if

end if

end if

end for

Algorithm 6 Allocate outlies O

Input: $G_{kNN}(V,E)$, Σ, L, O, k

Output: Σ, L

Sort outlies O in increasing order of the refine altitude to O'

for all $q \in O'$ **do**

for all $z \in pN_k(q)$ **do**

if $SNN_k(q,z) \geq \lfloor k/2 \rfloor$ and $L[z] \geq 0$ **then**

Set $L[q] = L[z]$ and add q to $B_{L[q]}$

Jump out of the loop and skip to the next outlier: q

end if

end for

end for

3. AMI performance on Artificial Datasets

Table III AMI performance on Artificial Datasets.

Data set	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-DPC	WC
Aggr	0.849(0.015)	0.983(0.000)	0.953(0.000)	0.993(0.001)	0.996(0.000)	0.986(0.000)	0.937(0.000)	0.950(0.000)	**0.996(0.000)**
CMC	0.194(0.065)	0.991(0.000)	0.991(0.000)	0.862(0.000)	**1.000(0.000)**	0.782(0.000)	0.758(0.000)	**1.000(0.000)**	0.920(0.000)
Comp	0.729(0.046)	0.945(0.000)	0.946(0.000)	0.869(0.003)	0.881(0.000)	0.867(0.000)	0.940(0.000)	0.828(0.000)	**0.984(0.000)**
D31	0.943(0.011)	0.906(0.000)	0.905(0.000)	0.909(0.001)	0.963(0.000)	0.878(0.000)	0.815(0.000)	**0.964(0.000)**	0.949(0.000)
Flam	0.409(0.25)	0.899(0.000)	0.837(0.000)	**0.954(0.018)**	0.935(0.000)	0.682(0.000)	0.834(0.000)	0.900(0.000)	0.927(0.000)
Jain	0.365(0.005)	0.856(0.000)	0.852(0.000)	0.939(0.029)	0.883(0.000)	**1.000(0.000)**	0.896(0.000)	0.379(0.000)	**1.000(0.000)**
Path	0.544(0.001)	0.862(0.000)	0.898(0.000)	0.870(0.008)	0.860(0.000)	0.762(0.000)	0.795(0.000)	0.901(0.000)	**0.907(0.000)**
R15	0.971(0.023)	0.979(0.000)	0.980(0.000)	0.989(0.004)	**0.994(0.000)**	0.940(0.000)	0.809(0.000)	**0.994(0.000)**	0.976(0.000)
Spir	0.000(0.000)	**1.000(0.000)**	**1.000(0.000)**	**1.000(0.000)**	0.963(0.000)	0.886(0.000)	0.794(0.000)	**1.000(0.000)**	**1.000(0.000)**
S2	0.927(0.017)	0.811(0.000)	0.811(0.000)	0.896(0.001)	**0.949(0.000)**	0.723(0.000)	0.784(0.000)	0.937(0.000)	0.945(0.000)
Average	0.5931	0.9232	0.9173	0.9281	0.9424	0.8506	0.8362	0.8853	**0.9604**

4. AMI performance on Real-world Datasets

Table III AMI performance on real-world Datasets

Data set	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-DPC	WC
Spec	**0.514(0.009)**	0.269(0.000)	0.220(0.000)	0.433(0.000)	0.469(0.000)	0.430(0.000)	—	0.004(0.000)	0.277(0.000)

Ecol	0.605(0.034)	0.493(0.000)	0.562(0.000)	0.517(0.004)	0.665(0.000)	0.521(0.000)	0.445(0.000)	**0.671(0.000)**	0.656(0.000)
Libr	0.529(0.018)	0.454(0.000)	0.465(0.000)	0.553(0.002)	0.468(0.000)	0.522(0.000)	—	0.583(0.000)	**0.600(0.000)**
Iono	0.128(0.024)	**0.601(0.000)**	0.581(0.000)	0.518(0.000)	0.396(0.000)	0.381(0.000)	—	0.001(0.000)	0.429(0.000)
Iris	0.734(0.046)	0.619(0.000)	0.732(0.000)	0.659(0.005)	0.869(0.000)	0.667(0.000)	0.714(0.000)	**0.912(0.000)**	0.770(0.000)
Seed	0.700(0.008)	0.586(0.000)	0.558(0.000)	0.608(0.004)	0.736(0.000)	0.540(0.000)	0.493(0.000)	**0.738(0.000)**	0.736(0.000)
Segm	0.428(0.072)	0.610(0.000)	0.610(0.000)	0.639(0.001)	**0.732(0.000)**	0.509(0.000)	—	0.000(0.000)	0.650(0.000)
Glas	0.392(0.025)	0.378(0.000)	0.378(0.000)	0.376(0.000)	0.364(0.000)	**0.418(0.000)**	0.380(0.000)	0.275(0.000)	0.341(0.000)
Wdbc	0.464(0.000)	0.367(0.000)	0.389(0.000)	0.395(0.007)	0.635(0.000)	0.349(0.000)	—	**0.752(0.000)**	0.423(0.000)
Wine	0.418(0.006)	0.586(0.000)	0.469(0.000)	0.381(0.017)	0.468(0.000)	0.383(0.000)	—	**0.874(0.000)**	0.349(0.000)
Oliv	0.743(0.017)	0.794(0.000)	0.794(0.000)	0.712(0.001)	0.358(0.000)	0.748(0.000)	—	0.837(0.000)	**0.838(0.000)**
Mnis	0.510(0.014)	0.266(0.000)	0.266(0.000)	0.225(0.000)	0.579(0.000)	0.467(0.000)	—	0.662(0.000)	**0.686(0.000)**
Usps	0.625(0.014)	0.248(0.000)	0.424(0.000)	0.503(0.000)	0.690(0.000)	0.534(0.000)	—	0.684(0.000)	**0.720(0.000)**
Average	0.5223	0.4824	0.4960	0.5015	0.5715	0.4976	0.5080	0.5379	**0.5750**

5. Noise ratio detected by different methods on synthetic datasets

Table IV Noise ratio detected by different methods on synthetic datasets

Data set	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-DPC	WC
Aggr	—	0.001(0.000)	0.020(0.000)	0.001(0.000)	—	0.006(0.000)	0.052(0.000)	—	0.000(0.000)
CMC	—	0.001(0.000)	0.001(0.000)	0.013(0.000)	—	0.002(0.000)	0.060(0.000)	—	0.016(0.000)
Comp	—	0.128(0.000)	0.140(0.000)	0.035(0.000)	—	0.000(0.000)	0.000(0.000)	—	0.003(0.000)
D31	—	0.064(0.000)	0.074(0.000)	0.054(0.000)	—	0.089(0.000)	0.122(0.000)	—	0.008(0.000)
Flam	—	0.008(0.000)	0.025(0.000)	0.008(0.000)	—	0.008(0.000)	0.067(0.000)	—	0.000(0.000)
Jain	—	0.013(0.000)	0.016(0.000)	0.008(0.000)	—	0.000(0.000)	0.273(0.000)	—	0.000(0.000)
Path	—	0.357(0.000)	0.370(0.000)	0.037(0.000)	—	0.360(0.000)	0.433(0.000)	—	0.020(0.000)
R15	—	0.012(0.000)	0.010(0.000)	0.000(0.000)	—	0.047(0.000)	0.162(0.000)	—	0.010(0.000)
Spir	—	0.000(0.000)	0.000(0.000)	0.000(0.000)	—	0.006(0.000)	0.026(0.000)	—	0.000(0.000)
S2	—	0.183(0.000)	0.185(0.000)	0.056(0.000)	—	0.002(0.000)	0.166(0.000)	—	0.001(0.000)

6. Noise ratio detected by different methods on real-world datasets

Table V Noise ratio detected by different methods on real-world datasets

Dataset	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-	WC
Spec	—	0.407(0.000)	0.488(0.000)	0.143(0.000)	—	0.047(0.000)	—	—	0.017(0.000)
Ecol	—	0.301(0.000)	0.241(0.000)	0.030(0.000)	—	0.057(0.000)	0.220(0.000)	—	0.048(0.000)
Libr	—	0.183(0.000)	0.196(0.000)	0.097(0.000)	—	0.044(0.000)	—	—	0.042(0.000)
Iono	—	0.336(0.000)	0.330(0.000)	0.256(0.000)	—	0.251(0.000)	—	—	0.336(0.000)
Iris	—	0.220(0.000)	0.000(0.000)	0.107(0.000)	—	0.493(0.000)	0.160(0.000)	—	0.013(0.000)
Seed	—	0.305(0.000)	0.348(0.000)	0.019(0.000)	—	0.181(0.000)	0.210(0.000)	—	0.000(0.000)
Segm	—	0.100(0.000)	0.100(0.000)	0.041(0.000)	—	0.007(0.000)	—	—	0.027(0.000)
Glas	—	0.182(0.000)	0.182(0.000)	0.107(0.000)	—	0.206(0.000)	0.313(0.000)	—	0.056(0.000)
Wdbc	—	0.313(0.000)	0.322(0.000)	0.014(0.000)	—	0.025(0.000)	—	—	0.019(0.000)
Wine	—	0.573(0.000)	0.640(0.000)	0.039(0.000)	—	0.000(0.000)	—	—	0.006(0.000)
Oliv	—	0.078(0.000)	0.078(0.000)	0.133(0.000)	—	0.018(0.000)	—	—	0.030(0.000)
Mnis	—	0.402(0.000)	0.402(0.000)	0.021(0.000)	—	0.005(0.000)	—	—	0.181(0.000)
Usps	—	0.698(0.000)	0.539(0.000)	0.044(0.000)	—	0.005(0.000)	—	—	0.196(0.000)

7. Arguments used by different methods on Synthetic datasets

Table VI Arguments used by different methods on Synthetic datasets

Data set	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-DPC	WC
	k	*eps, mps*	*eps, mps*	*k*	*p1, p2, p3, m*	*k, np*	*α, β*	*nc, k*	*t, k*
Aggr	7	1.7, 10	1.8, 10	12	4, 17, 1, 50	39, 0.32	0.66, 0.38	7, 15	9, 19
CMC	3	0.011, 1	0.011, 1	13	8, 30, 3, 50	27, 0	1.00, 1.00	22, 3	8, 35
Comp	6	1.5, 2	1.5, 2	8	5, 5, 2, 50	28, 0	1.00, 1.00	18, 6	16, 19
D31	31	0.95, 34	0.95, 34	35	32, 32, 3, 50	33, 0.01	0.60, 0.95	41, 31	27, 30
Flam	2	1.3, 7	1.3, 7	8	9, 9, 1, 50	22, 0.16	0.42, 0.56	5, 2	23, 23
Jain	2	2.3, 2	2.3, 2	15	5, 5, 4, 100	29, 0	1.00, 1.00	18, 2	15, 30
Path	3	2.4, 10	2.4, 10	6	8, 10, 1, 50	9, 0.38	0.52, 1.00	9, 3	9, 19

R15	15	0.53, 12	0.53, 11	30	7, 8, 1, 100	40, 0.01	1.00, 0.36	15, 10	13, 23
Spir	3	1.11, 1	1.11, 1	2	2, 2, 1, 50	14, 0.08	0.70, 1.00	3, 9	10, 15
S2	15	0.03, 32	0.03, 32	202	40,45,3,400	19, 0.52	1.00, 1.00	35, 15	31, 33

8. Arguments used by different methods on real-world datasets

Table VII Arguments used by different methods on real-world datasets

Data set	Method								
	K-means	DBSCAN	OPTICS	RNN	CHKNN	ADBSCAN	cutESC	SNN-DPC	WC
	k	*eps, mps*	*eps, mps*	*k*	*p1, p2, p3, m*	*k, np*	*α, β*	*nc, k*	*t, k*
Spec	48	0.6, 25	0.6, 25	2	5, 5, 1, 200	14, 0.1	—	48, 12	36, 36
Ecol	8	0.2, 21	0.23, 29	3	9, 10, 1, 200	20, 0	0.11, 0.27	8, 6	9, 20
Libr	15	0.9, 1	0.89, 1	4	18,18,2,100	13, 0	—	15, 11	7, 11
Iono	2	0.78, 9	0.8, 9	15	13,14,2,200	31, 0.25	—	2, 5	19, 56
Iris	3	0.13, 5	0.4, 6	5	5, 15, 1,100	25, 0.36	0.90, 0.16	3, 15	16, 27
Seed	3	0.24, 15	0.24,15	5	13,13,1,100	14, 0.33	0.74, 0.23	3, 6	17, 24
Segm	7	0.15, 1	0.15, 1	10	11,60,1,300	21, 0	—	7, 7	27, 41
Glass	7	0.27, 8	0.27, 8	5	6, 6, 1, 100	4, 0.05	0.95, 1	6, 20	15, 45
Wdbc	2	0.51, 65	0.51, 65	19	62,76,11,300	19, 0.12	—	2, 12	14, 34
Wine	3	0.50, 20	0.50, 20	42	18,18,3,100	22, 0.06	—	3, 18	16, 28
Oliv	40	0.73,1	0.73, 1	3	13,14,1,100	9, 0.06	—	40, 6	5, 7
Mnis	10	0.62,1	0.62, 1	3	40,45,3,500	17, 0	—	10, 14	100, 127
Usps	10	0.71, 61	0.71, 61	2	45,50,4,450	20, 0.01	—	10, 13	99, 99

9. The results generated by 9 clustering algorithms on different datasets

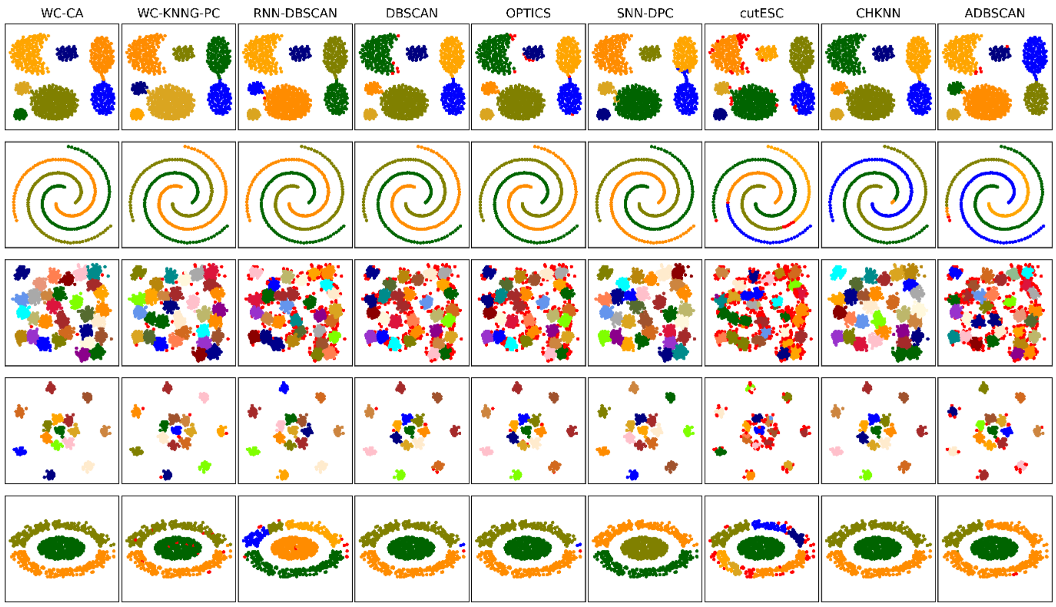

Fig. I Clustering results of different methods on Aggregation, CMC, D31, R15 and Spiral datasets. Different colors indicate different classes, but the red points are outliers.

10. WC-KNNG-PC on the dataset Pathbsed with different arguments

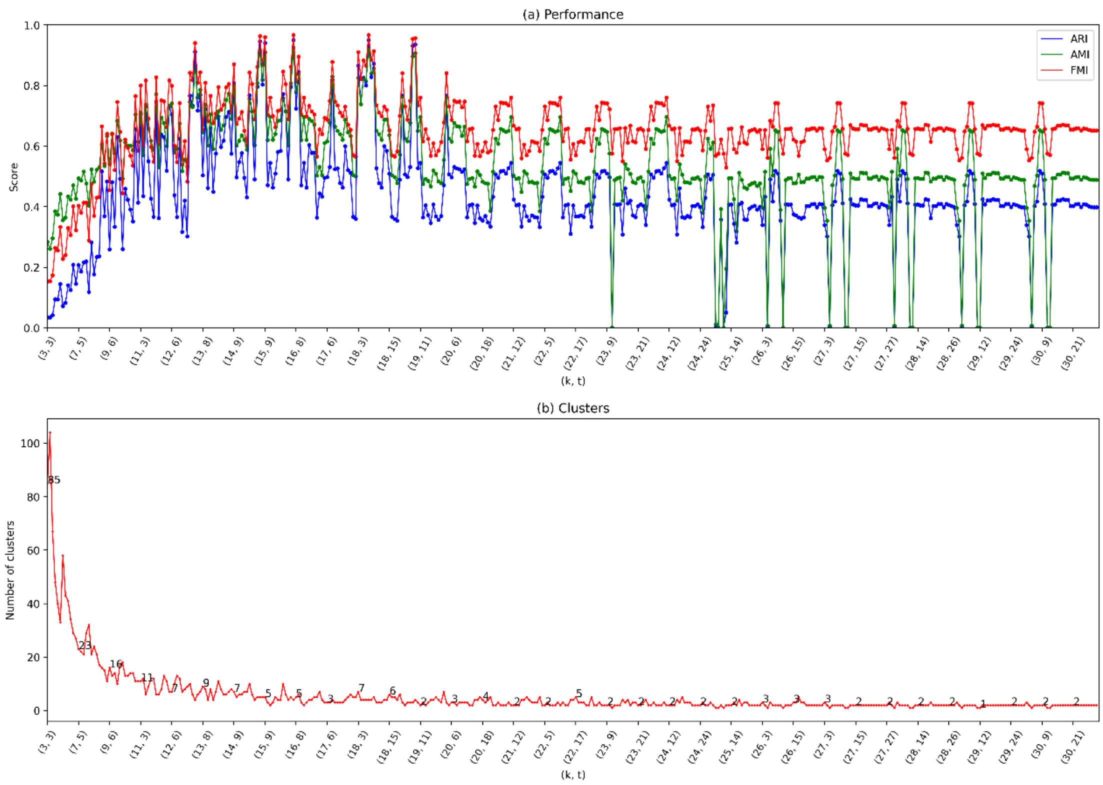

Fig. II WC-KNNG-PC on the Pathbsed dataset with different *t* and *k* which range from (3, 3) to (30, 30).

(a) Performances. (b) Clusters.

11. Run-time (seconds) analysis for the methodology

Table VIII Run-time (seconds) analysis for the methodology

Data set	P1	P2	P3	P4
Flam	0.744	0.005	2.430	0.001
Jain	0.528	0.013	1.578	0.003
CMC	0.584	0.056	1.152	0.015
D31	13.819	0.169	43.657	0.050
S2	1.610	0.010	5.583	0.000
Iris	0.248	0.003	0.714	0.000
Seed	0.375	0.005	1.156	0.001
Spec	8.955	0.019	13.369	0.002
Segm	11.566	0.192	31.239	0.008
Usps	585.798	4.858	1451.913	0.241