Geo-spatial Clustering with User-Specified Constraints. *

Anthony K. H. Tung †
Simon Fraser U.
khtung@cs.sfu.ca

Raymond T. Ng
U. of British Columbia
rng@cs.ubc.ca

Laks V.S. Lakshmanan
IIT, Bombay & Concordia U.
laks@cs.concordia.ca

Jiawei Han
Simon Fraser U.
han@cs.sfu.ca

Abstract

Capturing application semantics and allowing a human analyst to express his focus in mining have been the motivation for several recent studies on constrained mining. In this paper, we introduce and study the problem of constrained clustering—finding clusters that satisfy certain user-specified constraints. We argue that this problem arises naturally in practice. Two types of constraints are discussed in this paper. The first type of constraints are imposed by physical obstacles that exist in the region of clustering. The second type of constraints are SQL constraints which every cluster must satisfy. We provide a prelimary introduction to both types of constraints and discuss some techniques for solving them.

1 Introduction

Cluster analysis, which groups data for finding overall distribution patterns and interesting correlations among data sets, has numerous applications in pattern recognition, spatial data analysis, image processing, market research, etc. Cluster analysis has been an active area of research in computational statistics and data mining, with many effective and scalable clustering methods developed recently.

These methods can be categorized into partitioning methods [KR90, NH94, BFR98], hierarchical methods [KR90, ZRL96, GRS98, KHK99], density-based methods [EKSX96, ABKS99, HK98], grid-based methods [WYM97, SCZ98, AGGR98], and model-based methods [SD90, Fis87, CS96, Koh82]. In the context of GIS, cluster analysis can be very useful in identifying groups of similar points on the map and performing detail analysis of each group. This can be useful for tasks like facilities planning since a facility can then be allocated to serve each group of objects separately.

Unfortunately, the task of planning the location of facilities is usually quite complicated since users could like to enforce some constraints when performing such a task. One possible constraint might be due to the existence of obstacles in the clustering region. Let us illustrate this with an example.

Example 1.1 *A bank manager wishes to locate 4 ATMs in the area shown in Figure 1a to serve the customers who are represented by points in the figure. In such a situation however, obstacles may exist in the area which should not be ignored. This is because ignoring these obstacles will result in clusters like those in Figure 1b which are obviously wrong. Since cluster C_1 for example is split by a river, some customers on one side of the river will have to travel a long way to the allocated ATM on the other side of the river.* □

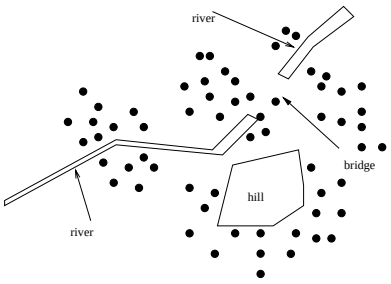
Besides constraints imposed by obstacles, users can also face constraints due to operational requirement as follows.

Example 1.2 Consider a package delivery company which is seeking to use a GIS to help de-

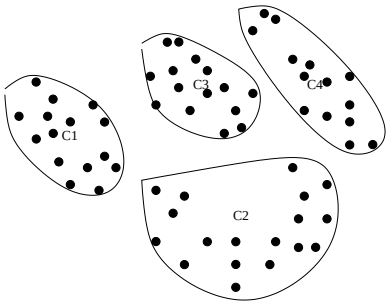
* Research was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada, and grants NCE:IRIS3 and NCE:GEOID from the Networks of Centres of Excellence of Canada.

† Person handling correspondence. Postal Address: Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, B. C., Canada.

Proceedings of the International Workshop on Multimedia Data Mining (MDM/KDD'2000), in conjunction with ACM SIGKDD conference. Boston, USA, August 20, 2000
(O.R. Zaïane, S.J. Simoff, eds.)



(a) Customers' location and obstacles.



(b) Clusters formed when ignoring obstacles.

Figure 1: Planning the location of ATMs

termine the locations for k service stations in a city. Suppose the GIS contains the information of customers based on the scheme: $customer(Name$, $AddrXcoord$,$AddrYcoord$, $MemberType$, $AvgMonthChg)$. The company may formulate this location selection problem as an instance of the clustering problem, using the address fields $AddrXcoord$ and $AddrYcoord$ to define the distance function $df()$.

Suppose further that the company has two kinds of customers in consideration: *gold* customers, who need frequent, regular services, and *ordinary* customers, who require occasional services. In order to save the cost and provide good service, the manager may add the following constraints: (1) that each station should serve at least 50 gold customers; and (2) that each station should serve at least 5000 ordinary customers. With the constraints, this becomes an instance of the constrained clustering problem. □

As can be seen, the problem of constrained clustering is a very practical problem faced by the users who are not given ways to specify the type of clusters that they want to discovered. In view of this, we introduce the notion of constrained clustering in this paper and introduce some initial work which is being done to address these problems. The organization of the rest of this paper is as follows. In the next section, we will give an introduction to the problem of clustering with obstacles entities. We will described techniques which are used to improve the scalability of our algorithm when obstacle constraints are taken into consideration. In Section 3, we will look at the problem of clustering with SQL aggregate constraints and discuss some prelimary work on clustering with such constraints. We will conclude our paper with Section 4.

2 Clustering with Obstacle Entities (COE)

In order to solve the problem shown in Example 1.1, let us first formally defined the problem as follows.

Definition 2.1 *We are given a set P of n points $\{p_1, p_2, ..., p_n\}$ and a set O of m* **non-intersecting** *obstacles $\{o_1, ..., o_m\}$ in a two dimensional region, R. Each obstacle o_i is represented by a simple polygon with $o_i.nv$ sides and each vertex of the polygon is denoted as $o_i.v_j$, $1 \leq j \leq o_i.nv$. The distance, $df(p,q)$ between any two points, p and q is defined as the length of the shortest Euclidean path from p to q without cutting through any obstacles. To distinguish this distance from the direct Euclidean distance, we will refer to this distance as* obstructed distance *in this paper. Our objective is to partition P into k clusters $Cl_1, ..., Cl_k$ such that the following square-error function, E, is minimized:*

$$E = \sum_{i=1}^{k} \sum_{p \in Cl_i} d^2(p, m_i)$$

In order to solve the above problem, a trivial solution is to argue that obstacles in effect only cause a change in the distance function and thus can be hidden from the actual clustering algorithm by simply providing a different distance function call to it. However our work in [THH00] shows that

a clustering algorithm which takes these obstacles into consideration can in fact be optimized to improve clustering efficiency.

In [THH00], we developed a clustering algorithm called COE-CLARANS to handle clustering with obstacles. COE-CLARANS is an improved version of CLARANS in [NH94] which is a k-medoid clustering algorithm. The CLARANS algorithm first randomly chooses k objects as the set of cluster centers, *current*. It then assigns the rest of the objects to the nearest cluster center and compute the square-error function E for the initial solution. A search is then done for a better solution by taking each cluster center following randomize order and trying to replace it with another randomly selected object not in *current*. If a better solution is found, i.e,. a lower value of E is computed for the new solution, *current* is set to the new solution and the whole process is repeated with the new *current*. For each cluster center, the attempt to find a better solution by center replacment is repeated *maxneighbor* times and the best solution is kept. If no better solution is found after *maxneighbor* attempts on all the k cluster centers, it is concluded that a local minima is reached. This process repeats *numlocal* times and the best local minima that is found will be output as the solution.

There are however certain issues which must be addressed in order to adopt CLARANS to cluster objects with obstacle constraints imposed. As can be seen, CLARANS is a generate-and-test algorithm which frequently recompute the square-error function E for testing a generated solution. To perform this operation, a scan must be done through the n objects to compute their distance from their cluster center. If the objects are stored in secondary storage, high I/O cost will be incurred. Furthermore, since the solution is generated by randomly picking another object to replace a cluster center, there is a good chance that it is not a better solution and thus does not justify the time spent on computing E. In the case of clustering with obstacles, such overhead is even higher as the obstacles have complicated the distance function. In order to overcome these problems, the following two approaches are adopted.

First, a pre-clustering step similar to those in BIRCH [ZRL96], ScaleKM [BFR98] and CHAMELEON [KHK99] is taken to group the objects into a set of *micro-clusters*. Ganti et. al. in [GGR99] gives an analogy to pre-clustering as follow:

> "... if each data point is a marble on a table top, we replace clusters of marbles by tennis balls and then look for clusters of tennis balls."

A *micro-cluster* is the tennis ball in the analogy. It is a group of points which are so close to each other that they are very likely to belong to the same cluster. To compress the data set, a point from each micro-cluster is selected to represent the micro-cluster. Since the size of these representative points is much smaller than the actual data set, they could be clustered using the COE-CLARANS algorithm in the main memory. To facilitate the clustering, information about the micro-cluster are stored together with the representative points. This information would include statistic like the number of points in the micro-cluster, the diameter of the micro-cluster, etc.

Second, to avoid the unnecessary computation of the square-error function E, an initial **lower bound** of E, E', is first computed. If E' is already higher than the best solution so far, then the generated solution can never be better than the best solution and thus can be abandoned without the need for E to be computed. To compute E', we underestimate the distance between the randomly chosen center o_{random} and the micro-clusters by using direct Euclidean distance instead of the obstructed distance. By doing so, each micro-clusters so formed will fall into one of the following categories:

1) p is correctly assigned to o_{random}.
Since the direct Euclidean distance between p and o_{random} must be shorter than the obstructed distance between p and o_{random}, we have underestimate the actual distance between p and o_{random}.

2) p is wrongly assigned to o_{random}.
Let o_i be the cluster center that p should rightfully be assigned to. Since p is assigned to o_{random} instead, the direct Euclidean distance between p

and o_{random} must be shorter than the obstructed distance between p and o_i which is computed before the iteration begins. Thus, we have underestimated the actual distance between p and o_i.

3) p is not assigned to o_{random}.
Since the obstructed distance of p to the rest of the $k-1$ cluster centers o_j is computed before the iteration begin, the distance used to compute E' must be correct.

As we can see, for all the three categories, we either underestimate or compute correctly the obstructed distance of a micro-cluster p to its nearest cluster center. As such E' must be a lower bound for the actual square-error function E.

By adopting the above two approaches, we are able to make our algorithm scalable for a large number of objects and a moderate number of obstacles. We illustrate the difference between clustering with obstacles and without obstacles in Figure 2. Further details of our work in this area can be found in [Hou99].

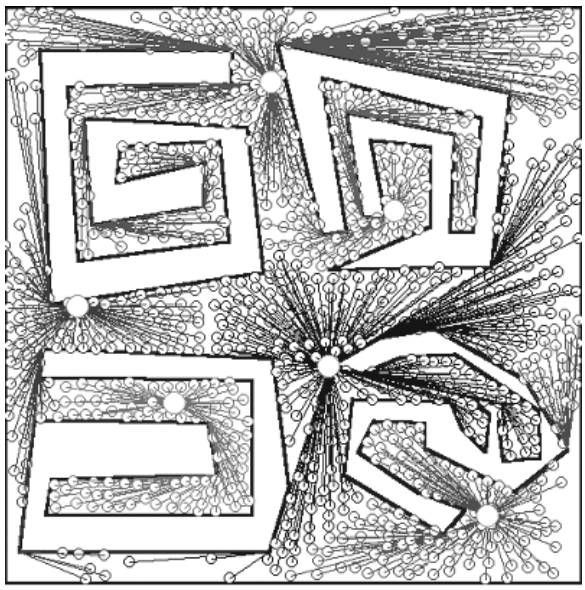
3 Clustering Under SQL Aggregate Constraints

In order to handle the type of constraints that we seen in Example 1.2, we look into the problem of clustering under SQL aggregate constraints in [TNLH00]. We define *SQL aggregate constraints* as follows.

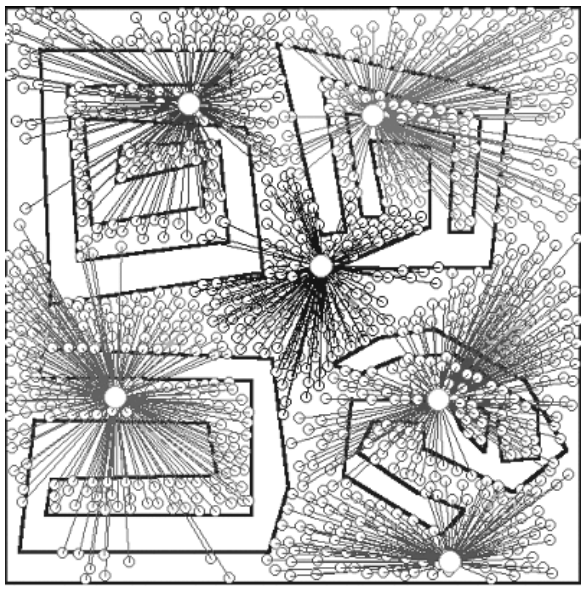
Definition 3.1 (SQL Aggregate Constraints) Let each object p_i in the database D be associated with a set of m attributes $\{a_1, \ldots, a_m\}$. The value of an attribute a_j of an object p_i is denoted as $p_i[a_j]$.

Let the aggregate functions $agg_1 \in \{max(), min(), avg(), sum()\}$ and $agg_2 \in \{count()\}$. Let θ be a comparator function, i.e., $\theta \in \{<, \leq, \neq, =, \geq, >\}$, and c represent a numeric constant. Given a cluster Cl, an SQL aggregate constraint on Cl is a constraint in one of the following forms: (i) $agg_1(\{p_i[a_j] \mid p_i \in Cl\})\ \theta\ c$; or (ii) $agg_2(Cl)\ \theta\ c$. □

While solving some of these SQL constraints can be rather complicated, a large number of them could however be reduced to a type of constraints called *existential constraints* defined as follows.



(a) Clustering when considering obstacles.



(b) Clustering when Ignoring Obstacles.

Figure 2: How Obstacles affect clusters.

Definition 3.2 (Existential Constraints) Let $W \subset D$ be any subset of objects. We often call them **pivot** objects. Let c be a positive integer. An **existential constraint** on a cluster Cl is a constraint of the form: $count(\{p_i | p_i \in Cl, p_i \in W\}) \geq c$. □

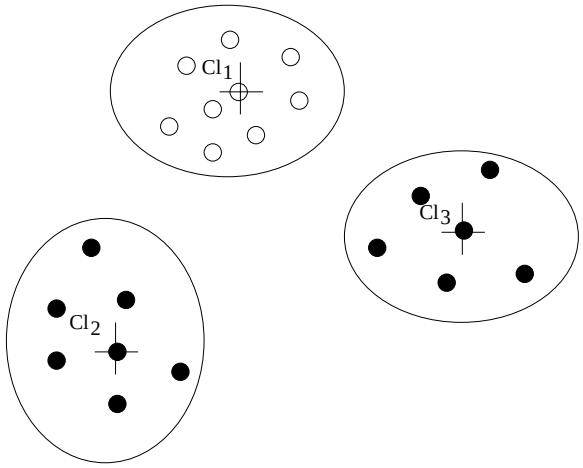
By examining the class of SQL constraints that we have defined, we can see that some of the SQL constraints can be easily reduced to an existential constraint. For example, "$count(Cl) \geq c$" is in fact a special case of existential constraints in which all objects are pivot objects. Similarly, a constraint like "$max(\{p_i[a_j] \mid p_i \in Cl\}) \geq d$" can also be reduced to an existential constraint in which the pivot objects are in the set $\{p_i | p_i[a_j] \geq d\}$ and each cluster must contain more than one pivot object.

Becauses of its importance, we focus on solving the constrained clustering involving in one existential constraint in [TNLH00]. More specifically, our problem definition is as below.

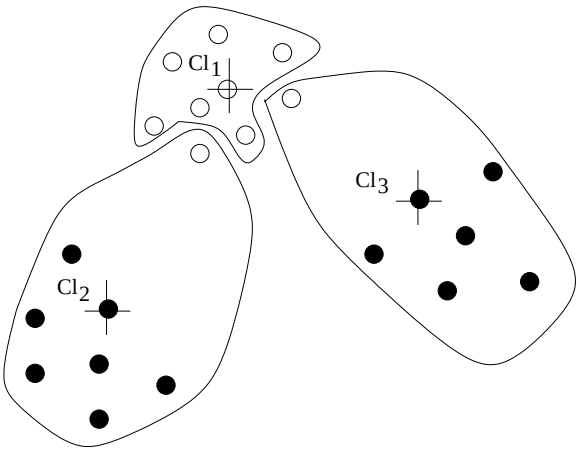
Definition 3.3 The Constrained Clustering (CC) Problem *Given a data set D with n objects, a distance function $df : D \times D \longrightarrow \Re$, a positive integer k, and* an existential constraints EC, *find a* k-clustering $(Cl_1, \ldots, Cl_k)$ *such that* $DISP = (\sum_{i=1}^{k} disp(Cl_i))$ *is minimized, and* each cluster Cl_i satisfies the constraint EC, *denoted as* $Cl_i \models \mathcal{C}$.

The "dispersion" or "square-error" of cluster Cl_i, $disp(Cl_i)$, measures the total distance between each object in Cl_i and some *representative* rep_i of Cl_i, i.e., $disp(Cl_i)$ defined as $\sum_{p \in Cl_i} df(p, rep_i)$. Typically, these representatives are the centroids or the medoids of the clusters which will minimize the dispersion of each cluster and thus their locations are good candidates for locating the facilities that serve the clusters.

With the introduction of an existential constraint, one major complication is that instead of being assigned to the nearest center, a pivot object might be assigned to a cluster center which is further away because of the need to satisfy the existential constraint. Let us consider the example shown in Figure 3a. In the figure, the hollow points represent pivot objects while the solid points are non-pivot objects. Without any constraint imposed on the clustering, a natural way to group the points is shown in Figure 3a. However if we impose a constraint that each cluster must at least contain one pivot point, then a solution could be in Figure 3b where one pivot point is "forced" to be in cluster Cl_2 and one in Cl_3 although both these points are actually nearer to the center of cluster Cl_1. Because of this, the constraint k-means algorithm which we introduce in [TNLH00] first tries to satisfy user-specified constraint before trying to refine the clusters by swapping objects between the clusters. In order for the clusters to be valid after the refinement, the swapping of the an object is only done if the change in membership of the object does not invalidate the user-specified constraint. More details of the algorithm can be obtained from [TNLH00].



(a) Clustering without constraints.



(b) Clustering with Constraints.

Figure 3: How an existential constraint affects clusters.

4 Conclusion

In this paper, we have introduced and studied the problem of having user-specified constraints in geo-spatial clustering. Even though constrained clustering problems arise naturally in practice, this appears to be the first attempt to tackle these problems. Two types of constraints are discussed in this paper. The first type of constraints are imposed by physical obstacles that exist in the region of clustering. The second type of constraints are SQL constraints which every cluster must satisfy. We discuss some techniques for solving these two types of constraints and hope that more work will be done in these area.

References

[ABKS99] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Conf. on Management of Data (SIGMOD'99)*, pages 49–60, Philadelphia, PA, June 1999.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, Seattle, Washington, June 1998.

[BFR98] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 9–15, New York, NY, August 1998.

[CS96] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, Oregon, August 1996.

[Fis87] D. Fisher. Improving inference through conceptual clustering. In *Proc. 1987 AAAI Conf.*, pages 461–465, Seattle, Washington, July 1987.

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining very large databases. *COMPUTER*, 32:38–45, 1999.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 73–84, Seattle, Washington, June 1998.

[HK98] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 58–65, New York, NY, August 1998.

[Hou99] J. Hou. *Clustering with Obstacle Entities*. M.Sc. Thesis, Simon Fraser University, Canada, December 1999.

[KHK99] G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *COMPUTER*, 32:68–75, 1999.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[NH94] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 144–155, Santiago, Chile, September 1994.

[SCZ98] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. 1998 Int. Conf. Very Large Data Bases (VLDB'98)*, pages 428–439, New York, NY, August 1998.

[SD90] J.W. Shavlik and T.G. Dietterich. *Readings in Machine Learning*. Morgan Kaufmann, 1990.

[THH00] A. K. H. Tung, J. Hou, and J. Han. COE: Clustering with obstacles entities, a preliminary study. In *Proc. 4th Pacific-Asia Conf.on Knowledge Discovery and Data Mining (PAKDD'00)*, Kyoto, Japan, 18-20, Apr. 2000.

[TNLH00] A. K. H. Tung, R. Ng, L. Lakshmanan, and J. Han. Constraint-based clustering in large database. In *Submitted to ICDT'00*, Jun. 2000.

[WYM97] W. Wang, J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proc. 1997 Int. Conf. Very Large Data Bases (VLDB'97)*, pages 186–195, Athens, Greece, Aug. 1997.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, Montreal, Canada, June 1996.