The DBSCAN Clustering Algorithm on P Systems

György Vaszil

Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
vaszil.gyorgy@inf.unideb.hu

Summary. We show how to implement the DBSCAN clustering algorithm (Density Based Spatial Clustering of Applications with Noise) on membrane systems using evolution rules with promoters and priorities.

1 Introduction

Clustering is the process of partitioning elements of a dataset according to some similarity measure in such a way that elements in the same cluster are similar, elements in different clusters are dissimilar. Clustering analysis is widely used in several areas of data mining as a tool to discover implicit patterns and deduce knowledge based on the data, or it can also be used for preprocessing before the application of other algorithms. The reader is referred to [3] for more details about clustering, and the area of data mining in general.

The density based clustering of applications with noise (DBSCAN) clustering algorithm was introduced in [2]. It clusters data points based on density (a point is dense if it has many neighbors within a given radius. The algorithm can be summarized in the following steps:

Input: A set of points, the neighborhood radius ϵ, and the density threshold $MinPts$.

1. Mark all points "univisited".
2. Pick an unvisited point p,
   - change its mark to "visited", and
   - count the number of points in its ϵ neighborhood to see if it is a core point, that is, if the number of points in its ϵ neighborhood is at least $MinPts$.
   - If p is not a core point, mark it as "noise", otherwise create C as a new cluster and add p to C, together with those points in its ϵ neighborhood which do not belong to any cluster yet.

3. Pick an unvisited point p' in C
   - change the mark of p' to "visited",
   - count the number of points in its ϵ neighborhood to see if it is a core point.
   - If p' is a core point, add those points to C in its ϵ neighborhood which do not belong to any cluster yet.
   - If there are unvisited points in C, go back to 3.
4. If there are unvisited points in the data set, go back to 2.

Output: The clustering result.

In the following we intend to give an implementation of this algorithm in terms of P systems. The system will use evolution rules with promoters and priorities. Our goal is to exploit the parallelism of P systems in order to parallelize, and thus, to speed up the DBSCAN algorithm which in its original version works with $O(n^2)$ time complexity on a sequential machine (where n is the number of points to be clustered). On P systems, the running time can be reduced to $O(n)$.

Ours is not the only proposal for clustering with P systems. As examples, see [4], or see [9] for a so called k-nearest base clustering algorithm on P systems with active membranes, and [8] for clustering with splicing P systems. Even a DBSCAN algorithm implementation was already presented in [11], we believe however, that our present proposal is conceptually simpler.

2 Preliminaries and Definitions

Let O be a finite nonempty set (the set of object) and $\mathbb{N}$ be the set of non-negative integers. A *multiset* M (or an *multiset* M for short), over O is a pair (O, f), where $f : O \to \mathbb{N}$ is mapping which gives the *multiplicity* of each object $a \in O$. The set $supp(M) = \{a \in O \mid f(a) > 0\}$ is called the *support* of M. If $supp(M) = \emptyset$, then M is the empty multiset. If $a \in supp(M)$, then $a \in M$, and $a \in^n M$ if $f(a) = n$. In the following we represent a multiset M over $O = \{a_1, \ldots, a_k\}$ by the string $a_1^{M(a_1)} \ldots a_k^{M(a_k)}$ (or any of its permutations).

Membrane systems, or P systems, were introduced in [5] as computing models inspired by the functioning of the living cell. The main component of such a system is a membrane structure with membranes enclosing regions as multisets of objects. Each membrane has an associated set of operators working on the objects contained by the region. These operators can be of different types, they can change the objects present in the regions or they can provide the possibility of transferring the objects from one region to another one. The evolution of the objects inside the membrane structure from an initial configuration to a final configuration corresponds to a computation having a result which is derived from some properties of the final configuration.

Several variants of the basic notion have been introduced and studied proving the power of the framework, see the monograph [6] for a comprehensive introduction, the handbook [7] for a summary of notions and results of the area, and the volumes [1, 10] for various applications.

An $n+3$-tuple $\Pi = (O, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ is a *P system* of degree n, where

- O is a finite set called the alphabet of objects Π;
- w_i, $1 \leq i \leq n$, is a finite multiset of objects containing the initial contents of compartment i of Π;
- R_i, $1 \leq i \leq n$, is a finite set of rules of the form $u \to v$ or $u \to v|_\alpha$ with $u, \alpha \in O^*$ and $v \in O \cup \{here, in, out\}$;
- $\rho \subset R \times R$ is a priority relation defined on the rules of $R = \bigcup_{1 \leq i \leq n} R_i$ which may also be empty.

For a P system $\Pi = (O, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ as above, an n-tuple $c = (u_1, \ldots, u_n)$ with $u_i \in O^*$ for each i, is called a *configuration* of Π and $c_0 = (w_1, \ldots, w_n)$ is called its *initial configuration.* The multisets $u_1, \ldots, u_n$ are also called the *contents* of compartments $1, \ldots, n$, in configuration c.

A P system changes its configurations by applying its rules in the so-called maximally parallel manner. A multiset of rules from R_i for some $1 \leq i \leq n$, as given above, is applicable to a configuration c, if and only if the union of the multisets on the lefthand sides of the rules is a submultiset of u_i, the contents of the ith region. As a result of applying the rules to c, each object of the multisets on the righthand sides of the rules replace the objects on the lefthand side. Moreover, if the objects are moved to the respective neighboring regions according to the target indicators $here, in, out$. Rules multisets are applied in all regions in parallel, producing a series of configuration changes.

We say that the configuration $c' = (v_1, \ldots, v_n)$ of Π is obtained directly from $c = (u_1, \ldots, u_n)$ by applying the rules in a maximally parallel manner, if the rule multisets applied in the regions are maximal, that is, by adding any rule to the multiset, they are not applicable simultaneously any more. A rule of the form $x \to y|\alpha$ is applicable only if $\alpha \in O^*$ (the promoter mutiset) is a submutiset of the respective region.

When the priority relation ρ is nonempty, we denote by $r_1 > r_2$ if $(r_1, r_2) \in \rho$, that is, if a rule r_1 has higher priority than r_2. In such a case, r_2 can only be applied to configurations where r_1 is not applicable.

A sequence of configurations $c_0, c_1, \ldots$ of Π, obtained directly from each other and starting from the initial configuration, is called a computation. The computation halts if no rule can be applied in the current configuration. The result of a halting computation are the multisets of objects in the compartments at halting.

3 Implementing the DBSCAN Algorithm

In order to perform the clustering algorithm, let us construct a P system $\Pi = (O, [\ ], w, R, \rho)$ with

$$O = \{p_i, p'_i, p_{i,j?}, p_{i,j}, p'_{i,j}, p_i^{cr}, p_i^{ns}, p_{i,j?}^{ns}, p_{i,j}^{cr}, p_{i,j}^{ncr} \mid 1 \leq i, j \leq n\} \cup$$
$$\{E_i, H_i \mid 1 \leq i \leq n\} \cup \{A, B, C, D, F, F', F''\}.$$

The objects of the form p_i represent the n points of the data set. We assume that we have a distance function $d : \{p_1, \ldots, p_n\}^2 \to \mathbb{N}$.

The initial contents of the system is the multiset

$$w = Ap_1 \ldots p_n,$$

corresponding to the n points, and a synchronizing symbol A.

Now we present the rules of R and at the same time, describe the functioning of the system. Let $R = R_{pick} \cup R_{cehck} \cup R_{mark} \cup R_{check2} \cup R_{mark2}$, and let us describe these rule sets as follows.

$$R_{pick} = \{Ap_i \to Bp'_i \mid 1 \leq i \leq n\}.$$

Using the single occurrence of A, the application of one of these rules picks a point p_i for some $1 \leq i \leq n$ by changing it to its primed version p'_i. Now, in the presence of p'_i, the system checks whether the ith point is dense, that is, whether it has more than $MinPts$ points in its ϵ neighborhood. This is achieved by the rules

$$R_{check} = \{p_k \to E_i p_{k,i?}|_{Bp'_i}, p_{k,j}^{ncr} \to E_i p_{k,j}^{ncr}|_{Bp'_i}, p_k^{ns} \to E_i p_{k,i?}^{ns}|_{Bp'_i} \mid \text{ for}$$
$$1 \leq i, j, k \leq n, \text{ such that } d(p_i, p_k) < \epsilon\} \cup \{B \to C\}$$

where $d(p_i, p_k)$ denotes the distance between the locations of the ith and the kth points. The application of these rules produce a number of E_i objects which is equal to the number of points that are in the ϵ neighborhood of the ith point.

Now we mark the point corresponding to p'_i core or non-core, depending on the number of its ϵ neighbors with the following rules. If $m = MinPts$, then we have

$$R_{mark} = \{p'_i \to p_i^{cr}|_{C(E_i)^m} > p'_i \to p_i^{ns}|_C \mid 1 \leq i \leq n\} \cup$$
$$\{p_{j,i?} \to p'_{j,i}|_{Dp_i^{cr}}, p_{j,i?}^{ns} \to p'_{j,i}|_{Dp_i^{cr}} \mid 1 \leq i, j \leq n\} \cup$$
$$\{p_{j,i?} \to p_j|_{Dp_i^{ns}}, p_{j,i?}^{ns} \to p_j^{ns}|_{Dp_i^{ns}} \mid 1 \leq i, j \leq n\} \cup$$
$$\{E_i \to \varepsilon|_C \mid 1 \leq i \leq n\} \cup \{C \to D, D \to F\}$$

where the symbol $>$ shows the priority among the first group of rules in R_{mark}. The rules here are used for two consecutive steps: First, the chosen point is marked core or noise (based the number of points in its ϵ neighborhood) by changing p'_i to p_i^{cr} or to p_i^{ns} depending on the number of E_i symbols that are present (these were created in the previous step, their number corresponds to the number of neighbors). Second, if the point is marked core, its ϵ neighborhood is also added to this cluster (the cluster denoted by i).

The next group of rules serves to see if the recently created cluster (cluster i) can be expanded further.

$$\begin{aligned}R_{check2} = \{&p^{cr}_{j,i} \to H_i p^{cr}_{j,i}|_{Fp'_{k,i}}, p^{ncr}_{j,i} \to H_i p^{ncr}_{j,i}|_{Fp'_{k,i}},\ p^{ns}_{j} \to H_i p^{ns}_{j,i?}|_{Fp'_{k,i}},\\ &p_j \to H_i p_{j,i?}|_{Fp'_{k,i}} \mid \text{ for } 1 \leq i,j,k \leq n, \text{ such that } d(p_j, p_k) < \epsilon\} \cup\\ &\{F \to F'\}.\end{aligned}$$

These rules examine the neighborhood of the kth point which has recently been added to the cluster denoted by i. The number of H_i symbols is the same as the number of points in the ϵ neighborhood of point (k).

The next group of rules is the following.

$$\begin{aligned}R_{mark2} = &\{p'_{k,i} \to p^{cr}_{k,i}|_{F'(H_i)^m} > p'_{k,i} \to p^{ncr}_{k,i}|_{F'} \mid 1 \leq i,k \leq n\} \cup\\ &\{p_{j,i?} \to p'_{j,i}|_{F''p^{cr}_{k,i}}, p^{ns}_{j,i?} \to p'_{j,i}|_{F''p^{cr}_{k,i}} \mid 1 \leq i,j,k \leq n\} \cup\\ &\{p_{j,i?} \to p_j|_{F''p^{ncr}_{k,i}}, p^{ns}_{j,i?} \to p^{ns}_{j}|_{F''p^{ncr}_{i}} \mid 1 \leq i,j \leq n\} \cup\\ &\{H_i \to \varepsilon|_{F'} \mid 1 \leq i \leq n\} \cup \{F' \to F''\}\\ &\{F'' \to F|_{p'_{k,i}} > F'' \to A \mid 1 \leq i,k \leq n\},\end{aligned}$$

where $m = MinPts$, as before. Similarly to R_{mark}, these rules decide whether the cluster denoted by i should be expanded with the elements of the ϵ neighborhood of point (k). If the number of neighbors is sufficient, they are added to the cluster, and also marked for further investigation. If there are points which are newly added to the cluster, that is, if further neighborhood checks are necessary, then the symbol F' is changed to F'', and then back to F, so the rules in R_{check2} become applicable again, and the checking process can repeated. If no new points are added to the cluster, F'' is changed to A, so the rules in R_{check} are activated, and the search for additional clusters can start with the identification of a yet unclassified point by R_{pick}.

To see how the system Π operates, consider the initial configuration

$$Ap_1 \ldots p_n.$$

By applying a rule $Ap_i \to Bp'_i \in R_{pick}$ for some $1 \leq i \leq n$, a not yet classified point (i) is chosen from the point set $(1), \ldots, (n)$, and we obtain

$$Bp_1 \ldots p'_i \ldots p_n.$$

To show how the system works, we start with a more general case

$$Bx_1 \ldots p'_i \ldots x_n,$$

where $x_i \in \{p_i, p^{ns}_i, p^{cr}_i, p^{cr}_{i,j}, p^{ncr}_{i,j}\}$.

Now, because B is present, the rules of R_{check} are applicable, so we get

$$C \ldots p'_i \ldots y_1 E_i \ldots y_l E_i \ldots$$

where $y_j \in \{p_{k,i?}, p^{ns}_{k,i?} \mid \text{for some } 1 \leq k \leq n\}$, $1 \leq j \leq l$. All symbols which correspond to unclassified points or noise points (k) in the ϵ neighborhood of point (i) are marked as candidates for inclusion in the same cluster as (i) (denoted by the index $i?$).

If the number of neighbors (equal to the number of E_i symbols) is not sufficient (less than $MinPts$), we get

$$D \dots p_i^{ns} \dots y_1 \dots y_l \dots,$$

and then

$$Fx_1 \dots p_i^{ns} \dots x_n$$

by the rules of R_{mark}. Now F is changed to F', F'', and then back to A, when the rules of R_{pick} become applicable again, and the system continues with choosing an other point for examination.

Otherwise, if the number of neighbors of point (i) are sufficient, we get

$$D \dots p_i^{cr} \dots y_1 \dots y_l \dots,$$

and then

$$F \dots p_i^{cr} \dots p'_{X_1,i} \dots p'_{X_l,i} \dots,$$

where $1 \leq X_j \leq n$, $1 \leq j \leq l$, marking the point (i) as core point, and marking its neighbors as members of the cluster denoted by i.

Now, the newly added points $(X_1), \dots, (X_l)$, corresponding to the symbols $p'_{X_1,i}, \dots, p'_{X_l,i}$ have to be checked, which is done by the rules of R_{check2} If the ϵ neighborhood of a point (k) contains a sufficient number of points, then similarly to R_{check}, the rules of R_{check2} introduce a sufficient number of H_i symbols for the rule $p'_{k,i} \to p^{cr}_{k,i}$ to be applicable, and point (k) is marked as a core point, denoted by a corresponding symbol $p^{cr}_{k,i}$. Otherwise, if the number of neighbors is not sufficient, then point (k) is marked as non-core, denoted by the symbol $p^{ncr}_{k,i}$ introduced by the rule $p'_{k,i} \to p^{ncr}_{k,i}$.

These checks are executed in parallel for all $p'_{X_k,i}$, resulting in marking some of them core, some of them non-core, and priming all neighbors of core points in two computational steps, using the rules in R_{check2} and R_{mark2}. If the result contains primed points, that is, points that have to be checked by counting the number of their neghbors, then F'' is rewritten to F, so the process can start all over again, otherwise it is rewritten to A, meaning that the cluster denoted by i cannot be expanded any more, the search for new clusters can begin by picking a new point using the rules of R_{pick}.

If all points are classified, then no rule of R_{pick} can be used, the system halts in a configuration containing the symbol A, and with all points (i), $1 \leq i \leq n$, having a corresponding symbol, which is either

- p_i^{cr} : point (i) is a core point, belonging to the cluster denoted by i,
- $p_{i,j}^{cr}$: point (i) is a core point, belonging to the cluster denoted by j,
- $p_{i,j}^{ncr}$: point (i) is a non-core point, belonging to the cluster denoted by j,
- p_i^{ns} : point (i) is a noise point, it does not belong to any cluster.

4 Conclusion

We have shown how to implement the DBSCAN clustering algorithm on P systems. The model we used worked with evolution rules, promoters and priorities, Due to the parallel nature of P systems, our implementation has a time complexity of $O(n)$ which is clear improvement compared to the time complexity of $O(n^2)$ of a sequential DBSCAN implementation. Our P system implementation presented in in this paper is not the first one, but we believe that it is conceptually more simple than the implementation presented in [11].

Considering the parallelity of our approach, the points of the dataset are examined by the algorithm one-by-one, but the number of neighbors of the examined points is calculated in parallel. Moreover, all points examined in step 3 of the algorithm (see the introduction for the numbering of the steps of the algorithm) are examined in parallel which further reduces the time complexity of the P system implementation. An interesting challenge would be to find a way in which the parallelity of the algorithm can further be increased, and thus, the time complexity further reduced.

Acknowledgments

This research is supported in part by project no. K 120558 of the National Research, Development and Innovation Fund of Hungary, financed under the K 16 funding scheme, and by the EFOP-3.6.1-16-2016-00022 project, co-financed by the European Union and the European Social Fund.

References

1. Ciobanu, G., Pérez-Jiménez, M.J., Păun, G. (eds.): Applications of Membrane Computing. Natural Computing Series, Springer (2006)
2. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231. KDD'96, AAAI Press (1996)
3. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. (2011)
4. Jiang, Y., Peng, H., Huang, X., Zhang, J., Shi, P.: A novel clustering algorithm based on p systems. International journal of innovative computing, information & control: IJICIC 10, 753–765 (01 2014)
5. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108 – 143 (2000)
6. Păun, G.: Membrane Computing: An Introduction. Springer-Verlag, Berlin, Heidelberg (2002)
7. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010)

8. Xu, J., Liu, X., Xue, J.: Cluster analysis by a class of splicing p systems. In: Park, J.J.J.H., Pan, Y., Kim, C.S., Yang, Y. (eds.) Future Information Technology. pp. 575–581. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
9. Xue, J., Liu, X.: A k-nearest based clustering algorithm by P systems with active membranes. JSW 9(3), 716–725 (2014), `https://doi.org/10.4304/jsw.9.3.716-725`
10. Zhang, G., Pérez-Jiménez, M.J., Gheorghe, M.: Real-life Applications with Membrane Computing. Springer Publishing Company, Incorporated (2017)
11. Zhao, Y., Liu, X., Li, X.: An improved DBSCAN algorithm based on cell-like P systems with promoters and inhibitors. PLoS ONE 13, e0200751 (Dec 2018)