Net-Constrained Clustering Problem

Mindaugas Kepalas[1], **Julius Žilinskas**[1]

[1]Institute of Data Science and Digital Technologies, Vilnius University

Institute of Data Science and Digital Technologies

The problem

Probably everyone who has ever taken a course or got in touch with data science has heard about the famous **k-means algorithm**. It is a beautiful, fast and intuitive algorithm, which is used for finding structure in the data, e.g., for determining (initially unknown) groups of similar objects.

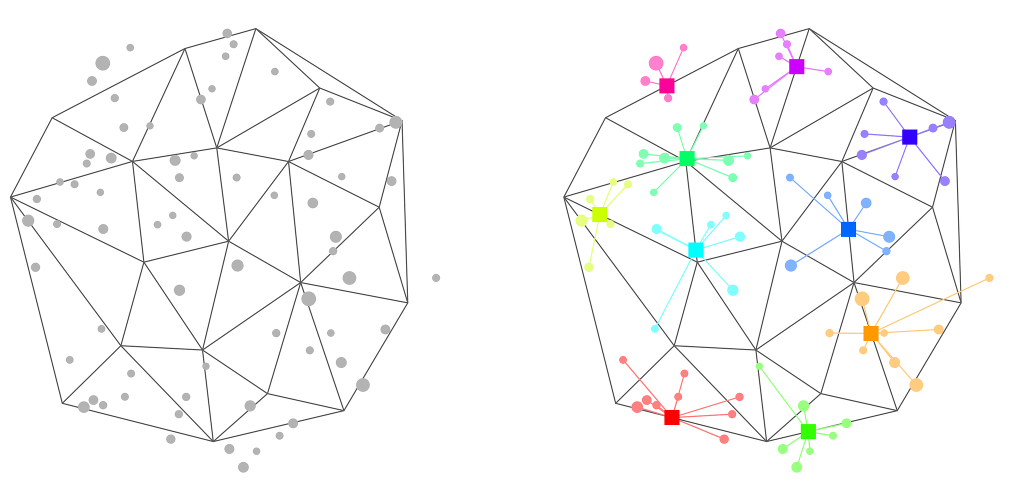
More precisely, **k-means** algoritm **seeks to solve the** so called **minimum-sum-of-squares clustering (MSSC) problem**: given a set of points $P_1, P_2, \dots, P_N$ with corresponding weights $w_1, w_2, \dots, w_N$ and given the number of groups K, we want to partition all the elements into K clusters $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$ with centers $Q_1, Q_2, \dots, Q_K$, so that the loss of the solution is minimal. Mathematically, the goal is to solve the following optimization problem:

$$\min_{(\mathcal{C}_1,Q_1),\dots,(\mathcal{C}_K,Q_K)} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} w_i \left\|P_i - Q_k\right\|_2^2 \qquad (1)$$

In our research, **we study MSSC problems for which the locations of the centers are constrained** to a subset of the space; in the illustrations, this set is defined by a union of a set of segments in the plane; we call it a "net" and label with letter $\mathcal{N}$. Our problem is thus mathematically stated as follows:

$$\min_{(\mathcal{C}_1,Q_1),\dots,(\mathcal{C}_K,Q_K)} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} w_i \left\|P_i - Q_k\right\|_2^2 \quad \text{s.t.} \quad Q_k \in \mathcal{N} \qquad (2)$$

Problem instance and its solution are illustrated in **Figure 1**.



(a) Some points with weights and the net-constraint

(b) A possible solution of the problem

Figure 1: Problem illustration. Note that cluster-centers satisfy the property $Q_k \in \mathcal{N}$

Net-constrained k-means algorithm

We will solve the problem shown in **Figure 1(a)** with a **net-constrained k-means** algorithm. Any **k-means**-type algorithm starts with an

- **Initialization Step:** sample K random centers $Q_k \in \mathcal{N}$, which will be used to define the initial clusters [**Figure 2(a)**]

One can imagine that initialized centers define **Voronoi cells** as in **Figure 2(b)** (those cells are only shown here for illustration purposes and do not have to be computed by the algorithm).

Next, iteratively apply the following two steps:

- **Assignment Step**: given (**fixed!**) cluster-centers Q_k, **determine** (update) clusters $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$ by assigning each point P_i to the closest center (cluster) [**Figure 2(c)**]
- **Location Step**: for **fixed** clusters $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$, **optimize** (update) cluster centers $Q_1, Q_2, \dots, Q_K$ [**Figure 2(d)**]

Figure 2(e) illustrates that after the **Location Step**, the closest center for a point might change (e.g., a few red points from the figure are now in the green cell). Therefore, in the re-**Assignment Step** [**Figure 2(f)**] those points "move" to another cluster. Now again follows **Location Step** [**Figure 2(g)**] and etc.

One can convince himself/herself, that after either **Assignment Step** or **Location Step**, the loss defined in (1) decreases or stays the same (in the later case the algorithm is terminated).

In **Assignment Step** (**2(e)** $\Rightarrow$ **2(f)**), the distance can only decrease for each point (because each P_i can only "move" to a closer center - otherwise stays at the old cluster if it cannot improve!), what results that the total loss cannot increase.

Now lets analyze **Location Step** (see **2(d)**, **2(g)**). Lets define the loss for cluster $\mathcal{C}_k$ given an arbitrary center $Q \in \mathbb{R}^2$:

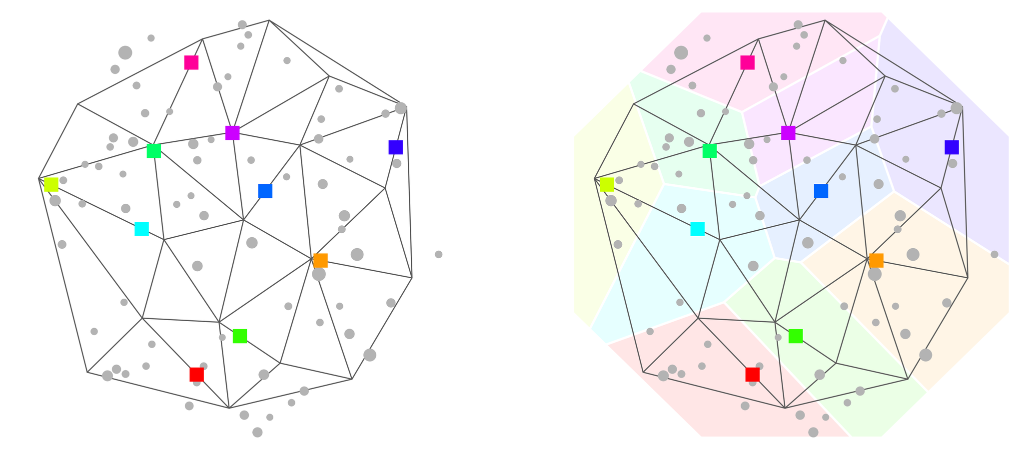
$$\mathcal{L}_k(Q) := \sum_{i \in \mathcal{C}_k} w_i \left\|P_i - Q\right\|_2^2$$

Initially, we have centers Q_k^{init} (gray squares) with loss $\mathcal{L}_k\left(Q_k^{\text{init}}\right)$. We now define Q_k^{opt} (coloured squares):

$$Q_k^{\text{opt}} := \arg\min_Q \left[\mathcal{L}_k(Q) \quad \text{s.t.} \quad Q \in \mathcal{N}\right]$$

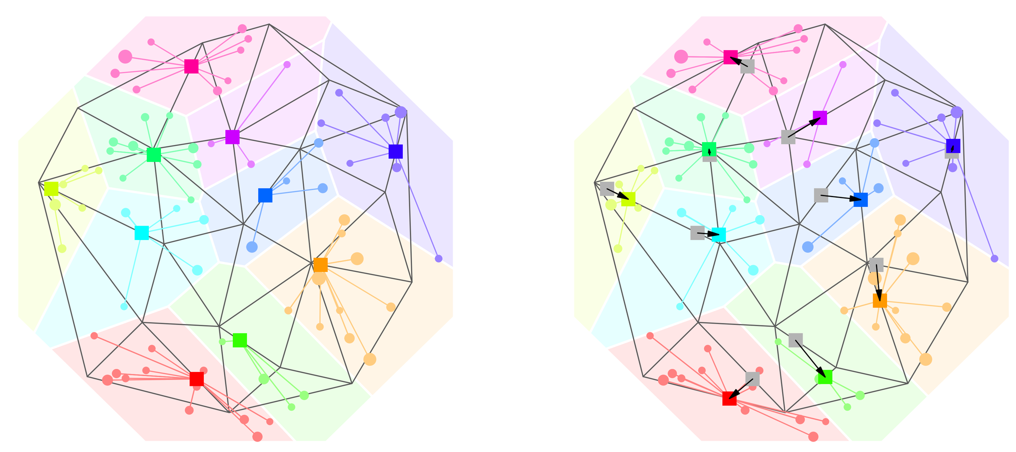
Since initially $Q_k^{\text{init}} \in \mathcal{N}$, we have $\mathcal{L}_k\left(Q_k^{\text{opt}}\right) \le \mathcal{L}_k\left(Q_k^{\text{init}}\right)$, and $\sum_{k=1}^{K} \mathcal{L}_k\left(Q_k^{\text{opt}}\right) \le \sum_{k=1}^{K} \mathcal{L}_k\left(Q_k^{\text{init}}\right)$ follows.

Because the loss (1) decreases with every step and is bounded from below by 0, the algorithm converges.



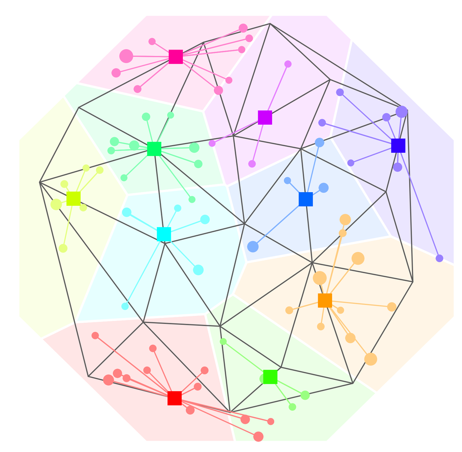
(a) Initialize: sample random centers on the net

(b) Voronoi cells generated by random centers

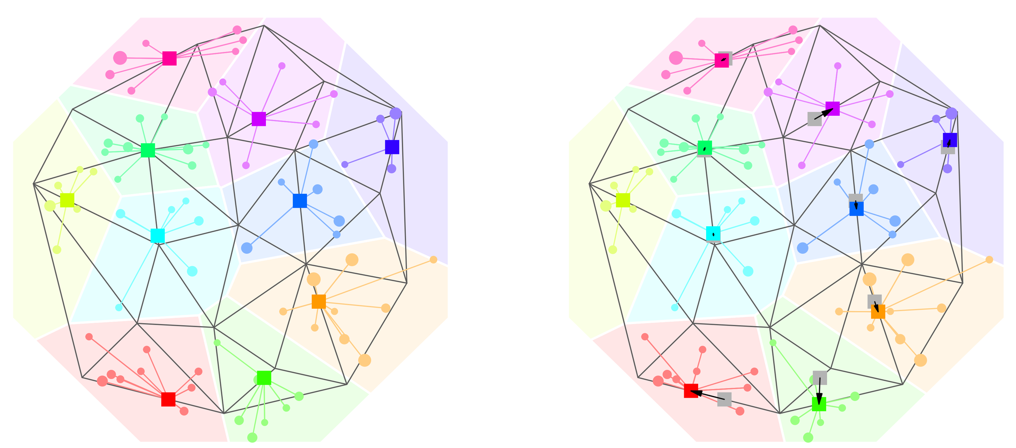


(c) Assignment Step: assign points to the centers

(d) Location Step: clusters determined; optimize centers



(e) Optimized centers generate new Voronoi cells, some points "move" to other clusters



(f) Assignment Step: update point assignment

(g) Location Step: clusters updated; optimize centers

Figure 2: Illustration of **net-constrained k-means algorithm** in $\mathbb{R}^2$

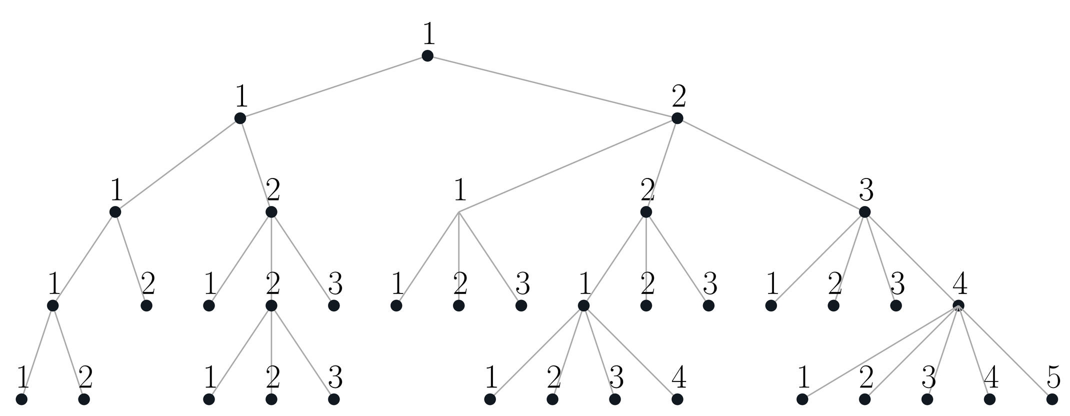
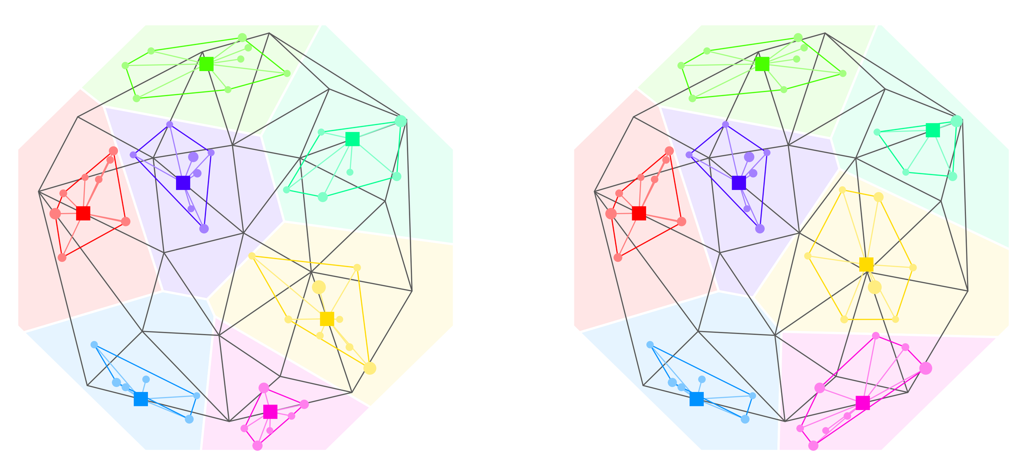
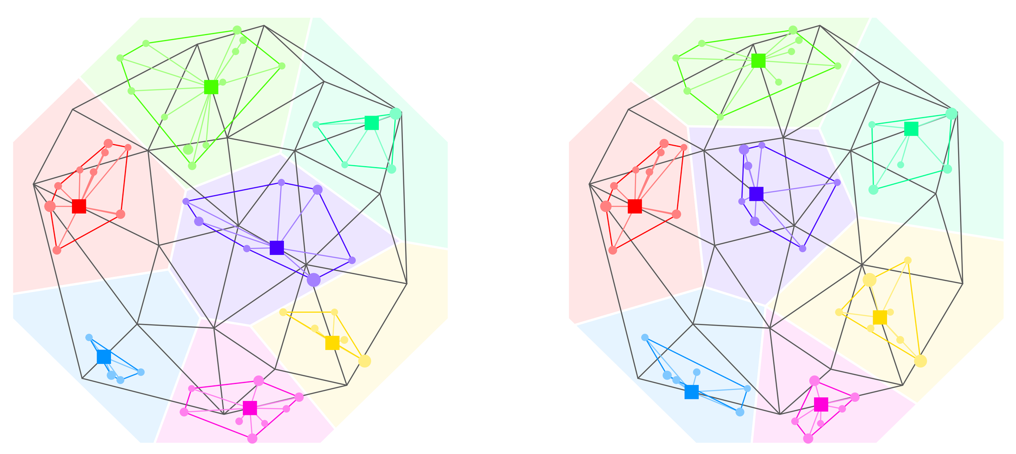


Figure 3: Branch-and-bound tree. Full tree for the first 4 points and some parts of it for the 5th point



(a) Locally-optimal solution. **Loss:** 2.055132

(b) Locally-optimal solution. **Loss:** 2.271582



(c) Locally-optimal solution. **Loss:** 2.459568

(d) Globally-optimal solution. **Loss:** 2.034595

Figure 4: Illustration that **k-means** algorithm does not guarantee a globallly-optimal solution. Note also that in all of the cases **convex-hulls of the clusters are within Voronoi-cells of the centers**

Global solution

We have presented a method to find a solution to problem (2): the **net-constrained-k-means algorithm**. Given initial (**random**) cluster-centers, this algorithm finds a solution which cannot be improved by applying **Assignment Step** or **Location Step**: a **locally-optimal** solution. However, this solution is not guaranteed to be **globally-optimal**, as is illustrated in **Figure 4**: here one can see four different local solutions.

Nevertheless, in **Figure 3(d)** we claim that the presented solution is globally-optimal: this is the best solution possible for the given problem. How do we know that?

One way to proove this would be to formulate problem (2) as a mixed-interger-quadratically-constrained-programming (**MIQCP**) problem and then use the available academic-free or commercial solvers suitable for solving such problems.

However, our attempts to solve the problem with **gurobi** were rather disappointing. By running the solver on different problem instances, we have discovered that **gurobi** can only solve very small problems.

Thus, we decided to try to solve the problem using branch-and-bound paradigm. The developed algorithm was used to prove that solution in **Figure 3(d)** is globally-optimal.

We note that from **gurobi** experiment time data we estimated that its MIQCP solver would take more than 1000 years to report (proove) the globally-optimal solution for the problem instance in **Figure 4**.

Branch-and-bound algorithm

Suppose that in the begining, all points are "unassigned" and all clusters are "closed". A cluster is "opened" if it is assigned a point, and lets agree that the clusters must be opened by increasing index: firstly we must open $\mathcal{C}_1$, then $\mathcal{C}_2$, $\mathcal{C}_3$ and etc.

Lets consider the first point P_1. From the rules we have agreed upon, we have that P_1 must be placed in $\mathcal{C}_1$.

For the second point P_2 there are two possibilities:

- **Assignment to an already opened cluster** $\mathcal{C}_1$
- **Openning a new cluster**: we open cluster $\mathcal{C}_2$ by assigning point P_2 to it

For other points $P_3, \dots, P_N$, we proceed in the same fashion: we assign a point into one of the opened clusters, or, if possible (e.g., the last cluster is still not opened), assign a point to a new cluster.

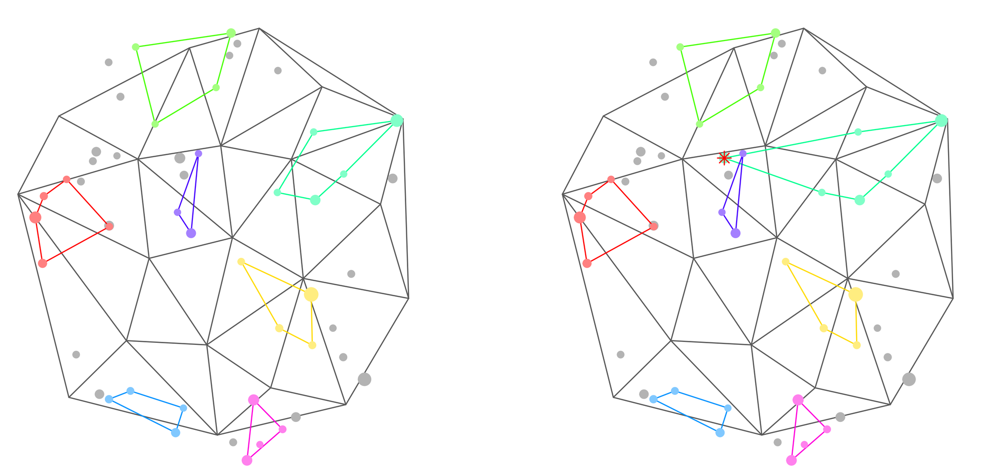
This procedure is illustrated in **Figure 3**: the root corresponds to P_1, the nodes at the second level correspond to P_2 and etc. The (full) tree enumerates all possible partitions of N elements into K non-empty subsets and contains $\left\{ N \atop K \right\}$ (Striling number of the second kind) "leaves" - which is a huge number. We use two main ideas for "cutting" the branches of the tree:

- **No-improvement cut**: cut the branch if its current loss is already larger than the loss of the best known solution
- **Convex-hull overlap cut**: cut the branch if any of the convex-hulls of the clusters overlap [see **Figure 5**]

While **No-improvement cut** is simple and intuitive, one can also proove that **Convex-hull overlap cut** is valid, too. In any local solution ($\Rightarrow$ in any global as well), each point is assigned to the closest center, thus: $i \in \mathcal{C}_k \Rightarrow P_i \in \text{VoronoiCell}\left(Q_k\right)$ (for any point P_i, any cluster $\mathcal{C}_k$). By the convexity property of Voronoi-cells, for any set of points within the cell we have:

$$\left\{P_i : i \in \mathcal{C}_k\right\} \subset \text{VoronoiCell}\left(Q_k\right) \quad \Rightarrow$$
$$\text{ConvexHull}\left(\left\{P_i : i \in \mathcal{C}_k\right\}\right) \subset \text{VoronoiCell}\left(Q_k\right) \quad \forall k \qquad (3)$$

Because Voronoi-cells of cluster centers do not overlap but only touch each other, neither can overlap the convex hulls of cluster members in any locally-optimal solution [see **Figure 4**].



(a) Valid branch

(b) Invalid branch

Figure 5: Illustration of **Convex-hull overlap cut**. After inserting the red-starred point into one of the clusters, convex-hulls start to intersect - any further point assignment can not lead to a globally-optimal solution and we can cut the corresponding branch