Fixed-size k-means

Mikko I. Malinen*, Juhani Järviö and Pasi Fränti
University of Eastern Finland, Joensuu, Finland
*E-mail: mmali@cs.uef.fi

Abstract

Balanced k-means aims at finding clusters of equal size. A slightly different variant requires predefined cluster sizes, which do not need to be equal. We present a generalized version of the balanced k-means algorithm to solve the fixed-size clustering problem with any predefined cluster sizes. The algorithm operates similarly to standard k-means but uses the Hungarian algorithm in the assignment step. We also discuss how the balanced k-means algorithm could be used as such by modifying the data.

1 Introduction

The goal of clustering is to group *n* data points into *k* clusters by minimizing intra-cluster distances and maximizing between-cluster distances. Each group is represented by a center point (centroid). Minimizing SSE in Euclidean space has been shown to be an NP-hard problem [1].
K-means [2] is the most common clustering algorithm. For a given number of clusters and some initial solutions, it finds a local minimum of *sum-of-squares errors* (SSE):

$$\text{SSE} = \sum_{j=1}^{k} \sum_{X_i \in C_j} \left\| X_i - C_j \right\|^2 \qquad (1)$$

where X_i denotes a data point and C_j denotes a centroid. It was shown in [3] that SSE not only minimizes intra-cluster distances but also maximizes between-cluster distances as a side effect.
The K-means algorithm consists of two repeatedly executed steps: the assignment step and the centroid step. They work as follows.
Assignment step: Assign the data points to clusters specified by the nearest centroid:

$$P_j^{(t)} = \left\{ X_i : \left\| X_i - C_j^{(t)} \right\| \leq \left\| X_i - C_{j*}^{(t)} \right\| \quad \forall j^* = 1, \ldots, k \right\} \qquad (2)$$

Update step: Calculate the mean of each cluster:

$$C_j^{(t+1)} = \frac{1}{\left| P_j^{(t)} \right|} \sum_{X_i \in P_j^{(t)}} X_i \qquad (3)$$

The two steps are repeated until the centroid locations no longer change. The assignment step and the update step are optimal with respect to SSE: The partition step minimizes SSE for a given set of centroids, and the update step minimizes SSE for a given partition. The solution, therefore, converges to a local optimum but without a guarantee of global optimality.
To obtain better results than those achieved by k-means, slower agglomerative algorithms [4, 5, 6] or more complex variants of k-means [7, 8, 9, 10] are often used.

Balanced clustering is the same as normal clustering, but it requires that the cluster sizes must be equal (or differ by at most 1). Balanced clustering is necessary for balancing the workload and avoiding unbalanced energy consumption in the network. *Balanced k-means* [11] works the same way as standard k-means, but the assignment step is different due to the balance constraint. It uses the Hungarian algorithm in the assignment step.
In general, balanced clustering is a 2-objective optimization problem with two goals that contradict each other: minimize SSE and balance the cluster sizes. Traditional clustering aims to minimize SSE without considering the balance. Balancing, on the other hand, would be trivial if we did not care about SSE; simply divide the points into equal-sized clusters randomly.
Fixed-size clustering is a generalized version of balanced clustering. It requires having clusters of predefined sizes. For example, we need to allocate resources with a given (non-equal) demand for the resource. Balance clustering is a special case of fixed-size clustering where the predefined cluster sizes are equal.
In the literature, balanced clustering has been widely studied, but less attention is paid to the more general case of fixed-size clustering. In this paper, we revise the original balanced k-means algorithm [11] by changing the balanced constraint to any fixed-size constraint. Otherwise, the algorithm is the same. We will also discuss an alternative approach in which, instead of modifying the algorithm, we would modify the data to make it a balanced clustering problem.

2 Balanced clustering

Hard balance constrains all the clusters to be of size $n/k \pm 1$. It is a strict limit. *Soft balance* is more relaxed. It merely aims towards more balanced clustering but does not enforce it. We next give a brief review of existing algorithms for these two cases.

2.1 Hard balance

One possible approach to enforce balance is to redesign the assignment step of k-means by adding a balance constraint.

For instance, the *constrained k-means* [12] defines the assignment as a minimum cost flow problem with minimum cluster sizes. The problem is then solved by linear programming in $O(k^{3.5}n^{3.5})$ time. The *balanced k-means* [11] solves the assignment step by the *Hungarian algorithm*, reducing the time complexity to $O(n^3)$. However, both approaches are slow and poorly scalable.

The algorithm in [13] was claimed to have an average time complexity of $O(mn^{1.65})$ to $O(mn^{1.7})$, where m is the number of iterations. The algorithm in [14] directly converted the problem to linear programming using a heuristic function, but with inferior clustering accuracy compared to k-means-based approaches. *Regularized k-means* [15] introduced a balance regularization term to the objective function, extending the method to the case of soft balance.

Other approaches include the *Fuzzy c-means* [16], a *Memetic algorithm* [17], which combines a crossover and a responsive threshold search, alternating between two different local search procedures, and a *strategic oscillation* [18].

2.2 Soft balance

The most common approach for the soft-balanced case is to use a penalty term. The greater the difference in cluster sizes, the higher the penalty, and vice versa. In the case of perfect balance, the penalty is zero. One of the first approaches is *frequency-sensitive competitive learning* [19], where clusters compete for data points, but the penalty increases the more data points they win. As a result, smaller clusters will be favored.

Least squares linear regression was used with a balance constraint that aims to minimize the variance of the cluster sizes [20]. The resulting hyperplanes represent the cluster boundaries, which are iteratively improved.

The concept of balance was generalized from cluster size also to their variance and density in [21]. The algorithm adds a multiplicative weight in the assignment step of k-means, followed by a separate balancing phase, in which points are shifted from clusters with the larger weights to clusters with lower weights.

An algorithm called *τ-balanced clustering* [22] takes as input a user-given parameter τ, which denotes the maximal difference between the cluster sizes. A value $\tau = 1$ implies hard balance, and values $\tau > 1$ soft balance. In the assignment step, points are assigned to the cluster with the nearest centroid that does not violate this constraint.

Ward's agglomerative clustering also aims at minimizing SSE. Other agglomerative clustering variants that minimize all pairwise within-cluster distances result in more balanced cluster sizes than using SSE [23].

3 Fixed-size k-means

Some balanced clustering algorithms can also be applied to the fixed-size clustering problem. For example, constrained k-means [12] allows setting any lower bounds on the cluster sizes. We next show how the original balance k-means [11] can be turned into a fixed-size k-means.

First, we need to define the assignment problem. A formal definition for the (linear) assignment problem is as follows. Given two sets (A and S) of equal size and with a weight function W: A×S → R, the goal is to find a bijection f: A → S so that the cost function is minimized:

$$\text{Cost} = \sum_{a \in A} W\big(a, f(a)\big). \qquad (4)$$

In the proposed algorithm, sets A and S represent the cluster slots and the data points, respectively, as shown in **Figure 1**. Fixed-sized k-means proceeds in the same way as k-means, but the assignment phase is different. Instead of selecting the nearest centroid, we have n pre-allocated slots divided equally among the clusters. The datapoints can be assigned only to these predefined slots.

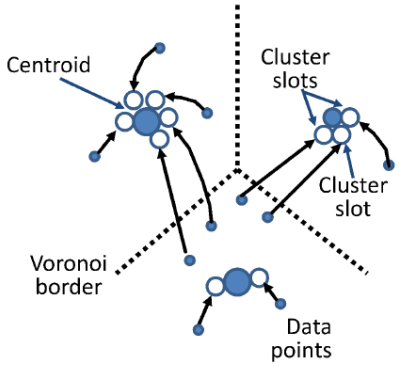

Figure 1 Assigning points to centroids via cluster slots

To find the optimal assignment minimizing SSE, we solve an assignment problem using the Hungarian algorithm [24]. We first construct a bipartite graph consisting of n data points and n cluster slots, as shown in **Figure 2**. We then partition the cluster slots in the clusters according to the assignments.

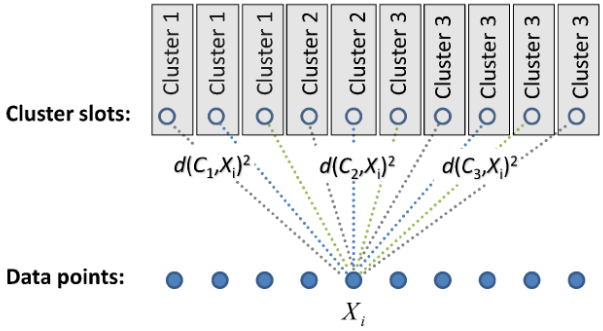

Figure 2 Minimum SSE calculation with fixed-sized clusters is done via a bipartite graph

Clusters slots that are allocated to the same cluster share the same centroid. The initial centroids can be drawn randomly from all data points. The edge weight is the squared distance from the point to the cluster centroid it is assigned to. Contrary to the standard assignment problem with fixed weights, here the weights dynamically change after each k-means iteration according to the newly calculated centroids. After this, we perform the Hungarian algorithm to get the minimal weight pairing. The squared distances are stored

in an $n \times n$ matrix, for the sake of the Hungarian algorithm. The update step is similar to that of k-means, where the new centroids are calculated as the means of the data points assigned to each cluster:

$$C_j^{(t+1)} = \frac{1}{n_i} \sum_{X_i \in C_j^{(t)}} X_i \qquad (5)$$

The weights of the edges are updated immediately after the update step. The pseudocode of the algorithm is in Algorithm 1. In the calculation of the edge weights, the cumulative sum of cluster sizes is:

$$cum(j) = \sum_{l=[1..j]} n_l \quad \forall j \in [1..k], \qquad (6)$$

where n_l is the cluster size, and the number of cluster slot is denoted by a. To determine which cluster the given cluster slot belongs to, we use the following equation:

$$\arg\min_j \; cum(j) \geq a. \qquad (7)$$

The edge weights are calculated as:

$$W(a,i) = dist(X_i, C^t_{\arg min_j \; cum(j) \geq a})^2 \quad \forall a \in [1..n] \quad \forall i \in [1..n]. \qquad (8)$$

After convergence of the algorithm, the partition of points X_i, $i \in [1..n]$, is:

$$X_{f(a)} \in P_{\arg\min_j cum(j) \geq a}. \qquad (9)$$

Algorithm 1 Fixed-sised clusters k-Means
Input: dataset X, cluster sizes n_l, number of clusters k
Output: partitioning of dataset.

Initialize centroid locations C^0.
$t \leftarrow 0$
repeat
  Assignment step:
    Calculate edge weights. Eq (8)
    Solve an Assignment problem.
  Update step:
    Calculate new centroid locations C^{t+1}. Eq (3)
  $t \leftarrow t+1$
until centroid locations do not change.
Output partitioning.

The convergence result in [12] (Proposition 2.3) applies also to the proposed fixed-size k-means as well. The result states that the algorithm terminates in a finite number of iterations at a locally optimal partition. At each iteration, the cluster assignment step cannot increase the objective function of the constrained k-means, as shown in Equation (3) in [12].

The cluster update step will either strictly decrease the value of the objective function, or the algorithm will terminate. There are a finite number of ways to assign n points to k clusters so that cluster h has at least π_h points. Since constrained k-means does not permit repeated assignments, and since the objective function is strictly nonincreasing and bounded below by zero, the algorithm must terminate at some locally optimal cluster assignment.

The same convergence result applies to the fixed-sized k-means as well. The assignment step is optimal with respect to SSE because the pairing and update steps are both optimal with regard to SSE, which is minimized clusterwise in the same way as in k-means.

4 Time complexity

The time complexity of the assignment step in k-means is $O(kn)$. The assignment step of the proposed fixed-sized k-means algorithm can be solved in $O(n^3)$ time with the Hungarian algorithm. This is also the bottleneck of the algorithm. The total time complexity is therefore $O(zn^3)$, where z is the number of iterations.

The greedy assignment heuristic could be considered as an alternative algorithm for faster speed. However, it is suboptimal and therefore does not guarantee convergence of the algorithm.

5 Alternative approach

We next discuss an alternative approach for solving the fixed-size clustering problem. Instead of modifying the algorithm, we can modify the dataset as follows, allowing the original balanced k-means algorithm to be used. The idea goes as follows.

Suppose that we have three clusters with the size constraints of 5, 3, and 2. Instead of having as many cluster slots as the sizes require, we create additional dummy slots so that their total number would become equal: 5, 5, 5. We then add 0+2+3 = 5 dummy data points so that their number matches that of the cluster slots (see **Figure 3**).

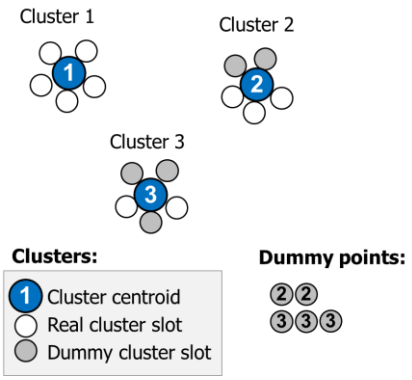

Figure 3 Adding dummy points and dummy cluster slots. The distance of the dummy data point to its pre-assigned cluster is zero, and to all other clusters, it is ∞

The dummy points are then numbered and forced to their pre-assigned clusters by setting their distances as follows. The distance of a dummy point to its own pre-assigned cluster is 0, and ∞ to all other clusters. In this way, the dummy points will always occupy equally many slots in a cluster as there are dummy slots. In this way, they do not affect the clustering, and any balanced clustering algorithm can be used.

We note that, when using a distance matrix, this scheme works easily. However, if we calculated the Euclidean distances on demand, the attribute values for the dummy points would require a bit of thinking work. Since we can easily modify the balanced k-means algorithm, we do not need this approach. However, if another algorithm were used as the starting point, this modification might become useful.

The drawback is the additional computation caused by the dummy points. In **Figure 1**, the addition is 50%. It would multiply the processing time by a factor of about 3.4. In the extreme case, there is one big cluster of size (n-k+1) and (k-1) small clusters of size 1. This would increase the time complexity from O(zn^3) to O(zn^6), which is why we do not consider this variant further.

6 Experiments

As an application, we present *seating planning* for a party. First, we need a distance matrix, where the compatibility of people is defined by distance. This must be done manually. The distance matrix can be created as follows.

The organizer sets 0 values between participants who are known to be good matches to sit at the same table. They are typically close friends or relatives. Then, for people who are not expected to carry on a conversation with each other, the distance is set to a maximum of, say, 100. These can be distances between the other PhD students and lab members, and the relatives of the doctor who cannot carry on a discussion on the thesis topic. Language barrier is another reason to set a high distance value.

The creation of the distance matrix resembles semi-supervised clustering where the user makes a priori known must-link and cannot-link constraints [25]. The *must-link* constraint between a pair of points requires that the two points be assigned to the same cluster. A *cannot-link* constraint, on the other hand, forces the two points to be assigned to different clusters.

The distance matrix is then converted to Euclidean space by multidimensional scaling [26]. The result is the data in a higher-dimensional space, but with distances preserved. We then perform fixed-sized k-means, taking data X, the number of tables k, and their sizes n_l as input. The output is the seating plan.

We tested the algorithm by creating a seating plan for Mikko I. Malinen's doctoral dissertation evening party (karonkka) in 2015. There were 22 people invited. In the compatibility distance matrix, there are 22·22 = 484 distances. The sizes of the party tables were [4, 4, 5, 6, 3], and we set k = 5 accordingly. Data became 10-dimensional.

We repeated the algorithm 1000 times, which took only a few seconds. The result is shown in **Figure 4**. The average within-cluster distances are 0.0, 5.5, 4.3, 4.8, 0.0. During the event, we also recognized that people were happy with the seating plan, so the result was kind of subjectively verified as well. The software and data are available here:

- Software: http://cs.uef.fi/ml/software/
- Data: http://cs.uef.fi/sipu/datasets/FKM/.

Our second example is Finland's parliament members. We position the parties in the traditional left-right axis as shown in **Figure 5**. The distance between each party is 10, except for the leftmost and rightmost parties, which are 20 units apart. The design is ad hoc but roughly follows their generally understood position.

There are 200 parliament members divided into nine parties. We use the original 2023 election results. We created k = 5 tables with sizes [55, 46, 36, 35, 28] by scaling up the table sizes from the dissertation example to match n = 200. We then run the fixed k-means algorithm 100 times. It was shown in [27] that the result of k-means can be significantly improved by repeating the algorithm multiple times. This turned out to be the case here as well.

Figure 6 shows the table allocations of a single run of the algorithm and the best result out of 100 runs. Both are sensible, but the best achieved a much lower average distance (MSE = 35.8) compared to the average result (MSE = 77.5). The most visible difference is that the PS seating has its own table. Left wing party (Vas) is smaller and must share the table with SDP in both solutions.

	JR	ER	IR	NR	Mrez	TK	VH	SW	PM	TM	SR	Mrus	OV	HC	RMI	AT	Outi	Olavi	ALM	RH	PF	MM
JR	0	0	0	0	100	100	100	100	7	10	10	10	10	100	100	100	100	100	100	100	100	10
ER	0	0	0	0	100	100	100	100	7	10	10	10	10	100	100	100	100	100	100	100	100	10
IR	0	0	0	0	100	100	100	100	7	10	10	10	10	100	100	100	100	100	100	100	100	10
NR	0	0	0	0	100	100	100	100	7	10	10	10	10	100	100	100	100	100	100	100	100	10
MR	100	100	100	100	0	7	7	5	7	7	7	7	7	5	5	7	7	7	7	100	100	100
TK	100	100	100	100	7	0	0	7	7	7	7	7	2	7	7	7	7	7	7	50	50	50
VH	100	100	100	100	7	0	0	7	7	7	7	7	2	7	7	7	7	7	7	50	50	50
SW	100	100	100	100	5	7	7	0	7	7	7	7	5	7	7	7	7	7	7	100	50	100
PM	7	7	7	7	7	7	7	7	0	5	5	5	7	7	7	7	0	0	0	100	100	100
TM	10	10	10	10	7	7	7	7	5	0	0	0	7	7	7	7	0	0	5	100	100	100
SR	10	10	10	10	7	7	7	7	5	0	0	0	7	7	7	7	0	0	3	100	100	100
MR	10	10	10	10	7	7	7	7	5	0	0	0	7	7	7	7	0	0	3	100	100	100
OV	10	10	10	10	7	2	2	5	7	7	7	7	0	7	7	7	7	7	7	50	50	50
HC	100	100	100	100	5	7	7	7	7	7	7	7	7	0	3	3	7	7	7	100	100	100
RMI	100	100	100	100	5	7	7	7	7	7	7	7	7	3	0	3	7	7	7	100	100	100
AT	100	100	100	100	7	7	7	7	7	7	7	7	7	3	3	0	7	7	7	100	100	100
OM	100	100	100	100	7	7	7	7	0	0	0	0	7	7	7	7	0	0	0	100	100	100
OM	100	100	100	100	7	7	7	7	0	0	0	0	7	7	7	7	0	0	0	100	100	100
ALM	100	100	100	100	7	7	7	7	0	5	3	3	7	7	7	7	0	0	0	100	100	100
RH	100	100	100	100	100	50	50	100	100	100	100	100	50	100	100	100	100	100	100	0	0	0
PF	100	100	100	100	100	50	50	50	100	100	100	100	50	100	100	100	100	100	100	0	0	0
MM	10	10	10	10	100	50	50	100	100	100	100	100	50	100	100	100	100	100	100	0	0	0

Figure 4 Seating plan data for the doctoral (karonkka) party of Mikko I. Malinen. There are 22 participants and five tables of sizes 4, 4, 5, 6, 3. Green color demonstrates a good match (distance ≤5), and red a poor match (distance >50)

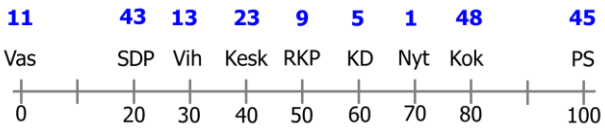

Figure 5 Finnish parliament parties, where their distance along the traditional left-right axis defines their compliance according to our ad hoc positioning

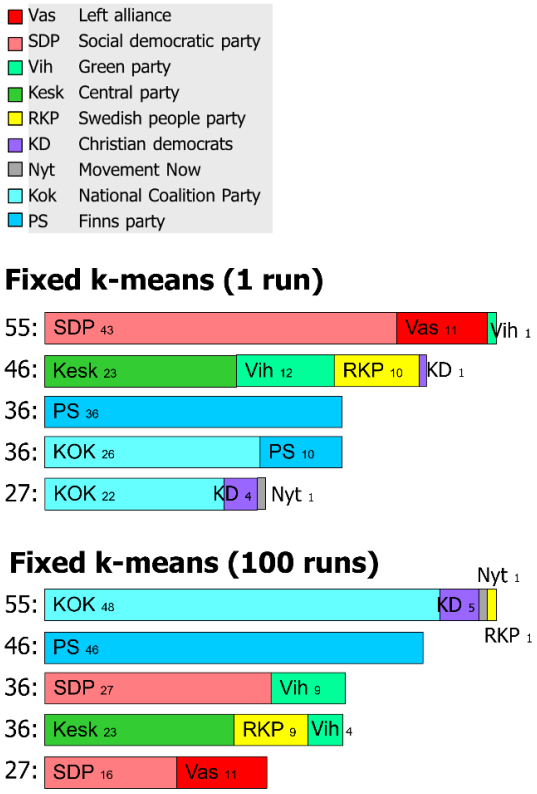

Figure 6 Seating of the parliament members for a single run of fixed k-means and the best of 100 repeats (below)

7 Conclusion

We have presented a fixed-size k-means algorithm for any given cluster sizes as constraints. The algorithm uses optimal assignment by the Hungarian algorithm, which takes $O(n^3)$ time and is practical only up to about 5000 datapoints. For larger datasets, a faster algorithm should be used.

One possibility would be to model the assignment as an optimal transportation problem and solve it using a faster $O(n^2k)$ time modified Hungarian algorithm [28]. Even this solution would still be rather slow.

Another possibility would be to modify the stochastic variant of the balanced k-means (BKM+) algorithm in [29] to handle fixed-size clusters. It was shown to provide better global optimization even in the case of normal clustering, and therefore, is expected to lead to improved results in both quality and processing time. This is left as future work.

Fixed-size k-means is also sensitive to initialization in the same way as standard k-means. This can be addressed by better initialization or by repeats [27] as was done in this paper. A better local optimizer, such as random swap [8], could also be used.

8 Literature

[1] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, NP-hardness of Euclidean sum-of-squares clustering, *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009. DOI: 10.1007/s10994-009-5103-0

[2] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in Proc. 5th *Berkeley Symp. Math. Statist. Probability*, vol. 1, pp. 281–297, University of California Press, 1967.

[3] P. Fränti, C. Cariou, and Q. Zhao, Cluster overlap as objective function, *Computers, Materials & Continua*, Article no. 66534, 2025.

[4] W. H. Equitz, A new vector quantization clustering algorithm, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, no. 10, pp. 1568-1575, Oct. 1989, doi: 10.1109/29.35395.

[5] P. Fränti, T. Kaukoranta, D.-F. Shen, and K.-S. Chang, Fast and memory efficient implementation of the exact PNN, *IEEE Trans. Image Processing*, vol. 9, no. 5, pp. 773–777, May 2000. DOI: 10.1109/83.841516

[6] P. Fränti and O. Virmajoki, Iterative shrinking method for clustering problems, *Pattern Recognition*, vol. 39, no. 5, pp. 761–775, May 2006. DOI: 10.1016/j.patcog.2005.09.012

[7] D. Arthur and S. Vassilvitskii, k-means++: The advantages of careful seeding, in Proc. 18th *ACM-SIAM Symp. Discrete Algorithms* (SODA), pp. 1027–1035, 2007.

[8] P. Fränti, Efficiency of random swap clustering, *Journal of Big Data*, vol. 5, Article 13, 2018. DOI: 10.1186/s40537-018-0122-y

[9] A. Likas, N. Vlassis, and J. J. Verbeek, The global k-means clustering algorithm, *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003. DOI: 10.1016/S0031-3203(02)00060-2

[10] D. Pelleg and A. Moore, X-means: Extending k-means with efficient estimation of the number of clusters, in Proc. 17th *Int. Conf. Machine Learning* (ICML), pp. 727–734, 2000.

[11] M. I. Malinen and P. Fränti, Balanced k-means for clustering, in *Joint Int. Workshop on Structural, Syntactic, and Statistical Pattern Recognition (S+SSPR)*, LNCS 8621, pp. 32–41, Joensuu, Finland, 2014. DOI: 10.1007/978-3-662-44415-3_4

[12] P. S. Bradley, K. P. Bennett, and A. Demiriz, Constrained k-means clustering, Technical Report MSR-TR-2000-65, Microsoft Research, Redmond, 2000.

[13] W. Tang, Y. Yang, L. Zeng, and Y. Zhan, Optimizing MSE for clustering with balanced size constraints, *Symmetry*, vol. 11, no. 3, p. 338, 2019. DOI: 10.3390/sym11030338

[14] Z. Zhu, D. Wang, and T. Li, Data clustering with size constraints, *Knowledge-Based Systems*, vol. 23, no. 8, pp. 883–889, 2010.

[15] W. Lin, Z. He, and M. Xiao, Balanced clustering: A uniform model and fast algorithm, in *Proc. 28th Int.*

Joint Conf. Artif. Intell. (IJCAI), pp. 2987–2993, 2019. DOI: 10.24963/ijcai.2019/414

[16] D. Chakraborty and S. Das, Modified fuzzy c-mean for custom-sized clusters, *Sādhanā*, vol. 44, article 182, 2019. DOI: 10.1007/s12046-019-1166-1

[17] Q. Zhou, J. K. Hao, and Q. Wu, Responsive threshold search based memetic algorithm for balanced minimum sum-of-squares clustering, *Information Sciences*, vol. 569, pp. 184–204, 2021. DOI: 10.1016/j.ins.2021.04.014

[18] R. Martín-Santamaría, J. Sánchez-Oro, S. Pérez-Peló, and A. Duarte, Strategic oscillation for the balanced minimum sum-of-squares clustering problem, *Information Sciences*, vol. 585, pp. 529–542, 2022. DOI: 10.1016/j.ins.2021.12.059

[19] A. Banerjee and J. Ghosh, Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres, *IEEE Trans. Neural Networks*, vol. 15, no. 3, pp. 702–718, May 2004. DOI: 10.1109/TNN.2004.824416

[20] J. Han, H. Liu, and F. Nie, A local and global discriminative framework and optimization for balanced clustering, *IEEE Trans. Neural Networks. Learning Systems*, vol. 30, no. 10, pp. 3059–3071, 2019. DOI: 10.1109/TNNLS.2018.2870131

[21] S. Gupta, A. Jain, and P. Jeswani, Generalized method to produce balanced structures through k-means objective function, *Int. Conf. IoT in Social, Mobile, Analytics and Cloud* (I-SMAC), pp. 586–590, 2018.

[22] Y. Lin, H. Tang, Y. Li, C. Fang, Z. Xu, Y. Zhou, and A. Zhou, Generating clusters of similar sizes by constrained balanced clustering, *Applied Intelligence*, vol. 52, pp. 5273–5289, 2022. DOI: 10.1007/s10489-021-02682-y

[23] M. I. Malinen and P. Fränti, All-pairwise squared distances lead to more balanced clustering, *Applied Computing and Intelligence*, vol. 3, no. 1, pp. 93–115, May 2023. DOI: 10.3934/aci.2023006

[24] R. Burkhard, M. Dell'Amico, and S. Martello, Assignment Problems, Revised reprint, SIAM, 2012. DOI: 10.1137/1.9781611972238

[25] K. Wagstaff and C. Cardie, Clustering with instance-level constraints, in Proc. 17th *Int. Conf. Machine Learning* (ICML), pp. 1103–1110, 2000.

[26] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, Chapman & Hall, London, 1994. DOI: 10.1201/9780367801700

[27] P. Fränti and S. Sieranoja, How much k-means can be improved by using better initialization and repeats?, *Pattern Recognition*, vol. 93, pp. 95–112, 2019. DOI: 10.1016/j.patcog.2019.04.014

[28] Y. Xie, Y. Luo, X. Huo, Solving a special type of optimal transport problem by a modified Hungarian algorithm, *Transactions on Machine Learning Research*, 1-27, 02, 2023.

[29] R. de Maeyer, S. Sieranoja, and P. Fränti, Balanced k-means revisited, *Applied Computing and Intelligence*, vol. 3, no. 2, pp. 145–179, 2023.