Les Cahiers du GERAD ISSN: 0711–2440

On strategies to fix degenerate k-means solutions

D. Aloise, N. Castelo Damasceno,
N. Mladenović, D. Nobre Pinheiro

G–2016–81

October 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

On strategies to fix degenerate k-means solutions

Daniel Aloise [a,c]
Nielson Castelo Damasceno [a]
Nenad Mladenović [b,c]
Daniel Nobre Pinheiro [a]

[a] *Department of Computer Engineering and Automation, Federal University of Rio Grande do Norte, Brazil*

[b] *LAMIH, Universit de Valenciennes et du Hainaut Cambrésis, Valenciennes, France*

[c] *GERAD, Montréal, Canada*

aloise@dca.ufrn.br
nielsen@dca.ufrn.br
nenad.mladenovic@univ-valenciennes.fr
daniel.npinheiro@bct.ect.ufrn.br

October 2016

Les Cahiers du GERAD
G–2016–81

Abstract: The k-means is a benchmark algorithm used in cluster analysis. It belongs to the large category of heuristics based on location-allocation steps that alternately locate cluster centers and allocate data points to them until no further improvement is possible. Such heuristics are known to suffer from a phenomenon called degeneracy in which some of the clusters are empty. In this paper, we compare and propose a series of strategies to circumvent degenerate solutions during a k-means execution. Our computational experiments show that these strategies are effective leading to better clustering solutions in the vast majority of the cases in which degeneracy appears in k-means. Moreover, we compare the use of our fixing strategies within k-means against the use of two initialization methods found in the literature. These results demonstrate how useful the proposed strategies can be, specially inside memory-based clustering algorithms.

Keywords: k-means, minimum sum-of-squares, degeneracy, clustering, heuristics

Acknowledgments: The authors wish to thank two anonymous referees for their constructive comments. Daniel Aloise and Nenad Mladenović were partially supported by CNPq-Brazil grants 308887/2014-0 and 400350/2014-9. Daniel Nobre Pinheiro is grateful to CAPES-Brazil.

1 Introduction

Generally speaking, the clustering problem consists of finding subsets, denoted *clusters*, from a given set of entities such that entities in the same cluster are similar while entities in different clusters should differ one from another. Among the many criteria used for clustering, one of the most used is the minimum sum of squared Euclidean distances from each entity to the centroid of the cluster to which it belongs. Partitioning n entities into k clusters with this criterion is known as minimum sum-of-squares clustering (MSSC).

A mathematical formulation of MSSC can be given by:

$$\begin{array}{l} \min\limits_{x,y} \quad \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k} x_{ij}\|p_i - y_j\|^2 \\ \text{subject to} \\ \sum\limits_{j=1}^{k} x_{ij} = 1, \quad \forall i = 1,\ldots,n \\ x_{ij} \in \{0,1\}, \quad \forall i = 1,\ldots,n; \forall j = 1,\ldots,k. \end{array} \tag{1}$$

The n entities $O = \{o_1, o_2, \ldots, o_n\}$ to be clustered are at given points $p_i = (p_i^r, r = 1,\ldots,s)$ of $\mathbb{R}^s$ for $i = 1,\ldots,n$; k cluster centers must be located at unknown points $y_j \in \mathbb{R}^s$ for $j = 1,\ldots,k$; the norm $\|\cdot\|$ denotes the Euclidean distance between the two points in its argument in the s-dimensional space under consideration. The decision variables x_{ij} express the assignment of the entity o_i to the cluster j. The number of entities n is greater than k, otherwise the problem is trivially solved by locating one cluster center at the position of each entity.

The problem is NP-hard in the plane for general values of k (Mahajan et al., 2009). In general dimension, MSSC is NP-hard even for $k = 2$ (Aloise et al., 2009). If both k and dimension s are fixed, the problem can be solved in $O(n^{sk+1})$ time (Inaba et al., 1994), which may be very time-consuming even for instances in the plane. Optimal solutions might still be provided by worst-case exponential algorithms based on advanced mathematical programming techniques. Aloise et al. (2012) provide a branch-and-price algorithm for MSSC that solves to optimality benchmark clustering instances with up to 2000 entities.

Regarding approximate methods, several hundred papers have been written on heuristics for MSSC and its variants (examples are Choromanska and Monteleoni (2012); Ding et al. (2015); Ordin and Bagirov (2015); Tao et al. (2014); Teboulle (2007)). Among them, the most popular one is indeed k-means (Forgy, 1965) with more than 2 million results in Google (see Steinley (2006) for an extensive survey). The k-means heuristic was voted by the IEEE Computer Society as the 2nd most influential algorithm in the data mining community (Wu and Kumar, 2009). It takes advantage of the following two properties of the MSSC:

(i) If y is fixed, the condition $x_{ij} \in \{0,1\}$ can be replaced by $x_{ij} \in [0,1]$, since in an optimal solution for the resulting problem each entity belongs to the cluster with the nearest center.

(ii) For a fixed x, first order conditions on the gradient of the objective function require that at an optimal solution the optimal cluster centers are always at the centroids of the clusters given by

$$\sum_{i=1}^{n} x_{ij}(y_j^r - p_i^r) = 0, \ \forall j, r, \ \text{i.e., } y_j^r = \frac{\sum\limits_{i=1}^{n} x_{ij} p_i^r}{\sum\limits_{i=1}^{n} x_{ij}}, \ \forall j, r. \tag{2}$$

From an initial partition, k-means proceeds by reassigning the objects to their closest centroid and updating their positions until stability is reached. In other words, k-means alternately applies properties (i) and (ii) above until a local minimum is attained. The name of the algorithm is usually imputed to MacQueen (1967), though his method concerns assigning each entity at a time to its closest centroid, then updating the location of the involved centroids right after an entity is reassigned from a cluster to another. The algorithm terminates when all entities are eventually assigned to their closest centroids. Nowadays, most of the literature calls k-means the algorithmic steps performed by the method of Forgy (1965). That is also the terminology convention adopted in this paper.

Depending on its initialization, k-means may lead to empty clusters. In other words, a better solution in the next iteration of the algorithm may be found but with a lower number of clusters. Clearly, such solutions may be easily improved by adding a new centroid at the position of any existing entity (as long as that entity is not the centroid of another cluster).

In this paper, we propose and compare six strategies to fix degenerate k-means solutions. They are based on simple and efficient ideas which can be straightforwardly embedded in any k-means implementation. The motivation for them is twofold. First, alternate clustering heuristics such as k-means arise in a wide number of fields such as logistics (Cooper, 1964), oil industry (Haverly, 1978), neurosciences (Mairal et al., 2012; Mak and Wolpaw, 2009), marketing (Blanchard et al., 2012), etc. They are known to share the shortcoming of suffering from degeneracy (Brimberg and Mladenović, 1999; Hofmans et al., 2015; Nugent et al., 2010). Second, one could argue that the degeneracy issue could be considered insignificant by using good initialization methods (Steinley and Brusco, 2007) or by repeating k-means for a number of times. However, many of the best methods found in the literature for MSSC uses k-means as a subroutine, as part of a higher optimization framework (Brusco and Steinley, 2007; Hansen and Mladenović, 2001; Pacheco and Valencia, 2003), so that the k-means can be started with all kinds of solutions, including degenerate ones. In these settings, degenerate solutions may lead to poor local minima or even anomalous behavior.

The paper is organized as follows. In the next section, we present a numerical example of degeneration in k-means. Section 3 presents a series of algorithmic strategies to remove k-means degeneracy, including four new ones proposed in this paper. Computational experiments that compare the strategies regarding their efficiency and effectiveness for MSSC are reported in Section 4. Finally, the conclusions are presented in Section 5.

2 Degeneracy of k-means

There are basically two types of possible degenerate solutions in MSSC optimization. These solutions arise when during a k-means execution:

1. one or more clusters are empty, i.e., they do not have any point assigned to it;
2. at least two centroids have the same coordinates, being in fact a unique cluster.

Figure 1 illustrates the occurrence of a degeneration of type 1 after k-means is applied for the following 20 two-dimensional points:

	x	y
p_1	24.64	22.00
p_2	20.03	32.38
p_3	21.26	20.93
p_4	24.85	28.22
p_5	29.94	18.45
p_6	20.66	-27.06
p_7	22.63	-22.44
p_8	26.11	-30.99
p_9	34.36	-25.48
p_{10}	25.55	-22.77
p_{11}	-26.48	34.98
p_{12}	-25.84	28.48
p_{13}	-24.10	18.17
p_{14}	-22.89	26.81
p_{15}	-16.61	22.16
p_{16}	-30.22	-29.43
p_{17}	-21.51	-34.20
p_{18}	-22.58	-33.14
p_{19}	-25.97	-30.87
p_{20}	-26.89	-27.08

with initial cluster centers located at p_3, p_9, p_{18} and p_{20} for $k = 4$ clusters. The red 'x' symbols correspond to the cluster centers. Figure 1(a) presents the initial allocation step of each point to its closest cluster center.

In Figure 1(b), the centers are updated to correspond to centroids (due to equations (2)). Figure 1(c) shows the second allocation step where a centroid is found to have no point assigned to it. The final degenerate solution is shown in Figure 1(d).

Although Figure 1 presents an example where a degenerate solution of type 1 is found in the first iteration, our k-means implementation keeps the empty centroids for subsequent iterations of the algorithm. This allow k-means to ocasionally reduce degeneration of type-1 automatically, i.e., by assigning points to that empty centroids in the next iterations. The empty centroids are eventually removed at the end of the k-means algorithm.

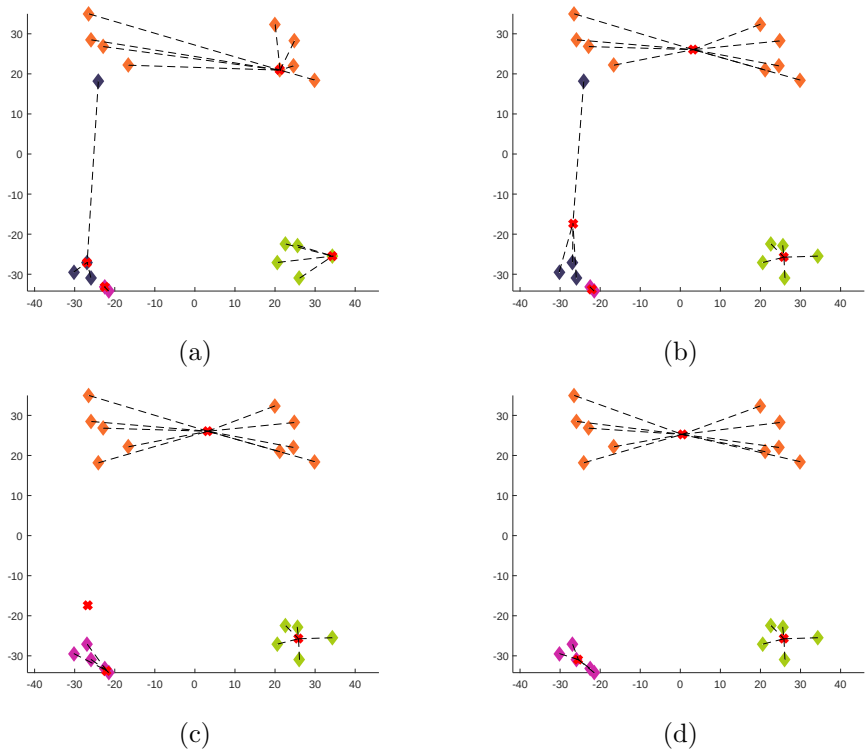

Figure 1: k-means example leading to a degenerate solution

The number of different partitions of n entities into k non-empty clusters is given by the Stirling number of second kind

$$S(n,k) = \frac{1}{k!}\sum_{t=0}^{k-1}(-1)^t\binom{k}{t}(k-t)^n. \tag{3}$$

It is well-known that $S(n,k)$ is first increasing, then decreasing with respect to k. The number of partitions of n entities into k or fewer clusters is given by

$$\beta = \sum_{j=1}^{k} S(n,j). \tag{4}$$

Thus, the proportion of non-degenerate partitions of n entities into k clusters is equal to

$$P(n,k) = \frac{S(n,k)}{\beta}. \tag{5}$$

The proportion of non-degenerate partitions $P(n,k)$ is relatively high for small k, but decreases rapidly with larger values of k towards approximately 0%.

Table 1 shows for different instances the number of times that a degenerate solution was found in 100 distinct k-means executions. The table separates in its last column the cases for which k-means was able to fix degeneracy automatically. The initial solutions were obtained from random partitions of the n data points among the k clusters.

Table 1: Number of degenerate solutions found in 100 k-means executions

Instance	n	k	# of degenerate sols.	# solved by k-means
Eilon (Eilon et al., 1971)	50	5	17	17
		10	90	83
		15	100	2
Ruspini (Ruspini, 1970)	75	5	67	66
		10	100	3
		20	100	0
Iris(Lichman, 2013)	150	5	60	60
		10	99	34
		20	100	0
Wine(Lichman, 2013)	178	5	11	11
		10	46	46
		20	96	16
B-Cancer (Lichman, 2013)	699	5	0	0
		10	73	73
		20	98	98
image segmentation (Lichman, 2013)	2312	5	2	2
		10	48	48
		20	99	99

We see in the table that random initial solutions lead very often to degenerate solution, specially when the relation n/k is small. Yet, the last column shows that k-means struggles to fix degeneracy automatically as the relation n/k decreases.

Let us denote d as the *degree of degeneracy*, i.e., the number of empty clusters in a degenerate solution. As supposed by equations (3–5), we observe in Table 2 that for the Eilon 50-point dataset from (Eilon et al., 1971) the distribution of d for varying values of k tends to be bell-shaped, resembling a normal distribution. Each row presents the number of times that degenerate solutions with different degrees of degeneracy were obtained out of 100 k-means executions for a given number of clusters k.

Table 2: Distribution of the degree of degeneracy obtained by 100 distinct k-means runs applied to the 50-point problem from (Eilon et al., 1971)

	degree of degeneracy (d)									
k	0	1	2	3	4	5	6	7	8	9
4	100									
8	100									
12	69	25	6							
16	12	34	34	18		2				
20	2	7	22	35	26	7	1			
24		5	23	24	22	14	6	5	1	
28			4	17	20	24	20	12	3	
32			3	14	16	26	22	12	6	1
36			1	3	12	22	34	18	9	1
40				6	13	16	36	20	8	1
44		1	1	12	32	47	7			
48		18	82							

3 Strategies for fixing degenerate solutions

In the sequel, we present six strategies to fix k-means degenerate solutions with d empty clusters. The first two were previously used in the literature whereas the last four are proposed here.

3.1 Random

This strategy consists of randomly selecting d new centroids from the given data points. It was first proposed by Brimberg and Mladenović (1999) for the multi-source Weber problem whose only difference to MSSC consists in using Euclidean distances instead of squared ones.

Figure 2 shows the next k-means iteration after the Random strategy is applied to the degenerate solution of Section 2. The light-blue 'x' symbols in the Figure 2 (as well as in the following ones) represent new cluster centers added to the partition.

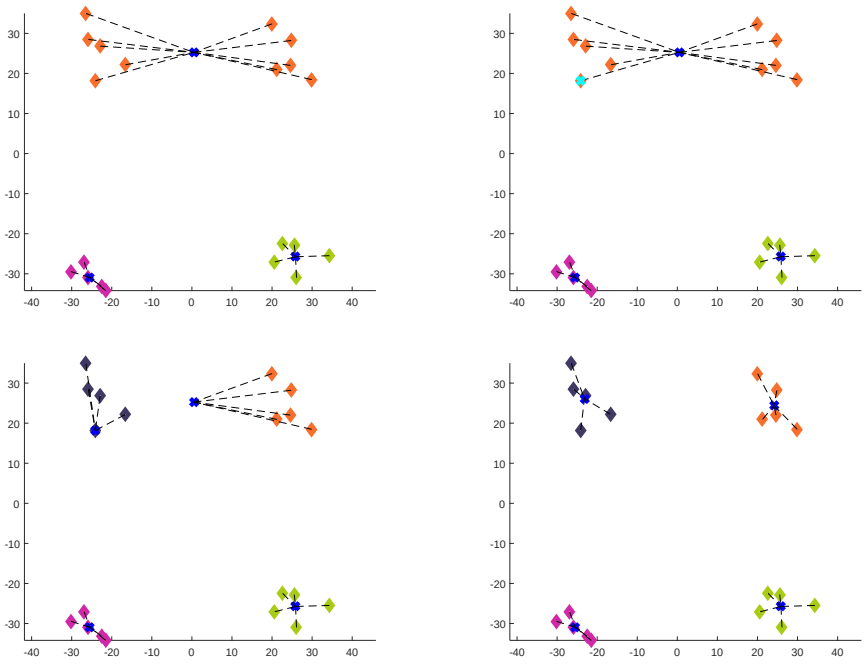

Figure 2: Application of strategy Random to the degenerate solution of Figure 1

3.2 Greedy

Among the possible assignments for the Random strategy, the best ones are those in which the new centroids are the d points with largest squared distances to their cluster's centroid (i.e., those with the d largest contributions to the objective function). This strategy was first applied by Bradley and Fayyad (1998) to restart k-means after its convergence to a solution containing empty clusters.

Since this is a greedy (myopic) decision, it is not necessarily true that k-means with the Greedy strategy will always outperform the method that uses the Random strategy to remove degeneracy. Figure 3 illustrates the following k-means iteration when the Greedy strategy is applied to the example of Section 2.

3.3 ε-Random

In this strategy, d new centroids are derived from the centroids of a degenerate solution. The procedure works in two steps. First, a centroid y is randomly selected from the solution. Then, each coordinate of the new centroid y' is obtained from the associated coordinate in y by perturbing it by $+\varepsilon$ or $-\varepsilon$ with 50% of probability. The value of $+\varepsilon$ is to be in practice as small as possible (e.g. the machine epsilon). The procedure is repeated until a non-degenerate solution is obtained.

Figure 4 presents for the subsequent iteration of k-means when ε-Random is used for the degenerate solution of Section 2, considering $\varepsilon = 3$ (for illustrative purposes only).

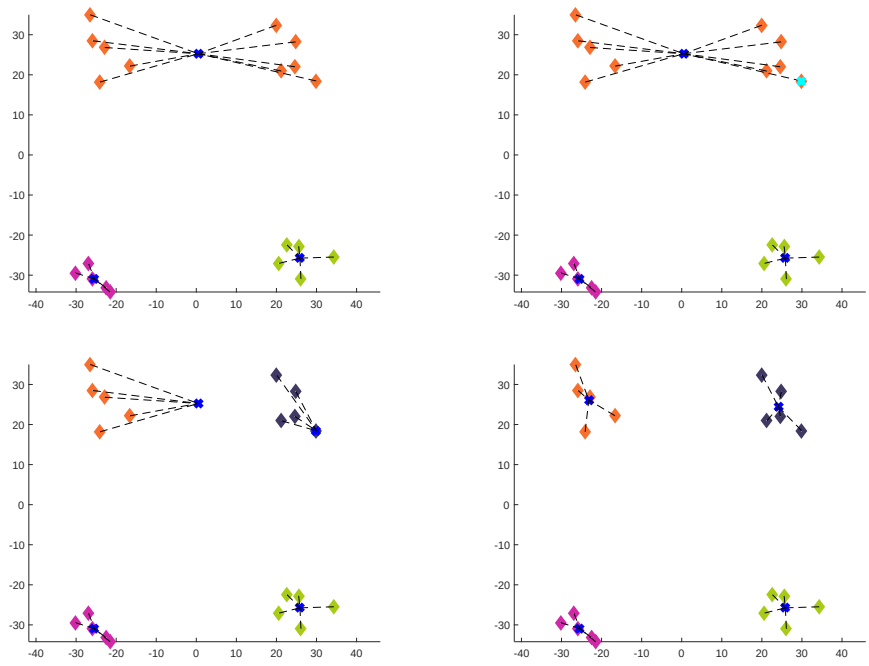

Figure 3: Application of strategy Greedy to the degenerate solution of Figure 1

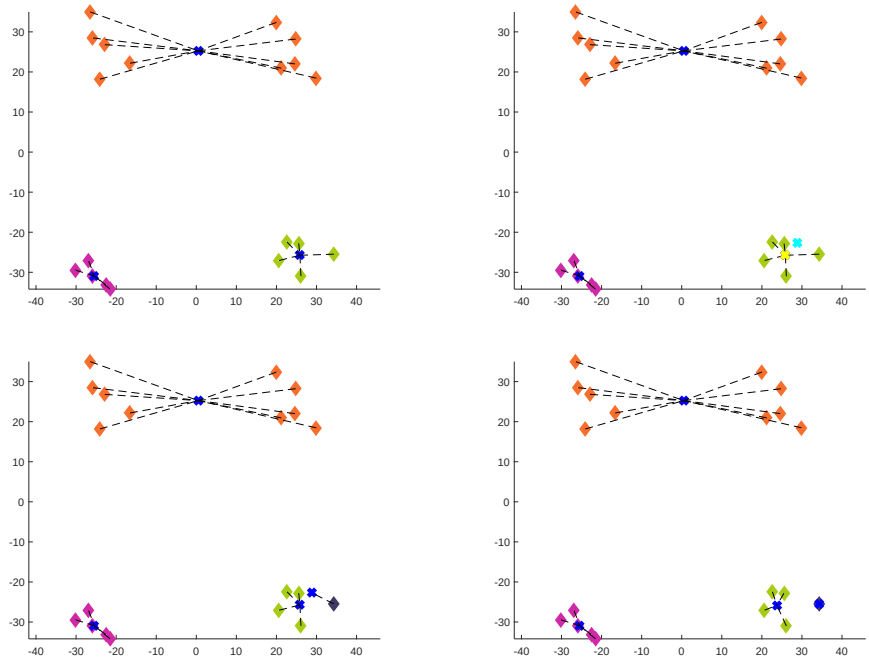

Figure 4: Application of strategy ε-Random to the degenerate solution of Figure 1

3.4 ϵ-Greedy

In this strategy, the new centroids are obtained from the centroids of the clusters having the largest MSSC values. This choice promotes the split of large clusters which are less likely to be homogeneous. Moreover, the perturbations for generating the new centroids are not done at random, but in the direction of the farthest entities from the selected centroids.

Let us denote y as the centroid of the cluster with the largest MSSC value and p^y as the coordinates of the farthest entity assigned to it. The coordinates of the new centroid y' are then given by:

$$y' = \epsilon p^y + (1 - \epsilon)y \tag{6}$$

Again, in practice, ϵ is set to a very small number (e.g. the machine epsilon).

Figure 5 illustrates the application of ϵ-Greedy to the degenerate solution of Section 2, considering $\epsilon = 0.2$ (for illustrative purposes only).

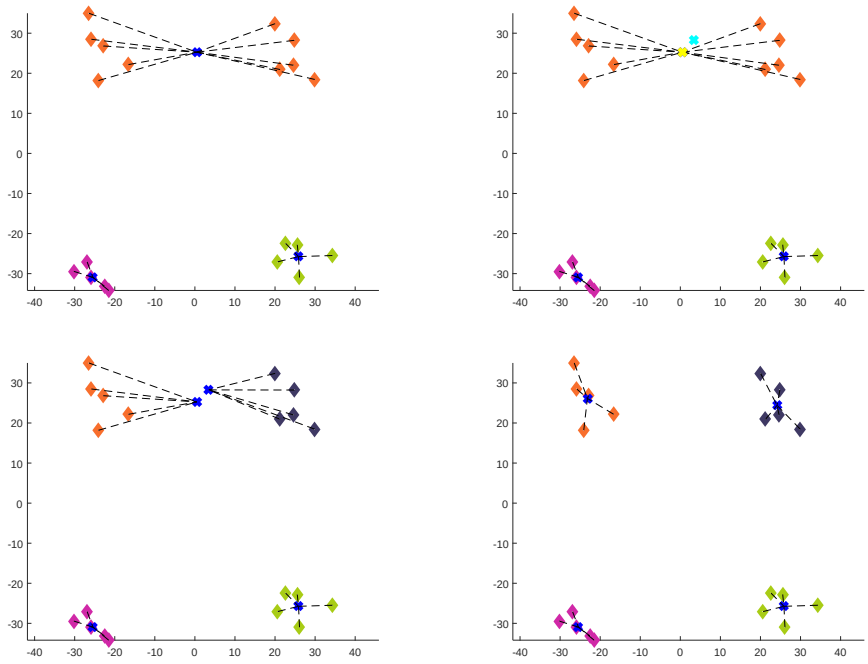

Figure 5: Application of strategy ϵ-Greedy to the degenerate solution of Figure 1

3.5 Mixed

This strategy combines the previous two. First, it makes a greedy decision by selecting the clusters with the largest MSSC values as in ϵ-Greedy, and second it performs a perturbation of $\pm\varepsilon$ at random as in ε-Random.

3.6 Discrete Bipartition - (DB)

This strategy considers the creation of d new centroids by performing bipartitions on the clusters having the largest MSSC values.

Given a set of points in general dimension, the MSSC is NP-hard even for two clusters. However, the optimal bipartition of the *discrete* version of the problem in which the cluster centers must be located at two of the given points can be solved by a simple enumeration. A straightforward implementation in time $O(n^3)$ with precomputed distances between each pair of points is given in Algorithm 1.

Algorithm 1 DB

input: cluster C composed of points $P = \{p_1, p_2, \ldots, p_{|C|}\}$;
Let $cost(i, j)$ denote the MSSC cost of the bipartition of P with cluster centers located in p_i and p_j
Let $cost^*$ denote the MSSC cost of the best bipartition found so far
$cost^* \leftarrow \infty$
for $i = 1, \ldots, |C| - 1$ **do**
  for $j = i + 1, \ldots, |C|$ **do**
    for $\ell = 1, \ldots, |C|$ **do**
      if $\|p_\ell - p_i\| \leq \|p_\ell - p_j\|$ **then**
        Assign p_ℓ to cluster center p_i;
      else
        Assign p_ℓ to cluster center p_j;
      end if
    end for
    if $cost(i, j) < cost^*$ **then**
      $cost^* \leftarrow cost(i, j)$;
      $i^* \leftarrow i; j^* \leftarrow i$;
    end if
  end for
end for
Perform the bipartition associated with i^*, j^*;

Figure 6 shows the next k-means iteration after that strategy is applied to the degenerate solution of Section 2. The light blue points correspond to the solution of the bipartition provided by Algorithm 1.

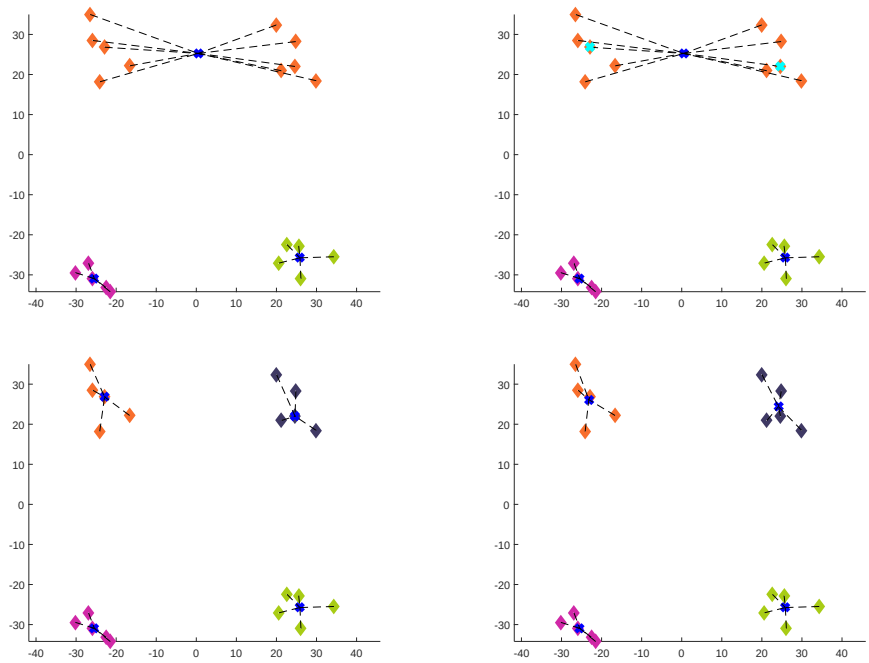

Figure 6: Application of strategy DB to the degenerate solution of Figure 1

4 Computational experiments

In this section, we present a series of experiments to assess the effectiveness of the different strategies proposed in the previous section to tackle degeneration in k-means. They can be applied either when k-means finishes its execution, in a *last improving* way, or as soon as degeneracy appears, in a *first improving* fashion. This differentiation is important. For instance, to the best of our knowledge, the Greedy strategy was never applied as soon as degeneracy occurs in k-means.

Our experiments consider 20 different runs of k-means from initial solutions which are known to lead to degenerate solutions. They are taken from random partitions of the n available data points into k clusters. The codes were implemented in Matlab and are available for download at.[1] The tests were performed on a Xeon(R) CPU X5650 2.67GHz and 62GB of RAM memory.

Table 3 reports results regarding the application of the classical k-means algorithm. They serve as benchmark to the results presented in the following tables. Regarding the 20 runs, column *avg.sol.* reports the average solution values obtained in each instance whereas column *#iter* shows the average number of k-means iterations and column *time* reports the average CPU times in seconds. Finally, column d_{max} refers to the maximum degree of degeneracy observed in the 20 distinct runs of k-means.

Table 3: k-means benchmark results

Instance	k	*avg.sol.*	*#iter*	*time*	d_{max}
Eilon	5	142.68	7.5	0.00	2
($n = 50$)	10	58.95	6.2	0.00	3
	15	39.53	6.0	0.00	4
	20	26.71	5.4	0.00	6
	25	19.46	4.7	0.00	7
Ruspini	5	10528.90	6.5	0.00	2
($n = 75$)	10	7378.85	6.3	0.00	6
	15	6226.26	6.6	0.00	10
	20	5617.79	6.8	0.00	15
	25	4543.16	7.1	0.00	17
	30	3816.75	6.0	0.00	18
Iris	5	52.36	10.4	0.00	2
($n = 150$)	10	39.64	13.7	0.01	5
	15	35.23	14.1	0.01	8
	20	29.56	13.7	0.01	11
	40	19.58	10.7	0.01	25
	50	17.60	9.9	0.01	31
	60	15.10	8.0	0.01	38
	70	12.06	8.2	0.01	34
Wine	5	918726.23	14.4	0.01	1
($n = 178$)	10	538874.95	16.4	0.01	2
	20	411222.20	16.2	0.02	6
	30	200997.43	23.4	0.03	9
	40	167113.87	16.4	0.03	12
	50	138635.35	11.5	0.02	18
	60	121802.38	9.8	0.02	23
	70	69040.03	9.2	0.03	21
	80	37198.99	9.1	0.03	26
B-Cancer	5	14195.17	19.1	0.04	1
($n = 699$)	10	11466.05	17.6	0.05	2
	20	9025.22	19.8	0.08	8
	50	7224.44	17.3	0.12	20
	100	6020.56	14.2	0.17	47
	200	4388.28	9.2	0.20	94
	300	3075.54	7.5	0.24	131
image segmentation	5	20656585.07	21.7	0.18	1
($n = 2310$)	10	11122078.21	25.8	0.28	3
	20	6081567.47	32.4	0.53	7
	50	3215416.44	33.8	1.09	25
	100	2601708.31	35.3	2.07	55
	500	1395120.19	21.8	5.97	284
	1000	781734.18	12.1	6.58	510

Table 4 presents for each instance and each degeneracy removal strategy:

- the average improvement (*%improv.*) over the average solutions obtained by the classical k-means heuristic (wrt column *avg.sol.* in Table 3);

[1] The actual address is ommited here in order to respect the double-blind review process

- the average deviation in the number of k-means iterations ($\Delta iter$) when the strategy is used (wrt column $\#iter$ in Table 3);
- the maximum degree of degeneracy (d_{max}) observed as a result of the 20 randomly initialized runs of k-means when the strategy is applied;
- the average additional computing ($\Delta time$ - in seconds) spent by the k-means algorithm when the strategy is applied (wrt column $time$ in Table 3).

The table is divided into two sets of columns, which are associated with the collected results when the degeneracy removal strategies are applied either in *first improving* or *last improving* ways. Whenever degeneracy does not appear at the end of a k-means run, the modified k-means which fixes degeneracy in a *last improving* basis cannot be applied. If that happens to all 20 different runs, the symbol '$*$' is used to represent that the *last improving* methodology was never used.

The value of $\varepsilon = 10^{-6}$ was used in strategies ε-Random and Mixed. The value of $\epsilon = 10^{-6}$ was used for strategy ϵ-Greedy.

Table 4: Comparison of the six fixing strategies

		first improving				*last improving*			
k	strategy	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$
(a)*Eilon* ($n = 50$)									
5	Random	0.24	-0.5	2	0.00	$*$	$*$	$*$	$*$
	Greedy	3.89	-0.0	2	0.00	$*$	$*$	$*$	$*$
	ε-Random	-1.34	-0.8	2	0.00	$*$	$*$	$*$	$*$
	ϵ-Greedy	**6.67**	-1.3	2	0.00	$*$	$*$	$*$	$*$
	Mixed	4.94	-1.5	2	0.00	$*$	$*$	$*$	$*$
	DB	6.31	-1.5	2	0.00	$*$	$*$	$*$	$*$
10	Random	0.83	0.8	3	0.00	2.28	0.7	3	0.00
	Greedy	3.11	-0.4	3	0.00	**2.73**	0.3	3	0.00
	ε-Random	0.36	0.1	3	0.00	0.52	0.2	3	0.00
	ϵ-Greedy	**4.57**	-0.5	3	0.00	2.26	0.2	3	0.00
	Mixed	3.39	-0.4	3	0.00	2.26	0.2	3	0.00
	DB	3.97	-0.7	3	0.00	2.26	0.2	3	0.00
15	Random	17.46	0.2	4	0.00	14.00	2.4	4	0.00
	Greedy	18.70	-0.2	4	0.00	20.89	1.7	4	0.00
	ε-Random	16.20	-0.2	4	0.00	9.04	1.6	4	0.00
	ϵ-Greedy	23.16	-0.7	4	0.00	29.86	1.6	4	0.00
	Mixed	22.47	-0.5	4	0.00	27.84	1.6	4	0.00
	DB	**26.02**	-0.7	4	0.00	**31.94**	1.7	4	0.00
20	Random	32.84	-0.1	6	0.00	25.36	2.7	6	0.00
	Greedy	34.07	-0.8	6	0.00	58.97	2.0	6	0.00
	ε-Random	25.09	-0.1	6	0.00	16.52	1.9	6	0.00
	ϵ-Greedy	47.55	-0.9	6	0.00	65.03	1.9	6	0.00
	Mixed	**51.39**	-0.8	6	0.00	**65.19**	2.0	6	0.00
	DB	50.55	-1.0	6	0.00	64.49	1.9	6	0.00
25	Random	50.32	0.7	7	0.00	39.94	3.5	7	0.00
	Greedy	**110.09**	-0.7	7	0.00	**137.12**	2.0	7	0.00
	ε-Random	31.71	-0.4	7	0.00	32.82	2.0	7	0.00
	ϵ-Greedy	100.81	-0.5	7	0.00	119.54	2.0	7	0.00
	Mixed	106.04	-0.4	7	0.00	112.39	2.0	7	0.00
	DB	105.80	-0.7	7	0.00	118.85	2.0	7	0.00
(b)*Ruspini* ($n = 75$)									
5	Random	-30.48	-0.3	2	0.00	$*$	$*$	$*$	$*$
	Greedy	**-3.79**	-0.8	2	0.00	$*$	$*$	$*$	$*$
	ε-Random	-48.18	-0.5	2	0.00	$*$	$*$	$*$	$*$
	ϵ-Greedy	-5.69	-1.2	2	0.00	$*$	$*$	$*$	$*$
	Mixed	-5.62	-1.2	2	0.00	$*$	$*$	$*$	$*$
	DB	-4.20	-1.2	2	0.00	$*$	$*$	$*$	$*$
10	Random	21.36	1.0	6	0.00	28.36	4.0	6	0.00
	Greedy	35.44	0.0	6	0.00	41.32	3.0	6	0.00

Table 4: Comparison of the six fixing strategies

		first improving				*last improving*			
k	strategy	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$
	ε-Random	17.53	1.1	6	0.00	29.18	3.8	6	0.00
	ϵ-Greedy	42.78	-0.2	6	0.00	47.18	2.6	6	0.00
	Mixed	43.03	0.0	6	0.00	49.27	2.3	6	0.00
	DB	**43.35**	-1.0	6	0.00	**49.41**	2.0	6	0.00
15	Random	67.09	1.0	10	0.00	73.07	4.9	10	0.00
	Greedy	94.15	-0.8	10	0.00	**121.85**	3.1	10	0.00
	ε-Random	39.11	0.2	10	0.00	63.67	4.0	10	0.00
	ϵ-Greedy	107.08	-1.5	10	0.00	114.04	2.1	10	0.00
	Mixed	104.62	-0.4	10	0.00	112.27	2.3	10	0.00
	DB	**113.50**	-1.8	10	0.00	117.52	2.0	10	0.00
20	Random	125.81	-0.1	15	0.00	121.81	6.1	15	0.01
	Greedy	**176.79**	-1.5	15	0.00	**182.78**	2.6	15	0.00
	ε-Random	100.08	-0.1	15	0.00	87.93	4.2	15	0.01
	ϵ-Greedy	172.55	-2.1	15	0.00	175.64	2.5	15	0.00
	Mixed	156.27	-1.3	15	0.00	161.46	3.4	15	0.01
	DB	172.16	-2.5	15	0.00	176.75	2.1	15	0.00
25	Random	159.11	-0.3	17	0.00	138.06	6.2	17	0.01
	Greedy	219.13	-1.9	17	0.00	**239.06**	3.0	17	0.01
	ε-Random	101.99	-0.6	17	0.00	125.01	4.8	17	0.01
	ϵ-Greedy	214.08	-2.0	17	0.00	217.77	2.6	17	0.01
	Mixed	202.36	-1.7	17	0.00	200.38	3.3	17	0.01
	DB	**223.72**	-2.8	17	0.00	232.33	2.1	17	0.00
30	Random	182.47	0.7	18	0.00	173.18	6.1	18	0.01
	Greedy	**297.37**	-1.2	18	0.00	**311.88**	2.9	18	0.01
	ε-Random	115.42	-0.1	18	0.00	139.44	5.1	18	0.01
	ϵ-Greedy	260.45	-1.3	18	0.00	270.44	2.5	18	0.01
	Mixed	257.00	-0.8	18	0.00	265.27	3.0	18	0.01
	DB	285.63	-1.6	18	0.00	305.08	2.1	18	0.00
(c)*Iris* ($n = 150$)									
5	Random	1.21	0.1	2	0.00	*	*	*	*
	Greedy	1.53	-1.9	2	0.00	*	*	*	*
	ε-Random	1.98	0.4	2	0.00	*	*	*	*
	ϵ-Greedy	0.75	-2.8	2	0.00	*	*	*	*
	Mixed	**2.58**	-0.4	2	0.00	*	*	*	*
	DB	-0.36	-3.7	2	0.00	*	*	*	*
10	Random	24.21	-3.1	5	0.00	13.86	4.2	5	0.01
	Greedy	**37.81**	-2.9	5	0.00	17.00	3.9	5	0.01
	ε-Random	11.94	-2.2	5	0.00	6.22	2.8	5	0.01
	ϵ-Greedy	35.44	-3.8	5	0.00	17.02	2.5	5	0.01
	Mixed	33.25	-3.1	5	0.00	**17.05**	3.6	5	0.01
	DB	32.95	-2.7	5	0.01	17.04	2.1	5	0.01
15	Random	51.68	-4.7	8	0.00	47.65	6.3	8	0.01
	Greedy	**65.98**	-3.3	8	0.00	**61.75**	7.8	8	0.02
	ε-Random	35.05	-2.4	8	0.00	23.78	6.5	8	0.01
	ϵ-Greedy	55.32	-5.7	8	0.00	57.78	3.8	8	0.01
	Mixed	57.99	-4.9	8	0.00	58.09	5.6	8	0.01
	DB	52.41	-6.9	8	0.00	58.85	2.3	8	0.01
20	Random	55.98	-4.5	11	0.00	53.17	6.0	11	0.01
	Greedy	**67.63**	-3.3	11	0.00	**67.18**	7.0	11	0.02
	ε-Random	35.53	-4.1	11	0.00	22.94	4.7	11	0.01
	ϵ-Greedy	53.93	-5.8	11	0.00	59.63	3.7	11	0.01
	Mixed	60.25	-5.0	11	0.00	62.31	4.5	11	0.01
	DB	57.38	-6.8	11	0.00	65.07	3.3	11	0.01
40	Random	76.66	-2.5	25	0.00	78.95	8.3	25	0.02
	Greedy	122.44	-3.5	25	0.00	129.81	3.4	25	0.02
	ε-Random	43.59	-1.6	25	0.01	56.56	5.7	25	0.02
	ϵ-Greedy	115.74	-4.3	25	0.00	127.24	3.2	25	0.02
	Mixed	119.20	-3.7	25	0.00	121.44	4.0	25	0.02
	DB	**126.08**	-4.3	25	0.00	**139.20**	3.1	25	0.02

Table 4: Comparison of the six fixing strategies

		first improving				*last improving*			
k	strategy	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$
50	Random	115.89	-2.1	31	0.00	102.29	8.6	31	0.03
	Greedy	**181.31**	-3.3	31	0.00	192.19	3.2	31	0.02
	ε-Random	64.44	-1.8	31	0.01	83.75	6.1	31	0.02
	ϵ-Greedy	157.51	-3.7	31	0.00	178.96	3.2	31	0.02
	Mixed	160.92	-3.6	31	0.00	170.96	3.8	31	0.02
	DB	177.66	-4.0	31	0.00	**199.20**	2.7	31	0.02
60	Random	117.06	-1.0	38	0.01	116.94	8.1	38	0.03
	Greedy	**218.23**	-2.0	38	0.01	**246.66**	3.0	38	0.02
	ε-Random	69.36	-0.8	38	0.01	78.13	7.2	38	0.03
	ϵ-Greedy	200.99	-2.6	38	0.00	210.50	2.9	38	0.02
	Mixed	187.87	-2.2	38	0.01	195.26	4.1	38	0.02
	DB	208.06	-3.0	38	0.00	232.61	2.2	38	0.02
70	Random	100.06	-0.9	34	0.01	102.05	8.7	34	0.03
	Greedy	**236.08**	-2.6	34	0.00	**258.55**	2.3	34	0.02
	ε-Random	56.59	-1.2	34	0.01	70.86	6.7	34	0.03
	ϵ-Greedy	199.98	-3.2	34	0.00	231.44	2.4	34	0.02
	Mixed	197.76	-2.5	34	0.01	199.39	3.2	34	0.02
	DB	221.45	-3.4	34	0.00	241.94	2.0	34	0.02
(d) *Wine* ($n = 178$)									
5	Random	-4.96	-2.8	1	0.00	*	*	*	*
	Greedy	-2.45	-6.2	1	0.00	*	*	*	*
	ε-Random	-1.74	-3.2	1	0.00	*	*	*	*
	ϵ-Greedy	**0.25**	-6.9	1	0.00	*	*	*	*
	Mixed	**0.25**	-6.8	1	0.00	*	*	*	*
	DB	**0.25**	-7.1	1	0.00	*	*	*	*
10	Random	2.56	-2.5	2	0.01	*	*	*	*
	Greedy	**91.77**	0.8	2	0.02	*	*	*	*
	ε-Random	1.39	-0.3	2	0.01	*	*	*	*
	ε-Greedy	-0.74	-2.2	2	0.01	*	*	*	*
	ε-Mixed	-0.74	-2.2	2	0.01	*	*	*	*
	DB	-0.74	-2.1	2	0.01	*	*	*	*
20	Random	18.62	-2.7	6	0.00	35.01	4.4	6	0.02
	Greedy	**133.70**	-4.2	6	0.00	124.46	2.9	6	0.02
	ε-Random	0.93	-0.8	6	0.01	24.10	4.0	6	0.02
	ϵ-Greedy	69.60	-3.3	6	0.00	129.78	2.0	6	0.01
	Mixed	71.06	-3.5	6	0.00	**129.83**	2.1	6	0.01
	DB	74.34	-3.4	6	0.01	129.79	2.2	6	0.01
30	Random	23.73	-7.4	9	0.00	50.51	6.6	9	0.03
	Greedy	**103.68**	-13.3	9	-0.01	**92.83**	4.6	9	0.03
	ε-Random	13.25	-0.8	9	0.02	23.16	4.0	9	0.03
	ϵ-Greedy	64.97	-16.0	9	-0.02	86.52	2.7	9	0.02
	Mixed	71.69	-15.5	9	-0.02	87.43	2.8	9	0.02
	DB	67.46	-15.5	9	-0.01	86.61	2.2	9	0.02
40	Random	81.01	-4.3	12	0.00	85.07	7.8	12	0.04
	Greedy	**208.07**	-8.3	12	-0.01	**202.13**	4.4	12	0.03
	ε-Random	16.89	-2.3	12	0.01	46.37	4.8	12	0.03
	ϵ-Greedy	175.52	-9.6	12	-0.01	176.31	3.2	12	0.02
	Mixed	173.21	-9.7	12	-0.01	176.09	3.7	12	0.03
	DB	178.23	-9.9	12	-0.01	182.54	2.5	12	0.02
50	Random	105.68	-3.3	18	0.00	109.51	7.6	18	0.04
	Greedy	**344.38**	-5.7	18	-0.01	**338.85**	6.2	18	0.04
	ε-Random	28.41	-0.5	18	0.01	73.17	5.4	18	0.03
	ϵ-Greedy	299.81	-5.3	18	0.00	300.66	3.4	18	0.03
	Mixed	293.06	-5.0	18	0.00	291.00	4.2	18	0.03
	DB	320.14	-5.1	18	0.00	321.77	4.1	18	0.03
60	Random	133.25	-1.2	23	0.01	150.84	7.8	23	0.04
	Greedy	**584.66**	-3.5	23	0.00	544.24	5.3	23	0.03
	ε-Random	51.66	1.0	23	0.02	63.02	5.2	23	0.03
	ϵ-Greedy	524.00	-4.2	23	0.00	520.42	2.4	23	0.02
	Mixed	505.43	-2.9	23	0.00	503.70	3.4	23	0.03

Table 4: Comparison of the six fixing strategies

		first improving				*last improving*			
k	strategy	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$
	DB	552.00	-4.4	23	0.00	**561.90**	3.2	23	0.03
70	Random	87.62	-1.1	21	0.01	115.36	8.6	21	0.05
	Greedy	**471.88**	-3.2	21	0.00	**479.00**	2.8	21	0.03
	ε-Random	53.70	0.8	21	0.02	53.57	3.9	21	0.03
	ϵ-Greedy	416.84	-4.0	21	0.00	462.69	2.8	21	0.03
	Mixed	406.39	-3.7	21	0.00	435.25	3.4	21	0.03
	DB	432.48	-4.0	21	0.00	450.28	2.7	21	0.03
80	Random	110.21	-1.4	26	0.01	61.18	7.7	26	0.06
	Greedy	**370.37**	-3.6	26	0.00	**400.32**	2.1	26	0.03
	ε-Random	47.27	-0.8	26	0.01	52.20	3.7	26	0.03
	ϵ-Greedy	311.48	-4.1	26	-0.01	343.01	2.2	26	0.03
	Mixed	303.42	-3.6	26	0.00	328.25	3.1	26	0.03
	DB	320.97	-4.2	26	-0.01	357.99	2.1	26	0.03
(e) *B-Cancer* ($n = 699$)									
5	Random	-0.13	0.4	1	0.03	*	*	*	*
	Greedy	-0.68	-1.7	1	0.02	*	*	*	*
	ε-Random	**0.14**	-1.4	1	0.02	*	*	*	*
	ϵ-Greedy	-0.62	-2.2	1	0.02	*	*	*	*
	Mixed	-0.30	-2.8	1	0.02	*	*	*	*
	DB	0.02	-2.3	1	0.09	*	*	*	*
10	Random	-0.50	-0.2	2	0.03	*	*	*	*
	Greedy	0.99	-2.4	2	0.02	*	*	*	*
	ε-Random	-0.92	0.1	2	0.03	*	*	*	*
	ϵ-Greedy	0.83	0.2	2	0.03	*	*	*	*
	Mixed	**1.06**	-0.9	2	0.03	*	*	*	*
	DB	0.66	0.4	2	0.26	*	*	*	*
20	Random	-2.63	-2.2	8	0.03	*	*	*	*
	Greedy	**2.48**	-1.4	8	0.04	*	*	*	*
	ε-Random	-0.91	-3.4	8	0.02	*	*	*	*
	ϵ-Greedy	1.67	-0.3	8	0.04	*	*	*	*
	Mixed	-0.18	-1.7	8	0.03	*	*	*	*
	DB	0.67	1.0	8	0.44	*	*	*	*
50	Random	9.12	-3.2	20	0.03	7.30	8.0	20	0.15
	Greedy	**19.74**	-4.5	20	0.02	13.66	5.9	20	0.13
	ε-Random	6.65	-3.1	20	0.03	6.44	6.3	20	0.13
	ϵ-Greedy	15.91	-4.9	20	0.02	14.83	5.1	20	0.12
	Mixed	16.05	-5.6	20	0.01	14.59	6.8	20	0.14
	DB	14.73	-6.7	20	0.18	**14.94**	5.1	20	0.16
100	Random	21.60	-3.3	47	0.02	22.26	8.8	47	0.27
	Greedy	**55.12**	-4.3	47	0.02	**50.86**	4.4	47	0.18
	ε-Random	20.60	-2.8	47	0.04	24.19	8.6	47	0.27
	ϵ-Greedy	43.77	-5.9	47	-0.02	46.34	5.5	47	0.21
	Mixed	44.19	-5.2	47	0.00	43.07	5.8	47	0.21
	DB	46.21	-5.6	47	0.02	48.96	4.5	47	0.20
200	Random	47.55	-0.7	94	0.08	44.61	14.0	94	0.66
	Greedy	**135.86**	-2.0	94	0.04	**146.67**	2.5	94	0.21
	ε-Random	46.36	-0.4	94	0.10	43.51	9.8	94	0.50
	ϵ-Greedy	106.24	-3.0	94	0.00	127.47	2.8	94	0.23
	Mixed	97.33	-2.3	94	0.03	103.35	4.5	94	0.30
	DB	117.15	-3.0	94	0.01	133.90	2.7	94	0.22
300	Random	86.44	4.6	131	0.39	81.55	22.9	131	1.46
	Greedy	**454.92**	-1.5	131	0.05	**414.09**	2.0	131	0.25
	ε-Random	100.87	0.3	131	0.15	88.91	10.8	131	0.76
	ϵ-Greedy	338.24	-2.5	131	-0.01	336.26	2.2	131	0.26
	Mixed	288.13	-1.8	131	0.03	281.04	4.4	131	0.39
	DB	352.54	-2.6	131	-0.02	351.25	2.1	131	0.25

Table 4: Comparison of the six fixing strategies

		first improving				*last improving*			
k	strategy	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$	*%improv.*	$\Delta iter$	d_{max}	$\Delta time$
(f)*image segmentation* ($n = 2310$)									
5	Random	-2.08	0.5	1	0.11	*	*	*	*
	Greedy	**18.43**	-3.1	1	0.06	*	*	*	*
	ε-Random	-4.33	-1.7	1	0.08	*	*	*	*
	ϵ-Greedy	11.85	-2.9	1	0.07	*	*	*	*
	Mixed	-1.74	2.1	1	0.13	*	*	*	*
	DB	-2.92	-0.5	1	4.09	*	*	*	*
10	Random	-0.10	-2.8	3	0.10	*	*	*	*
	Greedy	**5.73**	5.4	3	0.26	*	*	*	*
	ε-Random	0.64	0.2	3	0.16	*	*	*	*
	ϵ-Greedy	1.64	-0.6	3	0.15	*	*	*	*
	Mixed	1.30	-2.4	3	0.12	*	*	*	*
	DB	0.37	0.4	3	3.12	*	*	*	*
20	Random	-0.83	-4.3	7	0.14	*	*	*	*
	Greedy	**6.69**	-7.0	7	0.08	*	*	*	*
	ε-Random	-1.83	-4.4	7	0.15	*	*	*	*
	ϵ-Greedy	1.01	-4.0	7	0.17	*	*	*	*
	Mixed	0.01	-5.9	7	0.11	*	*	*	*
	DB	-0.01	-4.3	7	2.87	*	*	*	*
50	Random	5.42	-8.1	25	0.07	*	*	*	*
	Greedy	**20.56**	-7.6	25	0.11	*	*	*	*
	ε-Random	0.45	-5.1	25	0.23	*	*	*	*
	ϵ-Greedy	16.57	-7.4	25	0.14	*	*	*	*
	Mixed	16.35	-9.8	25	0.00	*	*	*	*
	DB	11.14	-7.7	25	1.77	*	*	*	*
100	Random	30.47	-13.8	55	-0.42	28.28	19.0	55	2.63
	Greedy	**86.43**	-9.3	55	0.05	**76.89**	11.2	55	1.92
	ε-Random	23.29	-11.5	55	-0.19	18.94	17.4	55	2.50
	ϵ-Greedy	80.83	-11.6	55	-0.18	39.17	15.2	55	2.33
	Mixed	78.85	-14.4	55	-0.43	37.76	15.1	55	2.31
	DB	79.47	-15.6	55	0.69	39.16	12.6	55	2.14
500	Random	90.74	-8.2	284	-1.19	100.67	17.7	284	10.15
	Greedy	496.85	-9.7	284	-1.64	503.83	10.5	284	6.98
	ε-Random	103.40	-2.4	284	1.49	131.78	25.3	284	13.19
	ϵ-Greedy	500.14	-10.7	284	-2.00	536.97	8.6	284	6.15
	Mixed	506.11	-10.4	284	-1.79	533.90	12.1	284	7.54
	DB	**517.49**	-11.9	284	-2.44	**562.47**	5.8	284	4.69
1000	Random	90.70	-2.6	510	0.24	87.22	18.6	510	18.03
	Greedy	1151.53	-4.3	510	-0.99	1268.46	4.6	510	6.44
	ε-Random	95.92	-1.7	510	1.26	156.98	15.6	510	16.05
	ϵ-Greedy	1150.80	-5.0	510	-1.72	1279.17	3.7	510	5.66
	Mixed	1067.73	-4.8	510	-1.58	1153.41	6.4	510	7.71
	DB	**1200.68**	-5.8	510	-2.43	**1334.56**	3.6	510	5.44

The results in Table 4 reveal that:

- The average number of iterations of the modified k-means which fixes degeneracy in a *first improving* basis is usually smaller than if degeneracy is fixed in a *last improving* basis. Furthermore, fixing degeneracy as soon as it occurs very often causes a reduction in the number of k-means iterations as represented by the negative values in column $\Delta iter$. As seen in many values shown in column *time*, whenever that reduction is large enough, the overall CPU time of the modified k-means with *first improving* is smaller than the CPU time spent by the classical k-means heuristic.
- Since the modified k-means that fixes degeneracy in a *last improving* basis does a larger number of iterations than the classical version, more computing time is required for heuristic convergence. For example, for the *image segmentation* data set with $k = 1000$, the average additional time of an execution of k-means using the Random strategy is approximately 18 seconds. Since the computational complexity

of k-means is a nondecreasing function in n and k, the additional times yielded by fixing degeneracy also increase with these two parameters.

- The maximum degree of degeneracy (column d_{max}) is observed to be the same regardless of the strategy used. This is due to the fact that the largest degrees are found right after the initialization step which randomly allocates data points to clusters.
- Degeneracy fixing strategies lead to better solutions than those obtained by the classical k-means algorithm in the majority of the tested cases ($\approx 88\%$) for *first improving* strategies, and in 100% of the cases that *last improving* strategies could be applied. Moreover, the improvements yielded from degeneracy fixing strategies might be quite large, attaining more than 1000% for example in data set *image segmentation* with $k = 1000$. We observe that the higher the degree of degeneracy for each data set, the larger are the average improvements yielded by the different proposed strategies.
- k-means is more likely to fix degeneracy alone when k is small. For 12 out of 42 instances, k-means was able to remove degeneracy automatically in all 20 evaluated runs. Even in these cases, the *first improvement* strategies lead to solutions in average 1.79% better than those obtained by k-means alone. Besides, the modified k-means with those strategies is very often faster than the classical method.
- Comparison between *first improving* and *last improving* methodologies are possible in 30 out of 42 instances. Statistical tests t for paired observations (Jain, 2008) made it possible to conclude with 90% confidence that, except for the Random strategy, fixing degeneracy through a *last improving* methodology in k-means yields better solutions than those obtained by fixing degeneracy as soon as it occurs. However, we notice from the table that in general fixing degeneracy as soon as it occurs has very minor impact over the CPU time spent by the classical k-means algorithm (in some cases, the CPU time is even reduced). Consequently, the *first improving* methodology should be preferable over the *last improving* one for practical purposes in which k-means is applied several times from different random initial solutions.
- Strategies based on greedy decisions lead to larger improvements than those whose decisions are taken at random. For instance, Greedy is always better than Random except for one single case for data set *B-cancer* with $k = 5$ using the *first improving* methodology. Strategy ϵ-Greedy is better than ε-Random 96.4% of the tested instances. Regarding the mixed approach Mixed, it is almost always better than ε-Random (i.e., in 97.6% of the tested cases), while it is outperformed by ϵ-Greedy in approximately 70% of the cases.
- The six strategies are compared in each instance for each degeneracy removal methodology, totalizing 84 tested cases. Points are given based on the Formula 1 scoring rules used between 1991 and 2002. This means that the six ranks received a descending number of points $(10, 6, 4, 3, 2, 1)$. The table below presents the overall score summed over all tested cases, which indicates the superiority of the Greedy and DB strategies.

strategy	score
Random	146
Greedy	565
ε-Random	102
ϵ-Greedy	325
Mixed	312
DB	438

The Greedy strategy is simple and quickly computed. The second best strategy is DB which might be certainly cumbersome for k-means, specially for clusters containing a large number of points. However, when clusters are small (which is likely the case when k is large), the DB strategy applied in *first improving* way was able to even decrease the total time of the classical k-means algorithm (see results for $k = 500$ and $k = 1000$ for the *image segmentation* data set).

Although not reported in Table 4, we observed that type-2 degeneration are rare in k-means. In fact, type-2 degeneration happened in our tests only at the initialization, and that, only for the data sets that contained repeated data points.

Our last set of experiments aims to compare the effectiveness of our approach versus different initialization procedures used in the literature to improve k-means results. They consist of rules for finding points to be used as initial cluster centers for k-means. The first one is the *k-means++* initialization method of Arthur and Vassilvitskii (2007) presented in Algorithm 2. The rationale behind *k-means++* initialization is that clusters are usually spread out in a good solution.

Algorithm 2 k-means++

input: points $P = \{p_1, \ldots, p_n\}$;
Let $i^* \in \{1, \ldots, n\}$;
$y_1 \leftarrow p_{i^*}$, where i^* is randomly chosen;
for $j = 2, \ldots, k$ **do**
  for $i = 1, \ldots, n$ **do**
    $D(i) \leftarrow \min_{\ell \in \{1,\ldots,j-1\}} \{\|p_i - y_\ell\|^2\}$;
  end for
  $y_j \leftarrow p_{i^*}$, where i^* is selected with probability proportional to $D(i^*)$;
end for
return y

The second initialization method makes use of the Ward's method (Ward Jr, 1963) to provide initial cluster centers for the k-means algorithm. Ward's method merges at each step of its hierarchical construction the two clusters that lead to the minimum increase in the MSSC objective function after merging. The desired solution is then obtained by cutting the resulting tree at k clusters. Helsen and Green (1991) proposed to execute Ward's method from different samples of points from the dataset, each time generating different cluster centers for k-means initialization.

Table 5 summarizes the best results reported in Table 4 and compares them with average results obtained by k-means when initialized with cluster centers provided by *k-means++* and the method of Helsen and Green (1991), denoted *R-Ward*, in 20 runs. For the latter, samples are built using 50% of the data points, so that $n \geq k$ in all runs.

We observe that the degeneracy fixing strategies produced the best results in 16 of the 42 tested instances, i.e., in approximately 38% of the cases. This is an expressive number since the degeneracy fixing strategies can also be used within other memory-based clustering methods (e.g. Carrizosa et al. (2014); Hansen et al. (2005)), for which restarting the search from a complete new solution obtained by any initialization method is not efficient or desirable.

5 Concluding remarks

The k-means algorithm is widely used in many different domains. A possible event in a k-means execution is the appearance of degenerate solutions in which one or more clusters are empty. In this paper, we analyzed in detail the importance of tackling degenerate solutions that appear during and after the execution of the k-means algorithm, both in terms of efficiency and the quality of the solutions obtained. With this objective, we performed an extensive comparison of strategies to eliminate degeneracy: four of them new to the literature, testing them as soon as degeneracy occurs and after k-means convergence. In the vast majority of the cases, the strategies were able to improve the quality of the solutions obtained by the classical k-means heuristic in the presence of degeneration, attaining more than 1000% in one of our tested instances. In particular, the achieved improvements are larger for k-means runs leading to degenerate solutions with large degrees of degeneracy.

Furthermore, we observed that the moment in which degeneracy is handled also influences the k-means overall performance. In general, treating degeneracy as soon as it occurs has very minor impact over the overall CPU time spent by a k-means execution. Another important characteristic of the degeneracy fixing strategies is that they are easy to implement and to be incorporated in any k-means implementation, being a flexible alternative to reinitialization procedures within memory-based clustering methods.

Table 5: Summary of the best results obtained by k-means with the degeneracy fixing strategies and with the k-means++ and R-Ward initialization procedures

		k-means++		*R-Ward*		*first improv. (best)*			*last improv. (best)*		
Instance	k	*%improv.*	Δ*time*	*%improv.*	Δ*time*	*%improv.*	Δ*time*	*strategy*	*%improv.*	*time*	*strategy*
Eilon	5	5.92	0.00	**7.32**	0.00	6.67	0.00	ϵ-Greedy	*	*	*
($n = 50$)	10	11.21	0.00	**20.35**	0.00	4.57	0.00	ϵ-Greedy	2.73	0.00	Greedy
	15	40.37	0.00	**49.50**	0.00	26.02	0.00	DB	31.94	0.00	DB
	20	58.12	0.00	50.22	0.00	51.39	0.00	Mixed	**65.19**	0.00	Mixed
	25	98.92	0.00	43.44	0.00	110.09	0.00	Greedy	**137.12**	0.00	Greedy
Ruspini	5	-3.76	0.00	**-2.35**	0.00	-3.79	0.00	Greedy	*	*	*
($n = 75$)	10	30.28	0.00	42.76	0.00	43.35	0.00	DB	**49.41**	0.00	DB
	15	94.31	0.00	114.31	0.00	113.50	0.00	DB	**121.85**	0.00	Greedy
	20	154.53	0.00	164.75	0.00	176.79	0.00	Greedy	**182.78**	0.00	Greedy
	25	206.43	0.00	190.48	0.00	223.72	0.00	DB	**239.06**	0.01	Greedy
	30	256.59	0.00	230.04	0.00	297.37	0.00	Greedy	**311.88**	0.01	Greedy
Iris	5	5.83	0.00	**8.98**	0.01	2.58	0.00	Mixed	*	*	*
($n = 150$)	10	42.34	0.00	**47.67**	0.01	37.81	0.00	Greedy	17.05	0.01	Mixed
	15	67.77	0.00	**77.03**	0.00	65.98	0.00	Greedy	61.75	0.02	Greedy
	20	81.86	0.00	**92.04**	0.00	67.63	0.00	Greedy	67.18	0.02	Greedy
	40	132.37	0.00	133.89	0.00	126.08	0.00	DB	**139.20**	0.02	DB
	50	183.68	0.01	175.15	0.00	181.31	0.00	Greedy	**199.20**	0.02	DB
	60	209.12	0.01	184.72	0.00	218.23	0.00	Greedy	**246.66**	0.02	Greedy
	70	236.08	0.01	169.83	0.00	236.08	0.00	Greedy	**258.55**	0.02	Greedy
Wine	5	-7.06	0.00	-4.51	0.01	**0.25**	0.00	ϵ-Gr/Mixed/DB	*	*	*
($n = 178$)	10	111.64	0.00	**128.03**	0.01	91.77	0.02	Greedy	*	*	*
	20	368.15	-0.01	**392.37**	0.00	133.70	0.00	Greedy	129.83	0.01	Mixed
	30	**313.10**	-0.02	279.64	-0.02	103.68	-0.01	Greedy	92.83	0.03	Greedy
	40	**460.04**	-0.01	362.79	-0.01	208.07	-0.01	Greedy	202.13	0.03	Greedy
	50	**604.74**	0.00	417.92	0.00	344.38	-0.01	Greedy	338.85	0.04	Greedy
	60	**790.26**	0.01	477.70	0.00	584.66	0.00	Greedy	561.90	0.03	DB
	70	**585.82**	0.01	317.57	0.00	471.88	0.00	Greedy	479.00	0.03	Greedy
	80	391.50	0.01	114.16	0.00	370.37	0.00	Greedy	**400.32**	0.03	Greedy
B-Cancer	5	-0.41	0.00	-0.40	0.19	**0.14**	0.02	ε-Random	*	*	*
($n = 699$)	10	3.60	0.01	**7.09**	0.19	1.06	0.03	Mixed	*	*	*
	20	6.88	0.00	**10.27**	0.16	2.48	0.04	Greedy	*	*	*
	50	29.51	0.01	**35.22**	0.12	19.74	0.02	Greedy	14.94	0.16	DB
	100	69.48	0.05	**79.27**	0.09	55.12	0.02	Greedy	50.86	0.18	Greedy
	200	**199.93**	0.18	125.09	0.06	135.86	0.04	Greedy	146.67	0.21	Greedy
	300	**536.39**	0.27	66.14	0.06	454.92	0.05	Greedy	414.09	0.25	Greedy
image segmentation	5	12.42	-0.03	**19.10**	5.32	18.43	0.06	Greedy	*	*	*
($n = 2310$)	10	4.06	0.00	**10.54**	5.23	5.73	0.26	Greedy	*	*	*
	20	7.79	-0.11	**14.18**	4.93	6.69	0.08	Greedy	*	*	*
	50	**31.17**	-0.17	30.14	4.44	20.56	0.11	Greedy	*	*	*
	100	**103.31**	-0.62	83.07	3.33	86.43	0.05	Greedy	76.89	1.92	Greedy
	500	551.28	-1.63	138.74	-1.46	517.49	-2.44	DB	**562.47**	4.69	DB
	1000	1149.68	0.49	66.35	-1.70	1200.68	-2.43	DB	**1334.56**	5.44	DB

References

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. Machine Learning, 75:245–249.

Aloise, D., Hansen, P., and Liberti, L. (2012). An improved column generation algorithm for minimum sum-of-squares clustering. Mathematical Programming, 131(1-2):195–220.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In 2007 ACM-SIAM Symposium on Discrete Algorithms (SODA'07), 1027–1035.

Blanchard, S. J., Aloise, D., and DeSarbo, W. S. (2012). The heterogeneous p-median problem for categorization based clustering. Psychometrika, 77(4):741–762.

Bradley, P. S. and Fayyad, U. M. (1998). Refining initial points for k-means clustering. In ICML, 98:91–99.

Brimberg, J. and Mladenović, N. (1999). Degeneracy in the multi-source weber problem. Mathematical programming, 85(1):213–220.

Brusco, M. J. and Steinley, D. (2007). A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning. Psychometrika, 72(4):583–600.

Carrizosa, E., Alguwaizani, A., Hansen, P., and Mladenović, N. (2014). New heuristic for harmonic means clustering. Journal of Global Optimization, 1–17.

Choromanska, A. and Monteleoni, C. (2012). Online clustering with experts. In International Conference on Artificial Intelligence and Statistics, 227–235.

Cooper, L. (1964). Heuristic methods for location-allocation problems. Siam Review, 6(1):37–53.

Ding, Y., Zhao, Y., Shen, X., Musuvathi, M., and Mytkowicz, T. (2015). Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In 32nd International Conference on Machine Learning (ICML-15), 579–587.

Eilon, S., Watson-Gandy, C., and Christofides, N. (1971). Distributed management. Hafner, New York.

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. Biometrics, 21:768.

Hansen, P. and Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. Pattern Recognition, 34:405–413.

Hansen, P., Ngai, E., Cheung, B. K., and Mladenovic, N. (2005). Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering. Journal of classification, 22(2):287–310.

Haverly, C. A. (1978). Studies of the behavior of recursion for the pooling problem. ACM SIGMAP Bulletin, (25):19–28.

Helsen, K. and Green, P. E. (1991). A computational study of replicated clustering with an application to market segmentation. Decision Sciences, 22(5):1124–1141.

Hofmans, J., Ceulemans, E., Steinley, D., and Van Mechelen, I. (2015). On the added value of bootstrap analysis for k-means clustering. Journal of Classification, 32(2):268–284.

Inaba, M., Katoh, N., and Imai, H. (1994). Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In Proceedings of the 10th ACM Symposium on Computational Geometry, 332–339.

Jain, R. (2008). The art of computer systems performance analysis. John Wiley & Sons.

Lichman, M. (2013). UCI machine learning repository.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of 5^{th} Berkeley Symposium on Mathematical Statistics and Probability, 2:281–297, Berkeley, CA.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is NP-hard. Lecture Notes in Computer Science, 5431:274–285.

Mairal, J., Bach, F., and Ponce, J. (2012). Sparse modeling for image and vision processing. Foundations and Trends in Computer Graphics and Vision, 8(2–3):85–283.

Mak, J. N. and Wolpaw, J. R. (2009). Clinical applications of brain-computer interfaces: current state and future prospects. Biomedical Engineering, IEEE Reviews in, 2:187–199.

Nugent, R., Dean, N., and Ayers, E. (2010). Skill set profile clustering: the empty k-means algorithm with automatic specification of starting cluster centers. In Educational Data Mining 2010, 151–160.

Ordin, B. and Bagirov, A. M. (2015). A heuristic algorithm for solving the minimum sum-of-squares clustering problems. Journal of Global Optimization, 61(2):341–361.

Pacheco, J. and Valencia, O. (2003). Design of hybrids for the minimum sum-of-squares clustering problem. Computational Statistics & Data Analysis, 43(2):235–248.

Ruspini, E. (1970). Numerical method for fuzzy clustering. Information Sciences, 2:319–350.

Steinley, D. (2006). K-means clustering: A half-century synthesis. British Journal of Mathematical and Statistical Psychology, 59(1):1–34.

Steinley, D. and Brusco, M. J. (2007). Initializing k-means batch clustering: A critical evaluation of several techniques. Journal of Classification, 24(1):99–121.

Tao, P. D. et al. (2014). New and efficient dca based algorithms for minimum sum-of-squares clustering. Pattern Recognition, 47(1):388–401.

Teboulle, M. (2007). A unified continous optimization framework for center-based clustering methods. Journal of Machine Learning Research, (8):65–102.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. Journal of the American statistical association, 58(301):236–244.

Wu, X. and Kumar, V. (2009). The top ten algorithms in data mining. CRC Press.